Lam Thu Bui   Yew Soon Ong
Nguyen Xuan Hoai   Hisao Ishibuchi
Ponnuthurai Nagaratnam Suganthan (Eds.)

LNCS 7673

Simulated Evolution and Learning

9th International Conference, SEAL 2012
Hanoi, Vietnam, December 2012
Proceedings

Springer

Lam Thu Bui   Yew Soon Ong
Nguyen Xuan Hoai   Hisao Ishibuchi
Ponnuthurai Nagaratnam Suganthan (Eds.)

Simulated Evolution and Learning

9th International Conference, SEAL 2012
Hanoi, Vietnam, December 16-19, 2012
Proceedings

Springer

Volume Editors

Lam Thu Bui
Le Quy Don Technical University, Faculty of Information Technology
100 Hoang Quoc Viet Street, Cau Giay District, Hanoi, Vietnam
E-mail: lam.bui07@gmail.com

Yew Soon Ong
Nanyang Technological University, School of Computer Engineering
Block N4, 2b-39, Nanyang Avenue, Singapore 639798, Singapore
E-mail: asysong@ntu.edu.sg

Nguyen Xuan Hoai
Hanoi University, HANU IT Research and Development Center
9th Km Nguyen Trai Road, Hanoi, Vietnam
E-mail: nxhoai@hanu.edu.vn

Hisao Ishibuchi
Osaka Prefecture University, Graduate School of Engineering
1-1 Gakuen-cho, Nakaku, Sakai, Osaka 599-8531, Japan
E-mail: hisaoi@cs.osakafu-u.ac.jp

Ponnuthurai Nagaratnam Suganthan
Nanyang Technological University, School of Electrical and Electronic Engineering
Block S2, B2a-21, Nanyang Avenue, Singapore 639798, Singapore
E-mail: epnsugan@ntu.edu.sg

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-34858-7 e-ISBN 978-3-642-34859-4
DOI 10.1007/978-3-642-34859-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012951174

CR Subject Classification (1998): F.1.1, I.2.6, I.6, G.1.6, H.3, D.2.2, J.3-4

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

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

Printed on acid-free paper

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

Preface

This volume contains the papers that were carefully selected for publication in these proceedings and presented at the 9th Simulated Evolution and Learning (SEAL2012) Conference held during December 16–19, 2012, at the Le Quy Don Technical University (LQDTU), Vietnam. SEAL has been an international forum for researchers discussing issues related to evolutionary optimization and machine learning. This biennial event started in Seoul, South Korea, in 1996 and was thereafter held in Canberra, Australia, in 1998, Nagoya, Japan, in 2000, Singapore, in 2002, Busan, South Korea, in 2004, Hefei, China, in 2006 and Melbourne, Australia, in 2008, and Kanpur, India, in 2010.

SEAL 2012 continued to maintain its high technical quality with a rigorous reviewing process of an international Program Committee. This year, SEAL 2012 received 91 paper submissions from 20 countries. After a rigorous peer-review process with three reviews per paper, 50 papers were accepted for presentation at the conference. The papers cover a wide range of topics in simulated evolution and learning. The accepted papers have been classified into the following main categories: (a) theoretical developments, (b) evolutionary algorithms, (c) swarm intelligence, (d) data mining, (e) learning methodologies, and (f) real-world applications.

The conference featured five distinguished keynote speakers. Hussein Abbass's talk on "Computational Red Teaming: Can Evolution and Learning Augment Human Behaviour?" focused on computational red teaming (CRT), a field that attempts to create a form of artificial intelligence (AI) whereby intelligence is measured as the ability of a computer environment to challenge humans. Kay Chen Tan's talk on "Advances in Evolutionary Multi-objective Optimization" showcased the incorporation of probabilistic graphical approaches in evolutionary mechanism that may enhance the iterative search process when interrelationships of the archived data have been learned, modeled, and used in the reproduction for multi-objective optimization.

Hisao Ishibuchi's talk on "Fuzzy Genetics-Based Machine Learning" discussed the use of genetic-based machine learing for single and multi-objective fuzzy rule-based classifier design. Yew Soon Ong's talk on "Towards a Unified Evolutionary and Memetic Search Model" presented a balance between generality (exploration through stochastic variation) and problem specificity (exploitation through lifetime learning). Kok Lay Teo's talk on "Optimal Discrete-Valued Control Computation: An Exact Penalty Function Approach" considered an optimal control problem in which the control takes values from a discrete set.

SEAL 2012 could not have been held successfully without the contributions and support of many people. We would like to express our sincere thanks to all members of the conference committees, authors, participants, the local organizing teams, and the sponsors. We are grateful the LQDTU for supporting our cause and encouraging us to organize the conference at LQDTU.

September 2012

Lam Thu Bui
Yew Soon Ong
Nguyen Xuan Hoai
Hisao Ishibuchi
P.N. Suganthan

Organization

Honorary Chair	The Long Pham
General Chair	Lam Thu Bui
Local Chair	Long Thanh Ngo
Competition Chair	Kai Quin

Program Chairs

Yew Soon Ong	Thanh Tinh Dao
Nguyen Xuan Hoai	Bao Son Pham

Technical Co-chairs

Hisao Ishibuchi	Kay Chen Tan
P.N. Suganthan	Juergen Branke

Steering Committee

Takeshi Furuhashi	Mengjie Zhang
Jong-Hwan Kim	Bob McKay
Lipo Wang	Xiaodong Li
Xin Yao	Kalyanmoy Deb

Publicity Chairs

Sung-Bae Cho
Jing Liu
Meng Hiot Lim

Program Committee

Tapabrata Ray	Kai Qin
Rong Qu	Pramod Singh
Adam Ghandar	Andre de Carvalho
Bo Liu	Yusuke Nojima
Uday Chakraborty	Gurunathan Saravana Kumar
Martin Holena	Meinolf Sellmann

Luis Paquete
Guenther Raidl
Pavel Kromer
Hussein Abbass
Jose A. Lozano
Jing Liu
Martin Pelikan
Nugroho Fredivianus
Shuming Wang
Oliver Schuetze
Amiram Moshaiov
Jun Zhang
Lothar Thiele
Sung-Bae Cho
Carlos M. Fonseca
Varun Aggarwal
Jason Teo
Christian Grimme
Khoi Le
Hartmut Schmeck
Antonio Gaspar-Cunha
Rafal Drezewski
Marc Schoenauer
Swagatam Das
Frédéric Lardeux
Yaochu Jin
Huey-Ming Lee
Mengjie Zhang
Joanna Kolodziej
Quang Uy Nguyen
Hiroyuki Sato
Ashutosh Tiwari
Gustavo Recio
Zhun Fan
Lam Thu Bui
Andries Engelbrecht
Efrén Mezura-Montes
Jin-Kao Hao
Robert Lygoe
Quang Huy Nguyen
El-Ghazali Talbi
Madan Dabbeeru
Kalyan Veeramachaneni
Gary Yen
Patrick Siarry
Rolf Drechsler
Helio Barbosa
Lino Costa
Akira Namatame
Laetitia Jourdan
Kalyanmoy Deb
Thomas Philip Runarsson
Christopher Turner
Hirosato Seki
Zexuan Zhu
Long Thanh Ngo
Xin Yao
Tru Cao
Matthieu Basseur
Huynh Thi Thanh Binh
Xiaodong Li
Sanaz Mostaghim
Jesper Hattel
Quang Anh Tran
Dario Landa-Silva
Luis Martí
Marzuki Khalid
Hiroaki Ishii
Jonathan Chan
Daryl Essam
Jaeseok Choi
Martin Middendorf
Henrik Saxen
Julian Molina
Maoguo Gong
Sushil Louis
Lily Rachmawati
Vineet Khare
Anna Kononova
Dimo Brockhoff
Kyriakos Giannakoglou
Piotr Wozniak
Ling Wang
Kay Chen Tan
Yifeng Zeng
C.P. Lim
Mario Koeppen

Table of Contents

The Influence of the Number of Initial Feasible Solutions on the Performance of an Evolutionary Optimization Algorithm

Saber M. Elsayed, Ruhul A. Sarker, and Daryl L. Essam

School of Engineering and Information Technology,
University of New South Wales at Australian Defence Force Academy,
Canberra 2600, Australia
{s.elsayed,r.sarker,d.essam}@adfa.edu.au

Abstract. Constrained optimization is a well-known research topic in the evolutionary computation field. In these problems, the selected solution must be feasible. In evolutionary constrained optimization, the search space is usually much bigger than the feasible space of the problem. There is a general view that the presence or absence of any feasible individuals in the initial population substantially influences the performance of the algorithm. Therefore, the aim of this research is to analyze the effect of the number of feasible individuals, in the initial population, on the algorithm's performance. For experimentation, we solve a good number of well-known bench-mark problems using a Differential Evolution algorithm. The results show that the algorithm performs slightly better, for the test problems solved, when the initial population contains about 5% feasible individuals.

Keywords: Constrained optimization, differential evolution, feasible individual.

1 Introduction

Constrained optimization is an important research area as there is a huge number of real-world decision processes that require the solution of Constrained Optimization Problems (COPs). Evolutionary Algorithms (EAs), such as Differential Evolution (DE) [1, 2], and Genetic Algorithms (GA) [3, 4], have a long history of successfully solving COPs.

In any EAs, the initial step is to generate a population of individuals randomly, which will then be evolved in the later generations. As the search space is always bigger than the feasible space, the initial population may contain many infeasible individuals. The number of feasible individuals can even be zero when the feasible space is tiny in comparison to the search space. It is the general view that the presence or absence of the feasible individuals, in the initial population, affects the performance of EAs. The presence of any feasible individuals can bias the entire search process towards feasibility. Note that for any optimal solution the feasibility condition must be satisfied. Therefore, many researchers attempted to use heuristics methods, such as Sequential Quadratic Programming (SQP) [5], with the hope of quickly finding feasible individuals during the evolution process.

In this paper, we attempt to analyze the effect of the number of feasible individuals, in the initial population, on the performance of a DE algorithm. This analysis will help to judge both whether the initial population should include feasible individuals when there is none and also whether to increase the number when there are very few. These findings should attract further research on the topic.

To the best of our knowledge, no such research has appeared in the literature. However, there are some research papers that recognize the importance of feasibility or feasible individuals during the evolution process. Singh et al. [6] proposed a GA that preserves the infeasible solutions that are closer to the constraint boundaries. This is because when these infeasible solutions are combined with the feasible solutions, they produce solutions either on or closer to the constraint boundary. Although this process accelerates the rate of convergence, it cannot be done unless the feasible solutions are present. They proposed to use only 5% of the promising infeasible individuals in their algorithm. With a similar recombination approach, Mezura-Montes *et al.*[7] proposed to keep some of the infeasible points if they are better than their parent with a predefined probability. As of the authors, this approach aimed at maintaining a reasonable diversity of the population. Other studies have also analyzed information exploitation from the feasible and infeasible solutions, such as [8-10]. In another study that can be related to the existing research, Barkat-Ullah *et al.* [11] proposed a Search Space Reduction Technique (SSRT) as an initial step of GA. SSRT directed the selected infeasible agents in the initial population to move towards the feasible space. The algorithm showed good performance when it was tested on a number of test problems and a real world case problem. The algorithm has also been successfully extended and used in another study [12].

The organization of this paper is as follows: the next section provides a brief review of DE. Section 3 and 4 discusses the design of experiments and the results, respectively. Finally, the conclusions are presented in section 4.

2 Differential Evolution

In this section, an introduction to DE with a review of its operators is presented. In DE, an initial population is generated, and for each parent vector from the current population (target vector), a mutant vector (donor vector) is obtained. Finally, an offspring is formed by combining the donor with the target vector. A comparison is then made between each parent and its offspring, with the better being copied to the next generation [2]. DE usually converges fast, incorporates a relatively simple and self-adapting mutation, and the same settings can be used for many different problems [2]. It performs well when the feasible patches are parallel to the axes [13]. However, DE prematurely converges when dealing with a multimodal fitness function because it loses its diversity [14, 15].

2.1 Mutation

This operation enables DE to explore the search space and maintain diversity. The simplest form of this operation is that a mutant vector is generated by multiplying an amplification factor, F, by the difference between two random vectors and the result is added to a third random vector (DE/rand/1) [1, 2], as:

$$\vec{V}_{z,t} = \vec{x}_{r_1,t} + F \times \left(\vec{x}_{r_2,t} - \vec{x}_{r_3,t}\right) \tag{1}$$

where r_1, r_2, r_3 are random numbers (1,2, ..., PS), $r_1 \neq r_2 \neq r_3 \neq z$, x a decision vector, PS the population size, F a positive control parameter for scaling the DV and t the current generation.

2.2 Crossover

The DE family of algorithms usually depends on two crossover schemes, exponential and binomial, which are briefly discussed below.

In an exponential crossover, firstly, an integer, l, is randomly chosen within the range [1, D] and acts as a starting point in the target vector from where the crossover or exchange of components with the donor vector starts. Another integer, L, is also chosen from the interval [1, D] and denotes the number of components that the donor vector actually contributes to the target. After the generation of l and L, the trial vector is obtained, as:

$$u_{z,j,t} = \begin{cases} v_{z,j,t} & for\ j = \langle l \rangle_D, \langle l+1 \rangle_D, \dots, \langle l+L-1 \rangle_D \\ x_{z,j,t} & for\ all\ other\ j \in [1, D] \end{cases} \tag{2}$$

where $j = 1,2,\dots,D$, and the angular brackets, $\langle l \rangle_D$, denote a modulo function with a modulus of D, and a starting index of l.

The binomial crossover is performed on each of the j^{th} variables whenever a randomly picked number (between 0 and 1) is less than or equal to a crossover rate, Cr. In this case, the number of parameters inherited from the donor has a (nearly) binomial distribution, as:

$$u_{zj,t} = \begin{cases} v_{zj,t}, & if\ (rand \leq Cr\ or\ j = jrand) \\ x_{zj,t}, & otherwise \end{cases} \tag{3}$$

where $rand \in [0,1]$, and $j_{rand} \in [1,2,\dots,D]$ is a randomly chosen index which ensures $\vec{U}_{z,t}$ receives at least one component from $\vec{V}_{z,t}$.

3 Design of Experiments

In this section, we discuss the designs of experiments, test problems and our parameters settings.

We run experiments with different percentages of feasible individuals in the initial population, while solving a set of well-known COPs. The percentages of feasible and infeasible individuals considered are as follows.

1. 100% infeasible individuals in the initial population.
2. 99% infeasible individuals and 1% feasible individuals.
3. 95% infeasible individuals and 5% feasible individuals.
4. 90% infeasible individuals and 10% feasible individuals.
5. 85% infeasible individuals and 15% feasible individuals.
6. 80% infeasible individuals and 20% feasible individuals.

For simplicity, these experiments are named as DE (0%), DE (1%), DE (5%), DE (10%), DE (15%) and DE (20%). For these experiments, we maintain a pool of 100 infeasible and 20 feasible individuals for each test problem. These individuals were generated randomly only once. For each experiment listed above, we choose the required number of individuals randomly from the respective pools.

The DE algorithm, used in this study, uses a binomial crossover and a simple mutation strategy, as follows:

$$u_{zj,t} = \begin{cases} x_{\varphi j,t} + F \times \left(x_{r_1 j,t} - x_{r_2 j,t}\right), & if\ (rand \leq Cr\ or\ j\ =\ jrand) \\ x_{zj,t}, & otherwise \end{cases} \tag{4}$$

where φ is a random integer number within the range [10%, 50%] of PS, Cr is set at 0.95 and PS at100 individuals. These settings are based on [16].

In this research, for experimentation, we have used a set of 22 small scale problems and 2 large scale optimization problems, as discussed below.

Small scale problems: Eighteen test problems are used from [17]. Each problem had 25 independent runs, with the stopping criterion for each being 240k fitness function evaluations (FFEs). It should be mentioned here that g02, g12, g19 and g04 are not considered in this study as their feasible regions are large and hence there is no need to analyze them, while g20 and g22 are not considered as they are difficult or because no feasible solution was found as of the literature [17]. The detailed results are shown in Table 1.

Large scale problems: Two problems are from [13]. They are numbered as C09 and C14 in that paper. C09 contains one equality constraint, while C14 contains three inequality separable constraints, and all the constraints are multi-modal and shifted. It is worth while to mention here that D is set to 150 variables, instead of 10 and 30 as in [13], the shifting matrix (SHI) is five times that in [13] and FFEs is set to one million.

In selecting and ranking the individuals, the following rules are chosen [18]: *i*) when compared two feasible solutions, the fittest one (according to the fitness function) is better; *ii*) a feasible solution is always better than an infeasible one; and *iii*) when compared two infeasible solutions, the one having the smaller sum of constraint violation is preferred. The equality constraints are transformed to inequalities in the following form, where ε is a small value ($\varepsilon = 0.0001$).

$$|h_e(\vec{x})| - \varepsilon \leq 0, \quad for\ e = 1, \dots, E \tag{5}$$

All the algorithms are run on a PC with a 3.0 GHz Core 2 Duo processor, 3.5GB RAM and Windows XP. All experiments are coded using Matlab 7.8.0 (R2009a)

4 Analysis and Discussions

In this section, all the results are discussed and analyzed using different measurements.

4.1 Small Scale Problems

Firstly, as shown in **Table 1**, DE is able to obtain the optimal solutions for all experiments. Based on the average results, DE (0%) is able to obtain the best average results for 16 test problems, DE (1%) 16, DE (5%) 17, DE (10%) 16, DE (15%) 17 and DE (20%) 16 test problems, respectively.

Considering the statistical test, we have chosen a non- parametric test, known as the Wilcoxon Signed Rank Test [19]. Based on the test results, there is no significant difference between all variants. Therefore, it is worthy to use other comparison criteria, such as computational time, the average fitness evaluations, and the convergence pattern, as will be seen below.

Thus, the average computational time consumed by each DE to obtain the optimal solutions with an error of 0.0001, i.e., a stopping criterion of $[f(\vec{x}) - f(\overrightarrow{x^*}) \leq 0.0001]$, where $f(\overrightarrow{x^*})$ is the best-known solution, is calculated, and a comparison summary is presented in Table 2. Based on the results, DE (5%) is the best.

Similarly, the average number of FFEs consumed by each DE variant to obtain the optimal solutions with an error of 0.0001 are calculated and presented in Table 3. The results show that DE (5%) is the best.

An example of the convergence plot is also presented in Fig. 1. This figure shows that DE (5%) performs best.

The diversity in a population can play an important role in an algorithm's success. So we have calculated the diversity for each DE variant. To do this, the diversity measure is calculated as follows:

1. At generation *t*, the average diversity (DV_{run}) of a point (x_z) to the centre (x_c, i.e. $x_{c,z} = (\sum_{z=1}^{ps} x_z)/ps$) of all points in the current population is calculated, as follows:

$$DV_{run}^{t} = (\textstyle\sum_{z=1}^{PS} \|\vec{x}_z^t - \vec{x}_c^t\|)\ /PS \quad \forall\, run = 1,2,\dots,25 \tag{6}$$

2. The total diversity (DV) is then calculated, as follows:

$$DV = \textstyle\sum_{t=1}^{max_t}((\sum_{run=1}^{25} DV_{run}^{t})/25)) \tag{7}$$

where $max_t = FFEs/PS$ is the maximum number of generations.

Table 1. Fitness function values obtained by DE for all experiments for small scale problems

Prob.		**DE (0%)**	**DE (1%)**	**DE (5%)**	**DE (10%)**	**DE (15%)**	**DE (20%)**
g01	**Best**	**-15.0000**	**-15.0000**	**-15.0000**	**-15.0000**	**-15.0000**	**-15.0000**
	Avg.	**-15.0000**	**-15.0000**	**-15.0000**	**-15.0000**	**-15.0000**	**-15.0000**
	STD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
g03	**Best**	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**
	Avg.	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**
	STD	**6.80E-16**	**6.80E-16**	**6.80E-16**	**6.80E-16**	**6.80E-16**	**6.80E-16**
g04	**Best**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
	Avg.	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
	STD	**7.43E-12**	**7.43E-12**	**7.43E-12**	**7.43E-12**	**7.43E-12**	**7.43E-12**
g05	**Best**	**5126.4967**	**5126.4967**	**5126.4967**	**5126.4967**	**5126.4967**	**5126.4967**
	Avg.	**5126.4967**	**5126.4967**	**5126.4967**	**5126.4967**	**5126.4967**	**5126.4967**
	STD	**1.86E-12**	**1.86E-12**	**1.86E-12**	**1.86E-12**	**1.86E-12**	**1.86E-12**
g06	**Best**	**-6961.8139**	**-6961.8139**	**-6961.8139**	**-6961.8139**	**-6961.8139**	**-6961.8139**
	Avg.	**-6961.8139**	**-6961.8139**	**-6961.8139**	**-6961.8139**	**-6961.8139**	**-6961.8139**
	STD	**9.28E-13**	**9.28E-13**	**9.28E-13**	**9.28E-13**	**9.28E-13**	**9.28E-13**
g07	**Best**	**24.3062**	**24.3062**	**24.3062**	**24.3062**	**24.3062**	**24.3062**
	Avg.	**24.3062**	**24.3062**	**24.3062**	**24.3062**	**24.3062**	**24.3062**
	STD	**7.25E-15**	**7.25E-15**	**7.25E-15**	**7.25E-15**	**7.25E-15**	**7.25E-15**
g08	**Best**	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**
	Avg.	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**
	STD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
g09	**Best**	**680.6301**	**680.6301**	**680.6301**	**680.6301**	**680.6301**	**680.6301**
	Avg.	**680.6301**	**680.6301**	**680.6301**	**680.6301**	**680.6301**	**680.6301**
	STD	**4.64E-13**	**4.64E-13**	**4.64E-13**	**4.64E-13**	**4.64E-13**	**4.64E-13**
g10	**Best**	**7049.2480**	**7049.2480**	**7049.2480**	**7049.2480**	**7049.2480**	**7049.2480**
	Avg.	**7049.2480**	**7049.2480**	**7049.2480**	**7049.2480**	**7049.2480**	**7049.2480**
	STD	**1.86E-12**	**1.86E-12**	**1.86E-12**	**1.86E-12**	**1.86E-12**	**1.86E-12**
g11	**Best**	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**
	Avg.	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**
	STD	**1.13E-16**	**1.13E-16**	**1.13E-16**	**1.13E-16**	**1.13E-16**	**1.13E-16**
g13	**Best**	**0.0539**	**0.0539**	**0.0539**	**0.0539**	**0.0539**	**0.0539**
	Avg.	**0.0539**	**0.0539**	**0.0539**	**0.0539**	**0.0539**	**0.0539**
	STD	**7.08E-18**	**7.08E-18**	**7.08E-18**	**7.08E-18**	**7.08E-18**	**7.08E-18**
g14	**Best**	**-47.7649**	**-47.7649**	**-47.7649**	**-47.7649**	**-47.7649**	**-47.7649**
	Avg.	**-47.7649**	**-47.7649**	**-47.7649**	**-47.7649**	**-47.7649**	**-47.7649**
	STD	**1.01E-13**	**1.01E-13**	**1.01E-13**	**1.01E-13**	**1.01E-13**	**1.01E-13**
g15	**Best**	**961.7150**	**961.7150**	**961.7150**	**961.7150**	**961.7150**	**961.7150**
	Avg.	**961.7150**	**961.7150**	**961.7150**	**961.7150**	**961.7150**	**961.7150**
	STD	**4.64E-13**	**4.64E-13**	**4.64E-13**	**4.64E-13**	**4.64E-13**	**4.64E-13**
g16	**Best**	**-1.9052**	**-1.9052**	**-1.9052**	**-1.9052**	**-1.9052**	**-1.9052**
	Avg.	**-1.9052**	**-1.9052**	**-1.9052**	**-1.9052**	**-1.9052**	**-1.9052**
	STD	**4.53E-16**	**4.53E-16**	**4.53E-16**	**4.53E-16**	**4.53E-16**	**4.53E-16**
g17	**Best**	**8853.5397**	**8853.5397**	**8853.5397**	**8853.5397**	**8853.5397**	**8853.5397**
	Avg.	8910.5535	8904.3062	8904.5640	8910.6791	**8901.3180**	8909.9798
	STD	3.28E+01	3.56E+01	3.58E+01	3.28E+01	3.66E+01	**3.24E+01**
g18	**Best**	**-0.8660**	**-0.8660**	**-0.8660**	**-0.8660**	**-0.8660**	**-0.8660**
	Avg.	**-0.8660**	**-0.8660**	**-0.8660**	**-0.8660**	**-0.8660**	**-0.8660**
	STD	**2.27E-16**	**2.27E-16**	**2.27E-16**	**2.27E-16**	**2.27E-16**	**2.27E-16**

Table 1 *(Continued)*

g21	**Best**	**193.7245**	**193.7245**	**193.7245**	**193.7245**	**193.7245**	**193.7245**
	Avg.	230.3984	235.6376	**225.1593**	240.8767	240.8767	240.8767
	STD	6.00E+01	6.24E+01	**5.71E+01**	6.42E+01	6.42E+01	6.42E+01
g23	**Best**	**-400.0551**	**-400.0551**	**-400.0551**	**-400.0551**	**-400.0551**	**-400.0551**
	Avg.	**-400.0551**	**-400.0551**	**-400.0551**	**-400.0551**	**-400.0551**	**-400.0551**
	STD	1.29E-09	4.91E-13	2.36E-13	**1.74E-13**	1.74E-13	8.55E-13

Table 2. Average computational time of DE for all experiments

Average time in seconds	**DE (0%)**	**DE (1%)**	**DE (5%)**	**DE (10%)**	**DE (15%)**	**DE (20%)**
	4.30E+00	4.23E+00	**4.16E+00**	4.24E+00	4.20E+00	4.26E+00

Table 3. Average FFEs of DE to convere to the optimal solutions for all experiments

Average FFEs	**DE (0%)**	**DE (1%)**	**DE (5%)**	**DE (10%)**	**DE (15%)**	**DE (20%)**
	7.9611E+04	7.8825E+04	**7.7704E+04**	7.8956E+04	7.8406E+04	7.9417E+04

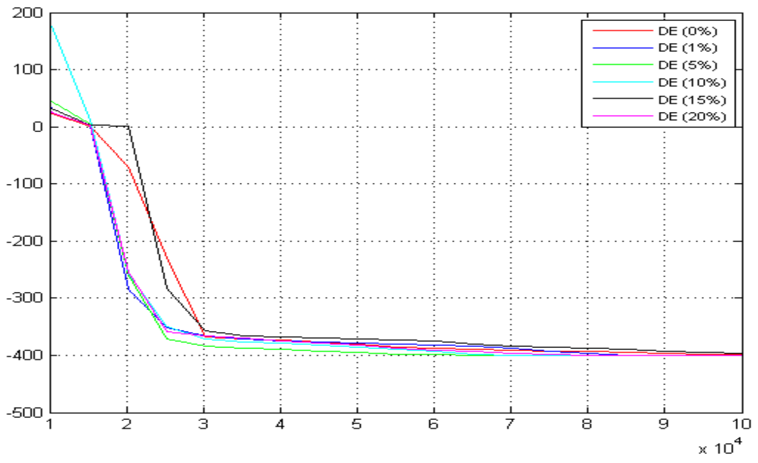


Fig. 1. A convergence plot of DE with all experiments for g23

Based on this measurement, a summary of comparison among all the variants is presented in Table 4. The results show that DE (5%) is the best. It is interesting to mention here that the average diversity increases until it reaches the peak of 7.8753 at DE (5%), and then deteriorates. However, if we analyze the average diversity for each problem, we find that DE (1%) is the best for the problem with separable inequality constraints (g01), DE (5%) is the best for those problems with polynomial inequality constraints (g04, g08 and g10), DE (10%) is the best for those problems with highly equality constraints (g03, g17, g21 and g23), DE (15%) is the best for those problems with polynomial objective function, DE (20%) is the best for those problems with separable equality constraints, and DE (1%) is the best for problems with exponential objective functions.

Table 4. Diversity results of DE for all experiments

	DE (0%)	**DE (1%)**	**DE (5%)**	**DE (10%)**	**DE (15%)**	**DE (20%)**
DV	15307.043	16274.956	**18847.536**	18670.962	17279.807	16480.546

An example of the change in average diversity of all DE variants during only one run is presented in Fig. 2.

4.2 Large Scale Problems

The detailed results for C09 and C14, with 150 decision variables are shown in Table 5. These two problems are analyzed based on the quality of solutions (the lower fitness function value means better performance) and the average diversity measure (the higher diversity value means better performance).

Based on the obtained results, DE (20%) is able to obtain the best solution for C09, while DE (15%) is the best for C14. Considering the average results, DE (5%), and DE (15%) perform best for C09 and C14, respectively. It is worth mentioning here that, in C09, the average results are improving from DE (0%) to DE (5%) then gradually deteriorate. However, in C14, the average results are improving from DE (0%) to DE (15%) then deteriorate. This means that the continuous increase of the number of feasible solutions in the initial population does not always guarantee better results.

Table 5. Fitness function values obtained for all experiments for large scale problems

Prob.		**DE (0%)**	**DE (1%)**	**DE (5%)**	**DE (10%)**	**DE (15%)**	**DE (20%)**
C09	**Best**	3.6266E+06	2.7529E+03	1.4912E+06	6.5547E+04	1.2466E+05	**4.3854E+02**
	Avg.	3.4826E+09	1.1244E+09	**5.5863E+08**	6.8502E+09	6.4068E+09	8.1105E+08
	STD	1.1364E+10	2.7745E+09	**7.4714E+08**	3.1984E+10	2.3996E+10	1.4867E+09
C14	**Best**	9.6977E+01	2.3629E+02	1.2516E+02	**1.7483E+01**	8.9648E+01	1.4760E+02
	Avg.	4.8658E+06	8.4327E+06	6.6195E+06	6.1549E+06	**1.4814E+06**	2.2320E+06
	STD	7.1572E+06	1.4013E+07	1.3849E+07	1.2792E+07	**4.3153E+06**	6.0215E+06

The average diversity of both problems has been calculated and is presented in Table 6. From this table, it is clear that DE (5%) is the best for C09, while DE (1%) is the best for C14.

Table 6. Average diversity results of DE for large scale problems

DV	**DE (0%)**	**DE (1%)**	**DE (5%)**	**DE (10%)**	**DE (15%)**	**DE (20%)**
C09	107.538	118.465	**133.081**	104.632	121.569	109.442
C14	0.188	**0.233**	0.128	0.159	0.152	0.182

Because no DE variant was able to obtain the optimal solutions, we have not provided the analysis of the average number of feasible solutions and average computational time.

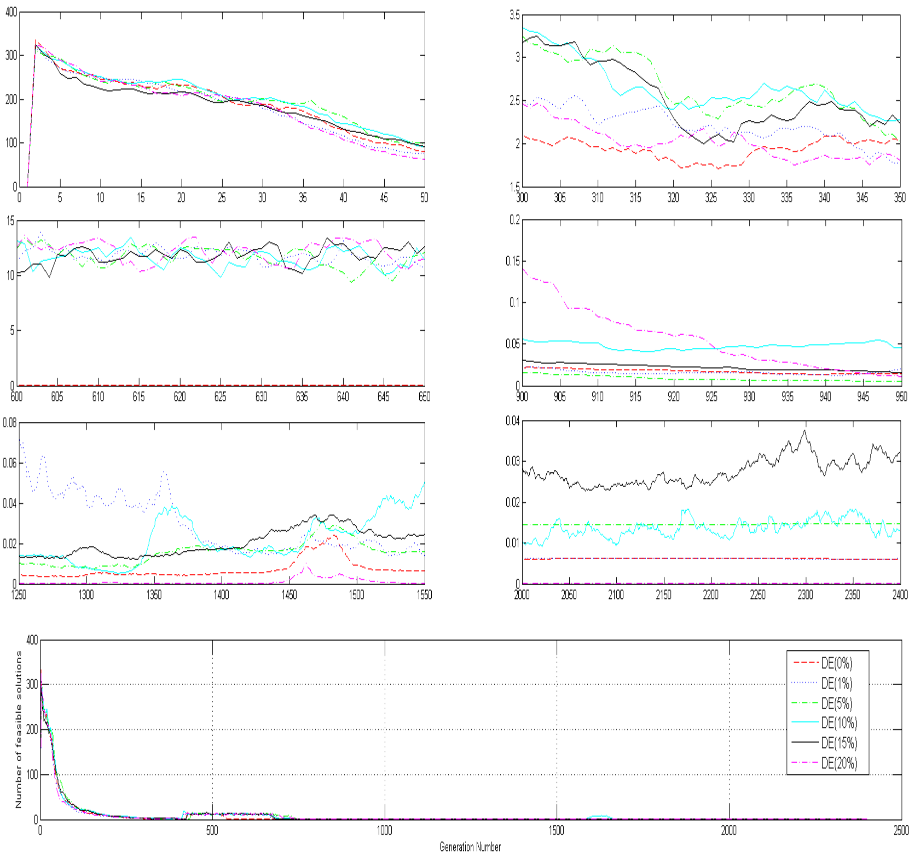


Fig. 2. Diversity values for one run

5 Conclusions

DE algorithms have shown superior performance to other EAs in solving COPs. However, starting with some initial feasible points could play a pivotal role in its performance and reduce the computational time required to obtain the optimal solutions. In this research, an analysis of the effect of the number of feasible points, within the initial population, on the performance of DE was provided. Using well-known bench-mark problems, the results were analyzed based on the quality of solutions, computational time, fitness evaluations and diversity measurements. The results showed that starting with 5% feasible points could lead to better performance, and hence researchers could reduce the amount of effort of using repairing techniques for more than this ratio of the initial infeasible solutions.

References

[1] Elsayed, S.M., Sarker, R.A., Essam, D.L.: Multi-operator based evolutionary algorithms for solving constrained optimization Problems. Computers and Operations Research 38, 1877–1896 (2011)

[2] Price, K.V., Storn, R.M., Lampinen, J.A.: Differential evolution: a practical approach to global optimization. Springer, Berlin (2005)

[3] Elsayed, S.M., Sarker, R.A., Essam, D.L.: GA with a new multi-parent crossover for constrained optimization. In: IEEE Congress on Evolutionary Computation, pp. 857–864 (2011)

[4] Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, MA (1989)

[5] Powell, M.: A fast algorithm for nonlinearly constrained optimization calculations. In: Watson, G. (ed.) Numerical Analysis, pp. 144–157. Springer, Heidelberg (1978)

[6] Singh, H.K., Ray, T., Smith, W.: Performance of infeasibility empowered memetic algorithm for CEC 2010 constrained optimization problems. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)

[7] Mezura-Montes, E., Velázquez-Reyes, J., Coello, C.A.C.: Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization. In: The 2005 Conference on Genetic and Evolutionary Computation, pp. 225–232. ACM, Washington DC (2005)

[8] Mezura-Montes, E., Coello, C.A.C.: A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Transactions on Evolutionary Computation 9, 1–17 (2005)

[9] Vieira, D.A.G., Adriano, R.L.S., Vasconcelos, J.A., Krahenbuhl, L.: Treating constraints as objectives in multiobjective optimization problems using niched Pareto genetic algorithm. IEEE Transactions on Magnetics 40, 1188–1191 (2004)

[10] Ray, T., Singh, H.K., Isaacs, A., Smith, W.: Infeasibility Driven Evolutionary Algorithm for Constrained Optimization. In: Mezura-Montes, E. (ed.) Constraint-Handling in Evolutionary Optimization. SCI, vol. 198, pp. 145–165. Springer, Heidelberg (2009)

[11] Barkat-Ullah, A.S.S.M., Sarker, R., Cornforth, D.: Search space reduction technique for constrained optimization with tiny feasible space. In: The 10th Annual Conference on Genetic and Evolutionary Computation, Atlanta, GA, USA, pp. 881–888 (2008)

[12] Barkat-Ullah, A.S.S.M., Elfeky, E.Z., Cornforth, D., Essam, D.L., Sarker, R.: Improved evolutionary algorithms for solving constrained optimization problems with tiny feasible space. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 1426–1433 (2008)

[13] Mallipeddi, R., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC 2010 competition and special session on single objective constrained real-parameter optimization. Technical Report, Nangyang Technological University, Singapore (2010)

[14] Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: 6th Int. Mendel Conference on Soft Computing, Brno, Czech Republic, pp. 76–83 (2000)

[15] Vesterstrom, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: IEEE Congress on Evolutionary Computation, pp. 980–1987 (2004)

[16] Elsayed, S.M., Sarker, R.A., Ray, T.: Parameters Adaptation in Differential Evolution. In: IEEE Congress on Evolutionary Computation (accepted, 2012)

[17] Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello, C.A.C., Deb, K.: Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical Report, Nanyang Technological University, Singapore (2005)
[18] Deb, K.: An Efficient Constraint Handling Method for Genetic Algorithms. Computer Methods in Applied Mechanics and Engineering 186, 311–338 (2000)
[19] Corder, G.W., Foreman, D.I.: Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach. John Wiley, Hoboken (2009)

Concurrent Differential Evolution Based on Generational Model for Multi-core CPUs

Kiyoharu Tagawa

School of Science and Engineering,
Kinki University, Higashi-Osaka 577-8502, Japan
tagawa@info.kindai.ac.jp

Abstract. In order to utilize multi-core CPUs more effectively, a new Concurrent Differential Evolution (CDE) is proposed. Then the proposed CDE (CDE/G) is compared with a conventional CDE (CDE/S). CDE/S uses only one population because it is based on the steady-state model. Therefore, CDE/S requires a time-consuming mutual exclusion or "lock" for every read-write access to the population. On the other hand, CDE/G is based on the generational model. By using a secondary population in addition to a primary one, CDE/G does not require any lock on the population and therefore is faster. Through the numerical experiment and the statistical test, it is demonstrated that CDE/G is superior to CDE/S in not only the run-time but also the quality of solutions.

Keywords: concurrent program, parallel processing, multi-core CPU, evolutionary algorithm, differential evolution, statistical test.

1 Introduction

Differential Evolution (DE) [1,2] can be regarded as a kind of Evolutionary Algorithm (EA). Because EAs maintain a lot of tentative solutions of the optimization problem manipulated competitively in the population, EAs have a parallel and distributed nature intrinsically. Therefore, many parallelization techniques have been contrived for various EAs [3,4]. These parallelization techniques of EAs can be introduced easily into DE. Actually, the parallel implementations of DE variants using networked computers and clustered computers have been reported [5,6,7]. Besides, Graphics Processing Units (GPUs) designed to accelerate graphics applications with several hundreds of simplified cores have been also used to run a parallelized DE program consisting of hundreds of threads [8].

Recently, multi-core CPUs, which have more than one processor (core), have been introduced widely into personal computers. In order to utilize the additional cores to execute costly application programs, concurrent implementations of them have been paid attention to [9]. Even though the number of available cores is not so large, the concurrent program executed on a multi-core CPU may be the most simple and easy way to realize a parallelized DE. Consequently, a concurrent program of DE, which is called Concurrent Differential Evolution

(CDE), has been proposed [10]. More specifically, the conventional CDE is a parallelized version of a neoteric DE based on the steady-state model.

The procedure of EAs for updating the individuals in the population is called "generation alternation model". Many EAs usually employ either of two types of generation alternation models [11]. The first one is called "generational model", while the second one is called "steady-state model". The original DE [1] is based on the generational model and holds two population: primary one and secondary one. After generating all individuals of the secondary population from those of the primary population, the primary one is replaced by the secondary one at a time. Alternative DE based on the steady-state model, which is sometimes called Sequential DE (SDE), has been also reported [12,13]. SDE holds only one population. Thereby, each individual in the population is updated one by one. Comparing with the generational model, the steady-state model seems to be suitable for parallelizing the procedures of EAs [14]. That is because the steady-state model does not require to synchronize the manipulations of all individuals in the population for replacing them by newborn individuals at a time.

The conventional CDE based on the steady-state model is called CDE/S in this paper. In our previous paper [10], it was shown that the run-time of CDE/S was reduced as the number of threads increased. However, CDE/S requires a time-consuming mutual execution or "lock" for every read-write access to the population. Besides, the quality of solutions obtained by CDE/S tends to fluctuate with the number of threads and the kind of optimization problem [15].

In this paper, a new CDE based on the generational model is proposed. The new CDE is called CDE/G. CDE/G is a lock free implementation of CDE. CDE/G accesses to the primary and secondary population without any lock and therefore faster. Instead of the mutual exclusion between threads, CDE/G uses a synchronized mechanism for multiple threads. Both CDE/G and CDE/S are coded by "Java", which is a popular language supporting the multi-threading program [16]. Through the numerical experiment and the statistical test conducted on a commodity multi-core CPU, it is demonstrated that CDE/G is superior to CDE/S in not only the run-time but also the quality of solutions.

The remainder of this paper is organized as follows. Section 2 gives a brief explanation of DE and SDE. Section 3 describes the procedures of CDE/S and CDE/G. Through the numerical experiment and the statistical test, CDE/G is compared with CDE/S in Section 4. Finally, Section 5 concludes the paper.

2 Differential Evolution

2.1 Representation

The real-parameter optimization problem is formulated as

$$\left[\begin{array}{ll}\text{minimize} & f(\boldsymbol{x}) = f(x_1, \cdots, x_j, \cdots, x_D), \\ \text{subject to} & \underline{x}_j \le x_j \le \overline{x}_j,\ j = 1, \cdots, D.\end{array}\right. \tag{1}$$

The solution of the optimization problem is a D-dimensional real-parameter vector $\boldsymbol{x} \in \Re^D$ that minimizes the objective function value $f(\boldsymbol{x}) \in \Re$. DE holds

N_p tentative solutions of the optimization problem, which are called individuals, in the population $\boldsymbol{P}$. The i-th individual $\boldsymbol{x}_i \in \boldsymbol{P}$ is represented as follows:

$$\boldsymbol{x}_i = (x_{1,i}, \cdots, x_{j,i}, \cdots, x_{D,i}) \tag{2}$$

where, $\underline{x}_j \leq x_{j,i} \leq \overline{x}_j,\ j = 1, \cdots, D;\ i = 1, \cdots, N_p$.

2.2 Strategy of DE

In order to generate a new individual of the population $\boldsymbol{P}$, DE uses a unique reproduction procedure called the strategy. Even though various strategies have been proposed for DE [1,2,12], a basic one named "DE/rand/1/exp" [2] is described and used in this paper. First of all, a parental individual $\boldsymbol{x}_i \in \boldsymbol{P}$ called "the target vector" is chosen from $\boldsymbol{P}$ in turn. Besides, three different individuals, say $\boldsymbol{x}_{i1}$, $\boldsymbol{x}_{i2}$ and $\boldsymbol{x}_{i3} \in \boldsymbol{P}$ $(i \neq i1 \neq i2 \neq i3)$, are selected randomly from $\boldsymbol{P}$. Furthermore, two integers r_1 and r_2 are selected stochastically from $[1,\ D]$. From the four parents, namely $\boldsymbol{x}_i$, $\boldsymbol{x}_{i1}$, $\boldsymbol{x}_{i2}$ and $\boldsymbol{x}_{i3}$, a new candidate for the individual $\boldsymbol{u} = (u_1, \cdots, u_j, \cdots, u_D)$ called "the trial vector" is generated as

$$u_j = \begin{cases} x_{j,i1} + F\,(x_{j,i2} - x_{j,i3}) & \text{for } j = r_1\%D + 1,\ (r_1+1)\%D + 1, \\ & \quad (r_1+2)\%D + 1,\ \cdots,\ (r_1+r_2)\%D + 1; \\ x_{j,i} \quad \text{for all other } j \in [1,\ D]. \end{cases} \tag{3}$$

where, the scale factor $F \in (0,\ 1+]$ is a user-defined control parameter.

2.3 Generation Alternation Model

The original DE proposed by Storn and Price [1] is based on the generational model and has two population: the primary one $\boldsymbol{P}$ and the secondary one $\boldsymbol{Q}$. The trial vector $\boldsymbol{u}$ is generated by using the individuals in $\boldsymbol{P}$. Then the trial vector $\boldsymbol{u}$ is compared with its corresponding target vector $\boldsymbol{x}_i \in \boldsymbol{P}$. If $f(\boldsymbol{u}) \leq f(\boldsymbol{x}_i)$ holds then $\boldsymbol{u}$ is selected to $\boldsymbol{z}_i \in \boldsymbol{Q}$. Otherwise, $\boldsymbol{x}_i$ is selected to $\boldsymbol{z}_i \in \boldsymbol{Q}$. After generating all $\boldsymbol{z}_i \in \boldsymbol{Q}$ $(i = 1, \cdots, N_p)$, $\boldsymbol{P}$ is replaced by $\boldsymbol{Q}$ at a time.

The neoteric DE or SDE [12] is based on the steady-state model and has only one population $\boldsymbol{P}$. As well as DE, the trial vector $\boldsymbol{u}$ is generated by using the individuals in $\boldsymbol{P}$ and compared with its corresponding target vector $\boldsymbol{x}_i \in \boldsymbol{P}$. However, if $f(\boldsymbol{u}) \leq f(\boldsymbol{x}_i)$ holds then $\boldsymbol{x}_i \in \boldsymbol{P}$ is replaced by $\boldsymbol{u}$ immediately.

3 Concurrent Differential Evolution

3.1 Concurrent Program

A concurrent program consists of multiple processes or threads. Therefore, if the concurrent program is executed on a multi-core CPU that has M $(M \geq 2)$ cores, M threads run in parallel at the maximum. However, not the concurrent program but the scheduler of Operating System (OS) decides how and when one of the threads is assign to core. Each thread has its own private working memory and no thread can access other threads' working memories. Besides, there is a common memory shared between all threads. For accessing the common memory, we assume Concurrent Read and Exclusive Write (CREW) [9] in which multiple threads may read the same memory location at the same time and one thread may write to a given memory location at any time. Of course, more than one thread can't read and write the same memory location at the same time.

3.2 Main Thread of CDE

CDE/S and CDE/G consist of a main thread and N_t ($N_t \geq 1$) worker threads. The population $\boldsymbol{P}$ exists in the common memory, which is shared between all threads. Because CDE/G is based on the generational model, CDE/G uses the secondary one $\boldsymbol{Q}$ in addition to the primary one $\boldsymbol{P}$. However, the main threads of CDE/S and CDE/G can be written in the same way. Except the detail of the worker thread denoted by `Worker(n)`, the main thread of CDE is provided as

```
1: Randomly generate $N_p$ individuals $\boldsymbol{x}_i \in \boldsymbol{P}$ ($i = 1, \cdots, N_p$).
2: Evoke Worker(n) ($n = 1, \cdots, N_t$) in parallel.
3: Wait until every Worker(n) ($n = 1, \cdots, N_t$) is completed.
4: Output the best $\boldsymbol{x}_b \in \boldsymbol{P}$ with the minimum $f(\boldsymbol{x}_b)$ and terminate.
```

CDE is based on a programming model known as "MapReduce" [9]. The programming model consists of two phases, "Map-phase" and "Reduce-phase". In the main thread of CDE, Step 2 corresponds to "Map-phase", while Step 4 corresponds to "Reduce-phase". All `Worker(n)` threads are evoked all together and executed concurrently in Step 2. Each `Worker(n)` gets the parents from $\boldsymbol{P}$. Then `Worker(n)` generates the trial vector $\boldsymbol{u}$ from them and evaluates the value of objective function $f(\boldsymbol{u})$ by using its private working memory. The results of all `Worker(n)` threads' efforts are consolidated in Step 4. The proposed CDE/G differs from the conventional CDE/S in the procedure of `Worker(n)`.

3.3 Worker Thread of CDE/S

The population $\boldsymbol{P}$ is divided into N_t sub-populations $\boldsymbol{P}_n$ called "chunks" such as $\boldsymbol{P} = \boldsymbol{P}_1 \cup \cdots \cup \boldsymbol{P}_n \cup \cdots \cup \boldsymbol{P}_{N_t}$. More specifically, the individual $\boldsymbol{x}_i \in \boldsymbol{P}$ with $i \,\%\, N_t = n$ is assigned to $\boldsymbol{P}_{n+1}$. Then CDE/S allocates each chunk $\boldsymbol{P}_n$ to `Worker(n)` statically. `Worker(n)` can read any individuals $\boldsymbol{x}_i \in \boldsymbol{P}$, but it may overwrite only the individuals $\boldsymbol{x}_i \in \boldsymbol{P}_n$. In other words, updating $\boldsymbol{x}_i \in \boldsymbol{P}_n$ is permitted only to `Worker(n)`. The procedure of `Worker(n)` is described as

```
1: For each individual $\boldsymbol{x}_i \in \boldsymbol{P}_n$, evaluate the value of $f(\boldsymbol{x}_i)$.
2: Set the generation as $g = 0$.
3: For each $\boldsymbol{x}_i \in \boldsymbol{P}_n$, i.e., the target vector, do from Step 3.1 to Step 3.3.
   3. 1 Randomly select $\boldsymbol{x}_{i1}$, $\boldsymbol{x}_{i2}$ and $\boldsymbol{x}_{i3} \in \boldsymbol{P}$ ($i \neq i1 \neq i2 \neq i3$).
   3. 2 Generate the trial vector $\boldsymbol{u}$ by (3). Evaluate the value of $f(\boldsymbol{u})$.
```

```
  3. 3  If $f(\boldsymbol{u}) \leq f(\boldsymbol{x}_i)$ holds then let $\boldsymbol{x}_i = \boldsymbol{u}$ and $f(\boldsymbol{x}_i) = f(\boldsymbol{u})$.
4:  If $g < G_m$ then set $g = g + 1$ and return to Step 3, otherwise complete.
```

In the procedure of CDE/S, most of the threads are just reading the individuals in the population $\boldsymbol{P}$. Overwriting the target vector $\boldsymbol{x}_i \in \boldsymbol{P}_n$ seldom occurs. It is not necessary to exclusively lock access to $\boldsymbol{x}_i \in \boldsymbol{P}_n$ while reading because multiple read operations can be done in parallel unless there is a write operation. Therefore, "read-write lock" [16] is employed for realizing the above CREW. In the implementation of CDE/S, N_t pairs of read-write locks are used to access $\boldsymbol{x}_i \in \boldsymbol{P}_n$ $(n = 1, \cdots, N_t)$. Because a unique `Worker(n)` may read and write the value of $f(\boldsymbol{x}_i)$ $(\boldsymbol{x}_i \in \boldsymbol{P}_n \subseteq \boldsymbol{P})$, any lock is not necessary to access $f(\boldsymbol{x}_i)$.

3.4 Worker Thread of CDE/G

As well as the primary population $\boldsymbol{P}$, the secondary one $\boldsymbol{Q}$ is also divided into N_t chunks $\boldsymbol{Q}_n$ $(n = 1, \cdots, N_t)$. CDE/G allocates each pair of chunks, $\boldsymbol{P}_n$ and $\boldsymbol{Q}_n$, to `Worker(n)` statically. `Worker(n)` can read any individuals $\boldsymbol{x}_i \in \boldsymbol{P}$, but it may overwrite only $\boldsymbol{x}_i \in \boldsymbol{P}_n$ and $\boldsymbol{z}_i \in \boldsymbol{Q}_n$. `Worker(n)` generates $\boldsymbol{Q}_n$ from $\boldsymbol{P}$ and replaces $\boldsymbol{P}_n$ by $\boldsymbol{Q}_n$. The procedure of `Worker(n)` is described as

```
1:  For each individual $\boldsymbol{x}_i \in \boldsymbol{P}_n$, evaluate the value of $f(\boldsymbol{x}_i)$.
2:  Set the generation as $g = 0$.
3:  For each $\boldsymbol{x}_i \in \boldsymbol{P}_n$, i.e., the target vector, do from Step 3.1 to Step 3.3.
  3. 1  Randomly select $\boldsymbol{x}_{i1}$, $\boldsymbol{x}_{i2}$ and $\boldsymbol{x}_{i3} \in \boldsymbol{P}$ $(i \neq i1 \neq i2 \neq i3)$.
  3. 2  Generate the trial vector $\boldsymbol{u}$ by (3). Evaluate the value of $f(\boldsymbol{u})$.
  3. 3  If $f(\boldsymbol{u}) \leq f(\boldsymbol{x}_i)$ holds then let $\boldsymbol{z}_i = \boldsymbol{u}$ $(\boldsymbol{z}_i \in \boldsymbol{Q}_n)$ and $f(\boldsymbol{x}_i) = f(\boldsymbol{u})$,
        otherwise let $\boldsymbol{z}_i = \boldsymbol{x}_i$ $(\boldsymbol{z}_i \in \boldsymbol{Q}_n$ and $\boldsymbol{x}_i \in \boldsymbol{P}_n)$.
4:  Wait until all the Worker(n) $(n = 1\ \cdots, N_t)$ threads arrive.
5:  Replace $\boldsymbol{P}_n$ by $\boldsymbol{Q}_n$ such as $\boldsymbol{x}_i = \boldsymbol{z}_i$ $(\boldsymbol{x}_i \in \boldsymbol{P}_n$ and $\boldsymbol{z}_i \in \boldsymbol{Q}_n)$.
6:  Wait until all the Worker(n) $(n = 1\ \cdots, N_t)$ threads arrive.
7:  If $g < G_m$ then set $g = g + 1$ and return to Step 3, otherwise complete.
```

CDE/G does not require the mutual exclusion or "lock". However, CDE/G requires to synchronize all the `Worker(n)` threads in each generation because we can't expect that the scheduler of OS executes all the threads in parallel. In the implementation of CDE/G, "cyclic-barrier" [16] is used to start and stop all the `Worker(n)` threads at Steps 4 and 6 in the above procedure of CDE/G.

4 Experiment

4.1 Setup of Experiment

DE, SDE, CDE/S and CDE/G were coded by the Java language and executed on a multi-core CPU (CPU: Intel® Core$^{\text{TM}}$i7@3.33[GHz]; OS: Microsoft Windows XP). The multi-core CPU has four cores each of which manipulates two threads at the same time. Therefore, eight threads run in parallel at the maximum.

The following six test functions were employed as the objective function $f(\boldsymbol{x})$ in (1). All benchmark problems have $D = 50$ dimensional real-parameters. The best objective function values are known as $f_p(\boldsymbol{x}) = 0$ $(p = 1, \cdots, 6)$.

- Sphere function (unimodal function):

$$f_1(\boldsymbol{x}) = \sum_{j=1}^{D} x_j^2$$

- Salomon function (multimodal function):

$$f_2(\boldsymbol{x}) = -\cos\left(2\,\pi\sqrt{\sum_{j=1}^{D} x_j^2}\right) + 0.1\sqrt{\sum_{j=1}^{D} x_j^2} + 1$$

- Rosenbrock function (multimodal function):

$$f_3(\boldsymbol{x}) = \sum_{j=1}^{D-1} (100\,(x_j^2 - x_{j+1})^2 + (1 - x_j)^2)$$

- Rastrigin function (multimodal function):

$$f_4(\boldsymbol{x}) = \sum_{j=1}^{D} (x_j^2 - 10\,\cos(2\,\pi\,x_j) + 10)$$

- Ackley function (multimodal function):

$$f_5(\boldsymbol{x}) = -20\,\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{j=1}^{D} x_j^2}\right) - \exp\left(\frac{1}{D}\sum_{j=1}^{D}\cos(2\,\pi\,x_j)\right) + 20 + e$$

- Griewank function (multimodal function):

$$f_6(\boldsymbol{x}) = \frac{1}{4000}\sum_{j=1}^{D} x_j^2 - \prod_{j=1}^{D}\cos\left(\frac{x_j}{\sqrt{j}}\right) + 1$$

The same control parameter values were used for DE, SDE, CDE/S and CDE/G. A desirable population size was recommended as $N_p = 5\,D = 250$ in the literature [1]. However, in order to assess the overhead time spent for the mutual exclusion between threads, we used smaller ($N_p = 100$) and bigger ($N_p = 500$) population sizes. Furthermore, the total number of the objective function evaluation was fixed to 2×10^5 in all cases. Thereby, all the methods were applied respectively to each of the six benchmark problems 100 times.

4.2 Result of Experiment

Figure 1 compares CDE/S (solid line) and CDE/G (broken line) in the run-time [sec] averaged over 100 independent runs, where the horizontal axis denotes the number of worker threads. The solid horizon and the broken horizon denote

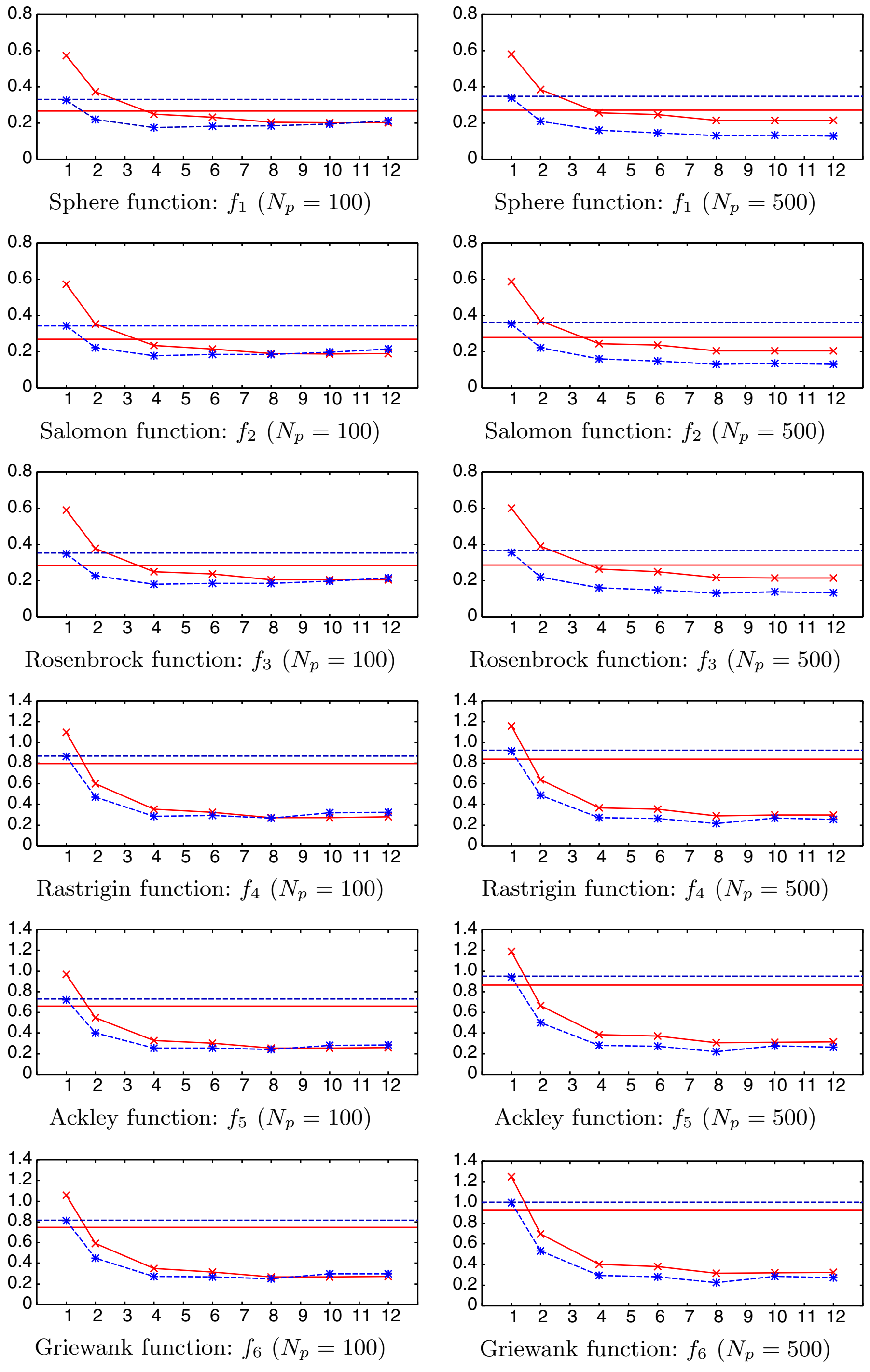


Fig. 1. Run-times of CDE/S (solid line) and CDE/G (broken line)

Table 1. Best objective function value found with $N_p = 100$

(a) Wilcoxon test between SDE and CDE/S

f_p	SDE	CDE/S with N_t workers						
		1	2	4	6	8	10	12
f_1	1.03E-12	1.03E-12	1.04E-12	9.79E-13	9.32E-12	1.29E-12	9.84E-13	2.80E-12
	(2.87E-13)	(2.87E-13)	(3.32E-13)	(3.45E-13)	(3.25E-12)	(4.07E-13)	(2.80E-13)	(9.46E-13)
	—	—	—	—	**	**	—	**
f_2	7.24E-1	7.24E-1	7.14E-1	7.13E-1	7.19E-1	7.14E-1	7.11E-1	7.06E-1
	(4.96E-2)	(4.96E-2)	(4.90E-2)	(4.31E-2)	(4.98E-2)	(4.77E-2)	(4.70E-2)	(4.92E-2)
	—	—	—	—	—	*	*	**
f_3	3.64E+1	3.64E+1	3.64E+1	3.64E+1	3.69E+1	3.63E+1	3.64E+1	3.65E+1
	(8.42E-1)	(8.42E-1)	(8.55E-1)	(8.84E-1)	(7.69E-1)	(9.59E-1)	(8.39E-1)	(8.56E-1)
	—	—	—	—	**	—	—	—
f_4	3.42E+1	3.42E+1	3.40E+1	3.41E+1	3.20E+1	3.28E+1	3.39E+1	3.24E+1
	(3.09)	(3.09)	(3.04)	(3.25)	(2.95)	(3.16)	(3.23)	(3.44)
	—	—	—	—	**	**	—	**
f_5	2.34E-7	2.34E-7	2.32E-7	2.22E-7	8.02E-7	2.68E-7	2.37E-7	3.94E-7
	(3.42E-8)	(3.42E-8)	(3.56E-8)	(3.32E-8)	(1.26E-7)	(4.12E-8)	(3.66E-8)	(6.66E-8)
	—	—	—	*	**	**	—	**
f_6	3.47E-12	3.47E-12	2.93E-12	1.77E-11	2.09E-11	3.33E-12	2.95E-12	9.32E-12
	(6.30E-12)	(6.30E-12)	(3.06E-12)	(1.38E-10)	(1.52E-11)	(2.40E-12)	(3.14E-12)	(9.61E-12)
	—	—	—	—	**	**	—	**

(b) Wilcoxon test between DE and CDE/G

f_p	DE	CDE/G with N_t workers						
		1	2	4	6	8	10	12
f_1	2.59E-12	2.59E-12	2.55E-12	2.60E-12	2.61E-12	2.57E-12	2.55E-12	2.60E-12
	(7.99E-13)	(7.99E-13)	(7.18E-13)	(6.81E-13)	(7.64E-13)	(7.31E-13)	(7.81E-13)	(7.58E-13)
	—	—	—	—	—	—	—	—
f_2	7.14E-1	7.14E-1	7.20E-1	7.19E-1	7.19E-1	7.15E-1	7.35E-1	7.24E-1
	(4.71E-2)	(4.71E-2)	(4.84E-2)	(5.08E-2)	(4.76E-2)	(5.11E-2)	(5.05E-2)	(4.65E-2)
	—	—	—	—	—	—	*	—
f_3	3.74E+1	3.74E+1	3.75E+1	3.74E+1	3.74E+1	3.75E+1	3.74E+1	3.74E+1
	(8.82E-1)	(8.82E-1)	(8.11E-1)	(8.40E-1)	(8.73E-1)	(8.24E-1)	(8.35E-1)	(8.01E-1)
	—	—	—	—	—	—	—	—
f_4	3.43E+1	3.43E+1	3.47E+1	3.48E+1	3.45E+1	3.44E+1	3.51E+1	3.49E+1
	(3.60)	(3.60)	(3.51)	(3.05)	(3.56)	(3.17)	(3.08)	(2.69)
	—	—	—	—	—	—	—	—
f_5	3.70E-7	3.70E-7	3.72E-7	3.68E-7	3.62E-7	3.87E-7	3.74E-7	3.76E-7
	(5.46E-8)	(5.46E-8)	(5.00E-8)	(5.43E-8)	(5.78E-8)	(6.33E-8)	(5.34E-8)	(5.07E-8)
	—	—	—	—	—	—	—	—
f_6	6.66E-12	6.66E-12	6.51E-12	8.51E-12	8.25E-12	7.13E-12	8.14E-12	6.84E-12
	(5.93E-12)	(5.93E-12)	(8.21E-12)	(1.32E-11)	(1.09E-11)	(1.06E-11)	(2.38E-11)	(6.19E-12)
	—	—	—	—	—	—	—	—

the run-times of SDE and DE respectively for reference. SDE is always faster than DE. The run-times of both CDE/S and CDE/G decrease as the number of worker threads increases. However, CDE/G is obviously faster than CDE/S in almost all cases. That is because CDE/G doesn't require the time-consuming mutual exclusion. From the run-time of CDE with $N_t = 1$, we can estimate the overhead time spent by a worker thread. The worker thread of CDE/S spends a large time for the read-write access to $\boldsymbol{x}_i \in \boldsymbol{P}$. By the way, we can observe the significant decrease of the run-times of CDE/S and CDE/G in the expensive functions, namely f_4, f_5 and f_6, defined by a lot of trigonometric functions.

Table 1 shows the average of the best objective function values found by DE, SDE, CDE/S and CDE/G with $N_p = 100$, where the standard deviation of them also appears in parentheses. Similarly, Table 2 shows the best objective function values found by the above methods with $N_p = 500$. From Table 1 and Table 2, the best objective function values found by CDE, i.e., the quality of

Table 2. Best objective function value found with $N_p = 500$

(a) Wilcoxon test between SDE and CDE/S

f_p	SDE	CDE/S with N_t workers						
		1	2	4	6	8	10	12
f_1	4.63E+1	4.63E+1	4.54E+1	4.31E+1	4.52E+1	4.24E+1	4.16E+1	4.10E+1
	(4.34)	(4.34)	(4.37)	(4.68)	(4.38)	(4.48)	(3.81)	(4.35)
	—	—	*	**	*	**	**	**
f_2	5.20	5.20	5.18	5.17	5.05	5.12	5.17	5.11
	(2.30E-1)	(2.30E-1)	(2.13E-1)	(2.33E-1)	(1.86E-1)	(2.13E-1)	(2.18E-1)	(1.96E-1)
	—	—	—	—	**	**	—	**
f_3	2.95E+3	2.95E+3	3.00E+3	2.84E+3	2.71E+3	2.75E+3	2.71E+3	2.66E+3
	(4.30E+2)	(4.30E+2)	(4.01E+2)	(3.69E+2)	(3.89E+2)	(4.31E+2)	(4.43E+2)	(4.15E+2)
	—	—	—	—	**	**	**	**
f_4	1.96E+2	1.96E+2	1.98E+2	1.96E+2	1.94E+2	1.96E+2	1.98E+2	1.96E+2
	(1.08E+1)	(1.08E+1)	(9.78)	(1.05E+1)	(1.00E+1)	(1.13E+1)	(9.84)	(9.39)
	—	—	—	—	—	—	—	—
f_5	3.07	3.07	3.08	3.07	3.07	3.04	2.99	2.99
	(8.22E-2)	(8.22E-2)	(8.36E-2)	(9.04E-2)	(9.99E-2)	(1.05E-1)	(9.57E-2)	(8.77E-2)
	—	—	—	—	—	—	**	**
f_6	1.41	1.41	1.41	1.40	1.44	1.39	1.37	1.37
	(3.91E-2)	(3.91E-2)	(4.32E-2)	(3.88E-2)	(4.35E-2)	(3.82E-2)	(4.03E-2)	(3.84E-2)
	—	—	—	**	**	**	**	**

(b) Wilcoxon test between DE and CDE/G

f_p	DE	CDE/G with N_t workers						
		1	2	4	6	8	10	12
f_1	5.39E+1	5.39E+1	5.47E+1	5.38E+1	5.56E+1	5.40E+1	5.36E+1	5.45E+1
	(5.77)	(5.77)	(5.20)	(5.99)	(5.20)	(5.08)	(5.30)	(4.68)
	—	—	—	—	*	—	—	—
f_2	5.20	5.20	5.24	5.25	5.26	5.25	5.23	5.23
	(2.28E-1)	(2.28E-1)	(2.66E-1)	(2.07E-1)	(2.37E-1)	(1.93E-1)	(2.13E-1)	(2.11E-1)
	—	—	—	—	—	—	—	—
f_3	3.41E+3	3.41E+3	3.43E+3	3.49E+3	3.50E+3	3.52E+3	3.53E+3	3.56E+3
	(4.61E+2)	(4.61E+2)	(4.83E+2)	(4.37E+2)	(4.85E+2)	(5.14E+2)	(5.17E+2)	(4.51E+2)
	—	—	—	—	—	—	—	*
f_4	1.97E+2	1.97E+2	1.98E+2	1.97E+2	1.97E+2	1.98E+2	1.96E+2	1.97E+2
	(1.01E+1)	(1.01E+1)	(9.22)	(1.01E+1)	(9.15)	(9.54)	(9.13)	(9.15)
	—	—	—	—	—	—	—	—
f_5	3.19	3.19	3.19	3.20	3.20	3.18	3.20	3.19
	(9.35E-2)	(9.35E-2)	(9.38E-2)	(7.70E-2)	(9.79E-2)	(9.80E-2)	(8.88E-2)	(8.08E-2)
	—	—	—	—	—	—	—	—
f_6	1.48	1.48	1.49	1.48	1.50	1.48	1.48	1.49
	(5.19E-2)	(5.19E-2)	(4.68E-2)	(5.39E-2)	(4.68E-2)	(4.57E-2)	(4.77E-2)	(4.21E-2)
	—	—	—	—	*	—	—	—

solutions, seem to change slightly with the number of worker threads. Therefore, the quality of solutions was analyzed statistically by using Wilcoxon test [17]. The null hypothesis was that there was no significant difference between two objective function values found by SDE (DE) and CDE/S (CDE/G).

The results of Wilcoxon test are also summarized in Table 1 and Table 2. The symbols "$*$" and "$**$" mean that the null hypothesis may be rejected with the risk less than 0.05 and 0.01 respectively. The quality of solutions obtained by CDE/S is likely to change with the number of threads and the kind of benchmark problem. The quality of solutions gets still worse in case of $N_p = 500$.

5 Conclusion

In this paper, a lock free implementation of CDE, i.e., CDE/G, was proposed and compared with the conventional CDE/S that needed the mutual exclusion between threads. Through the numerical experiment and the statistical test conducted on a multi-core CPU, it could be confirmed that CDE/G was superior to CDE/S with respect to the run-time and the quality of solutions.

In our future work, we would like to apply CDE/G to practical applications because CDE/G is more effective in the optimization of expensive functions.

References

1. Storn, R.M., Price, K.V.: Differential evolution - a simple and efficient heuristic for global optimization over continuous space. Journal of Global Optimization 11(4), 341–359 (1997)
2. Das, S., Suganthan, P.N.: Differential evolution - a survey of the state-of-the art. IEEE Trans. on Evolutionary Computation 15(1), 4–31 (2011)
3. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers (2001)
4. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Trans. on Evolutionary Computation 5(6), 443–462 (2002)
5. Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Parallel differential evolution. In: Proc. of IEEE Congress on Evolutionary Computation, pp. 2023–2029 (2004)
6. Zhou, C.: Fast parallelization of differential evolution algorithm using MapReduce. In: Proc. of Genetic and Evolutionary Computation Conference, pp. 1113–1114 (2010)
7. Dorronsoro, B., Bouvry, P.: Improving classical and decentralized differential evolution with new mutation operator and population topologies. IEEE Trans. on Evolutionary Computation 1(15), 67–98 (2011)
8. de Veronses, L., Krohling, R.: Differential evolution algorithm on the GPU with C-CUDA. In: Proc. of IEEE Congress on Evolutionary Computation, pp. 1–7 (2010)
9. Breshears, C.: The Art of Concurrency - A Thread Monkey's Guide to Writing Parallel Applications. O'Reilly (2009)
10. Tagawa, K., Ishimizu, T.: Concurrent differential evolution based on MapReduce. International Journal of Computers 4(4), 161–168 (2010)
11. Syswerda, G.: A study of reproduction in generational and steady-state genetic algorithms. In: Foundations of Genetic Algorithms 2, pp. 94–101. Morgan Kaufmann Publisher (1991)
12. Feoktistov, V.: Differential Evolution in Search Solutions. Springer (2006)
13. Tagawa, K., Ishimizu, T.: A comparative study of distance dependent survival selection for sequential DE. In: Proc. of IEEE International Conference on System, Man, and Cybernetics, pp. 3493–3500 (2010)
14. Davison, B.D., Rasheed, K.: Effect of global parallelism on a steady state GA. In: Proc. of Genetic and Evolutionary Computation Conference Workshops, Evolutionary Computation and Parallel Processing Workshop, pp. 167–170 (1999)
15. Tagawa, K.: A statistical study of concurrent differential evolution on multi-core CPUs. In: Proc. of Italian Workshop on Artificial Life and Evolutionary Computation, pp. 1–12 (2012)
16. Göetz, B., Peierls, T., Bloch, J., Bowbeer, J., Holmes, D., Lea, D.: Java Concurrency in Practice. Addison-Wesley (2006)
17. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures, 5th edn. CRC Press (2011)

Figure of Merit Based Fitness Functions in Genetic Programming for Edge Detection

Wenlong Fu[1], Mark Johnston[1], and Mengjie Zhang[2]

[1] School of Mathematics, Statistics and Operations Research
[2] School of Engineering and Computer Science,
Victoria University of Wellington, PO Box 600, Wellington, New Zealand

Abstract. The figure of merit (FOM) is popular for testing an edge detector's performance, but there are very few reports using FOM as an evaluation method in Genetic Programming (GP). In this study, FOM is investigated as a fitness function in GP for edge detection. Since FOM has some drawbacks from type II errors, new fitness functions are developed based on FOM in order to address these weaknesses. Experimental results show that FOM can be used to evolve GP edge detectors that perform better than the Sobel detector, and the new fitness functions clearly improve the ability of GP edge detectors to find edge points and give a single response on edges, compared with the fitness function using FOM.

Keywords: Genetic Programming, Edge Detection, Figure of Merit.

1 Introduction

Edges characterise object boundaries in a simple way and are therefore useful for boundary detection, image segmentation, image registration, and object detection [16]. Without using the specific knowledge of edges, methods based on Machine Learning often use features extracted from fixed neighbourhoods [14] or variant neighbourhoods [5] to classify each pixel. A numerical measure of the discrepancy of the outputs from an edge detector learned by a Machine Learning algorithm and the desired outputs of the training images is very important for determining the detector's performance, such as the detection accuracy (the ability to find edge points correctly) and the localisation (the distance from a predicted edge point to the true edge point). The F-measure has previously been used to evaluate the performance of learned edge detectors [14,5]. However, there are only a few reports using the figure of merit (FOM) [19] in the learning (training) stage.

Genetic Programming (GP) has been applied to edge detection since at least 1996 [9,18]. A fitness function which evaluates a detector in training, is usually based on the accuracy of predictions [21,5] when edge detection is considered as a binary classification problem. Generally, the computational load of a GP system is heavy, and the fitness function in the system should be as computationally inexpensive as possible. When a pixel predicted as an edge point is required to be very close to the true edge point, the computational cost for the matching operations will be obviously increased. The matching operation (one to one) aims at assigning predicted edge points to desired edge points as much as possible.

Although FOM has not been previously used as a fitness function in GP, it is worth investigating because FOM allows a tolerance of distance from predicted edge points to true edge points and its computational cost is much lower than the F-measure method, including matching the detected edge points to ground truth. FOM fails to measure type II errors (false negatives) when there is overlap in the matching between many predicted edge points and a few true edge points [1]. Whether this weakness makes GP perform poorly when evolving low-level GP edge detectors for natural images is not clear and needs to be investigated. If a tolerance distance for a predicted edge point to a true edge point is considered in the training stage, only information from the predicted pixel (true edge point or not) is not enough, and to match predicted edge points to true edge points, a true edge map of the whole individual image should be used for the matching operation. Therefore, the training data needs to include whole individual images. However, the traditional sampling techniques are based on local pixels and the target is only related to the predicted pixel without an allowed distance to the true edge point. When whole individual images are used as terminal in GP, the training data is based on individual images, so FOM can easily be used to evaluate detection performance.

1.1 Goals

The overall goals of this paper are to investigate FOM and its variants as fitness functions in GP for evolving low-level edge detectors using an entire image as input. Here, a low-level edge detector is based on raw pixels, and it is constructed by a GP system via automatically searching pixels. We will use an existing FOM variant as a fitness function in GP to check the detection performance. In order to address the drawbacks from FOM, we will develop a new FOM variant as a new fitness function. Specifically, we investigate the following objectives.

- Whether FOM as a fitness function in a GP system can be used to evolve good low-level edge detectors, compared with the Sobel detector [7].
- Whether fitness functions based on the FOM variants considering type II errors can improve the detection performance, compared with FOM.
- What properties exist in detected images from the evolved detectors using the fitness functions based on FOM.

1.2 Organisation

In the remainder of the paper, Section 2 briefly describes the background on the evaluation methods based on FOM in edge detection. Section 3 introduces an existing FOM variant as a fitness function and develops a new fitness function after modifying FOM. After the experimental design is presented in Section 4, Section 5 discusses the experimental results. Section 6 gives conclusions and future work directions.

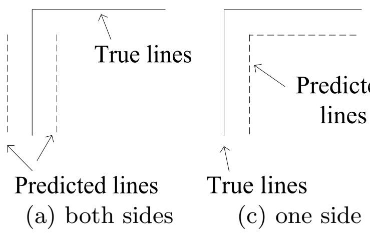


Fig. 1. The same FOM value with different detection results

2 Background

2.1 FOM

Edge detection is a subjective task and there is no standard description for *the* edges in an image. It is hard to evaluate the performance of edge detectors because there are no unique solutions for edge maps in most natural images. Canny [3] suggested three important general principles for evaluation of edge detectors: good detection, good localisation and single response. Good detection means that a detector should have low probability of failing to detect an edge or incorrectly marking a background pixel as an edge point. Good localisation means that a detected edge point should be as close as possible to the true edge point. Single response requires only marking one side at boundaries.

Evaluation based on statistics (e.g. detection accuracy) is useful for binary classification problems, but it is a poor measure of "reconstruction fidelity" in edge detection [1]. In addition, finding an optimal matching between predictions and ground truth has complexity $O(N_I log(N_I))$ [14], where N_I is the total number of pixels. Localisation is another performance criterion in edge detection. Pratt [19] proposed FOM to evaluate localisation. The standard FOM is defined by equation (1), where $d_1(i)$ is the distance of one predicted edge point i to the nearest true edge point, N_P is the number of the predicted edge points, N_T is the number of the true edge points, and α is a positive weight factor for distance $d_1(i)$. In this paper, FOM indicates the standard FOM. In general, α is small and is set to $\frac{1}{9}$ based on the overlap of a 3×3 window. The value of FOM ranges from 0 to 1; higher values indicate predictions with better quality detection.

$$FOM = \frac{1}{\max\{N_T, N_P\}} \sum_{i=1}^{N_P} \frac{1}{1 + \alpha d_1^2(i)} \tag{1}$$

Compared with FOM, F-measure needs to optimise the match between detected edge points and the true edge points when a tolerance distance from a predicted edge point to a true edge point is allowed [14]. FOM evaluates edge detectors in terms of both detection accuracy and localisation, but there are some drawbacks [1]. For instance, Figure 1 shows two different detected edge maps with the same FOM value. However the detection in Figure 1 (a) has overlap in one true line and misses the top line. It is found that FOM may not reflect some type II errors.

2.2 Related Work for Edge Detection Using GP

The existing work in edge detection using GP mainly focuses on low-level edge detection. Harris and Buxton [9] designed approximate response detectors in one-dimensional signals by GP, but it is based on the theoretical analysis of the ideal edge detector and the corresponding properties. Poli [18] suggested using similarly four macros for searching a pixel's neighbours in image processing using GP, and Ebner [4] used four shift functions and other functions to approximate the Canny detector. The Sobel detector is approximated by hardware design [10] with the relationship between a pixel and its neighbourhood as terminals. Bolis et al [2] simulate an artificial ant to search edges in an image. Zhang and Rockett [21] evolved a 13×13 window filter for comparison with the Canny edge detector. A 4×4 window is employed to evolve digital transfer functions (combination of bit operators or gates) for edge detection by GP [8]. Our previous work [6,5] used GP for edge detection based on ground truth without using windows. Also, Wang and Tan [20] used linear GP to find binary image edges, inspired by morphological operators, erosion and dilation, as terminals for binary images. For detecting boundaries, some image operators were employed to combine high-level detectors evolved by GP with a trained logistic regression classifier [13]. However, the evaluation in all these methods for a GP edge detector does not consider the offset for the detection results of training images.

3 Method

This section describes a method for using GP to evolve detectors and fitness functions based on FOM. Different from a training dataset based on individual pixels, the training dataset in this paper is based on individual images, and an entire image as input is used to evolve edge detectors.

3.1 Sets of Terminals and Functions

The GP system in [5] has successfully evolved edge detectors based on shifting functions rather than using a fixed window. It is suitable for evaluating detecting results with a limited offset. In this paper, the investigation is about using fitness functions based on FOM, so we adopt the terminal set and the function set from [5] to evolve low-level edge detectors. The terminal set contains the whole image x and random constants rnd in the range of $[-10, 10]$. The function set is $\{+, -, \times, \div, shift_{n,m}, abs, sqrt, square\}$, where function $shift_{n,m}$ shifts n columns and m rows; positive n (m) means to shift right (down), and negative means to shift left (up).

3.2 Fitness Functions Based on FOM

Since the potential drawbacks in FOM do not usually occur, FOM is employed as a fitness function (see (1)) to check whether it is suitable for evolving reasonable detectors for natural images.

In order to improve the sensitivity to type II errors, modification of FOM focuses on the matching direction. An existing FOM variant only uses distances from ground truth to predicted edges, and includes a factor about false (unmatched) edge points [17]. This FOM variant was only used to test artificial images in [17], but we directly use it as a new fitness function (F_{nn}) in the GP system to test natural images. F_{nn} is defined in (2), where N_{FM} is the number of false edge points, β is a factor for the response on the false edge points, and $d_2(i)$ for the nearest distance from a true edge point i to a predicted edge point. Note that the matching directions in d_1 and d_2 are different. In [17], it was suggested that β be set to 1. Here α is *also* used for d_2 because of the overlap. In F_{nn}, α and β are used to balance type I errors and type II errors, but the relationship between α and β is *not* clear. The second factor in F_{nn} is sensitive to β: when β is large, the detection results mainly focus on recall (the number of true positives).

$$F_{nn} = \left(\frac{1}{N_T} \sum_{i=1}^{N_T} \frac{1}{1 + \alpha d_2^2(i)} \right) \left(\frac{1}{1 + \frac{\beta N_{FM}}{N_T}} \right) \tag{2}$$

We propose a new FOM variant inspired by the Hausdorff distance [12], taking into account both distances between the ground truth and the predicted edges ($d_1(i)$ and $d_2(i)$). In order to fairly balance the measurement between type I errors and type II errors, F_b is designed based on the balance between $d_1(i)$ and $d_2(i)$. Function F_b is defined by (3), where, $N_{T \bigcup P}$ is the number of points that are either predicted as edge points or true edge points. Since recall is the number of true positives divided by the total number of true edge points and precision is the number of true positives divided by the total number of predicted edge points, recall can be considered as the measure from the ground truth to predicted edge maps, and precision for the reverse direction. Function F_b considers both directions and merges them together, rather than using a weight to balance the result from $d_1(i)$ and the result from $d_2(i)$. Compared with the suggestion on modifying FOM in [1], F_b only considers the edge points (the true edge points and predicted edge points) rather than all pixels of images, which means that the computational cost for F_b is lower than the metric based on all pixels in an image because the number of (true and predicted) edge points in one image is usually much lower than the total number of pixels in the image.

$$F_b = \frac{1}{N_{T \bigcup P}} \sum_{i=1}^{N_{T \bigcup P}} \frac{1}{(1 + \alpha d_1^2(i))(1 + \alpha d_2^2(i))} \tag{3}$$

4 Experimental Design

We use the training images from the Berkeley Segmentation Dataset (BSD) [14]. The BSD consists of natural images (of size 481×321 pixels) with ground truth provided. The images in the BSD come from different places and have different contents. In order to evaluate the detection results from these training images,

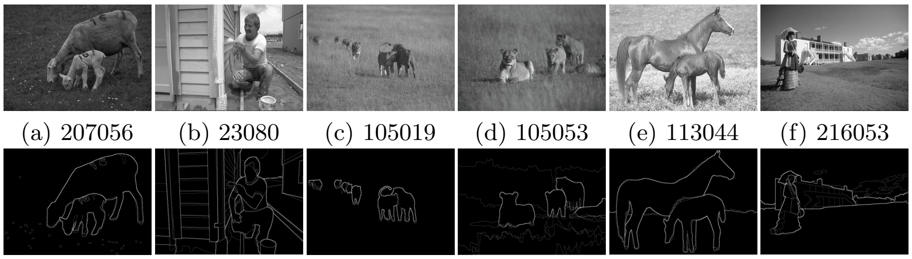


Fig. 2. Training images from BSD dataset and the ground truth

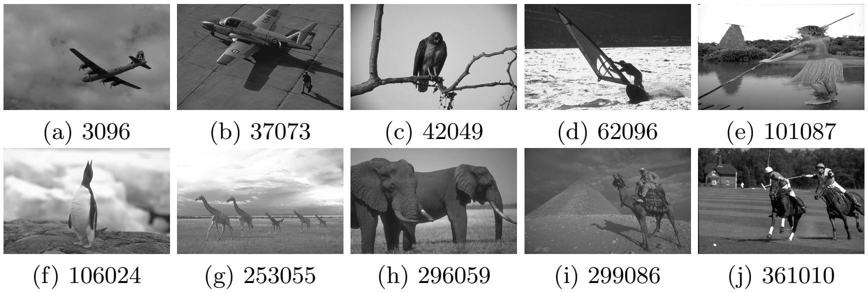


Fig. 3. Test images from BSD dataset

the ground truth are combined from five to seven observations as gray-level images. Since there are redundancies existing in the training data (containing 200 images) and the computational cost is high for GP, we only select six images with rich edge contents as the training dataset for checking the new fitness function performances. Fig. 2 shows the training images and the ground truth.

To reduce the computation time, we randomly sample five different subimages of size 41×41 from each image as training subimages, and use one subimage as input. Ten additional images (Fig. 3) from the BSD test set are used as the test dataset. There is no post-processing during the training and test stages, following [15].

The parameter values for the different fitness functions are: α (in FOM, F_{nn}, F_b) $\frac{1}{9}$; β (in F_{nn}) 1. The parameter values for the GP system are: population size 500; maximum generations 200; maximum depth (of a GP detector) 10; and probabilities for (single point) mutation 0.35, (subtree exchange) crossover 0.60 and elitism (replication) 0.05. The GP experiment is repeated for 30 independent runs. 30 different random seeds are used to generate the same initial population when different fitness functions are used.

5 Results and Discussion

5.1 Overall Results

Table 1 gives the means and standard deviations of the detection results (FOM) from the three fitness functions (FOM, F_b and F_{nn}) for the ten test images.

Table 1. Means ± Standard Deviations of FOM for the Ten Test Images from the Fitness Functions FOM, F_b and F_{nn}

Image	FOM	F_{nn}	F_b	Sobel
3096	0.588 ± 0.102	0.710 ± 0.069	0.733 ± 0.068	0.740
37073	0.693 ± 0.079	0.734 ± 0.051	0.761 ± 0.073	0.436
42049	0.678 ± 0.094	0.754 ± 0.052	0.797 ± 0.048	0.646
62096	0.369 ± 0.075	0.563 ± 0.127	0.453 ± 0.094	0.402
101087	0.648 ± 0.061	0.730 ± 0.035	0.765 ± 0.042	0.444
106024	0.416 ± 0.070	0.424 ± 0.048	0.471 ± 0.045	0.307
253055	0.488 ± 0.078	0.551 ± 0.048	0.589 ± 0.050	0.402
296059	0.517 ± 0.114	0.660 ± 0.054	0.636 ± 0.085	0.536
299086	0.535 ± 0.083	0.601 ± 0.049	0.649 ± 0.066	0.426
361010	0.454 ± 0.057	0.677 ± 0.049	0.631 ± 0.084	0.437

Table 2. Comparisons Among the Fitness Functions FOM, F_b and F_{nn}

Image	(FOM,F_{nn})	(FOM,F_b)	(F_{nn},F_b)	(FOM,Sobel)	(F_{nn},Sobel)	(F_b,Sobel)
3096	↓	↓	–	↓	–	–
37073	–	↓	–	↑	↑	↑
42049	↓	↓	↓	–	↑	↑
62096	↓	↓	↑	–	↑	–
101087	–	↓	↓	↑	↑	↑
106024	–	↓	↓	↑	↑	↑
253055	↓	↓	↓	↑	↑	↑
296059	↓	↓	–	–	↑	↑
299086	↓	↓	↓	↑	↑	↑
361010	↓	↓	↑	–	↑	↑

The reason for only using the standard FOM to evaluate test performance is to check whether the GP edge detectors' performance evolved by F_b and F_{nn} can be improved, compared with the fitness function directly using FOM. For comparison, the maximum FOM values from the Sobel detector based on 20 different threshold levels are given. Table 2 shows the comparisons between the detection results from these fitness functions using multiple one-sample or paired t-tests with overall significance level 0.05 and p-values adjusted by the Holm's method [11] over all comparisons. Here, "↑" indicates the first function is significantly better than the second function, "↓" being significantly worse and "–" being not significantly different.

Compared with the Sobel detector, the three fitness functions all have better detection results for most images; only for image 3096, the results from FOM are significantly worse than the result from the Sobel detector. The detected ten image results from F_{nn} are significantly better than the results from the Sobel detector, except for image 3096. The edge detectors from F_b have eight images with significantly better detection than the Sobel detector. Therefore, we conclude that the fitness functions based on FOM can be used to evolve good low-level edge detectors.

Compared with the standard FOM, its variants F_{nn} and F_b are significantly better. Firstly, F_{nn} has seven images with significantly better detection than FOM, and all results from F_b are significantly better than the results from FOM. From Table 1, the improvement for the detected results from F_b is around 0.10 for each image, compared with the results from FOM. Secondly, F_{nn} only contains images 62096 and 361010 with significantly better detection than F_b based on Table 2. However, half of the ten images are significantly better detected by the edge detectors using F_b than the edge detectors from F_{nn}. Therefore, F_b is significantly better than F_{nn} in most cases.

5.2 Detected Images

Fig. 4 shows the detected images by the best detectors from the three fitness functions and the Sobel detector. To investigate the detection features from these fitness functions, the best detection results from four typical images are presented in Fig. 4. First of all, the results from FOM are similar to the results from the Sobel detector, namely double responses for most edges. However, the results from F_b and F_{nn} only have a single response for most edges, e.g., the edge of the sail in image 62096. Secondly, the results from F_b have high recall, e.g., finding the top boundary of the sail in image 62096 and the double lines in the right middle boundary in image 296059. Lastly, the influence from noise and textures on the results from F_{nn} are heavier than the results from F_b, e.g., the background of image 106024.

In summary, compared with the Sobel detector, the detectors evolved by the fitness functions based on FOM (FOM, F_{nn} and F_b) have high recall but bring low precision for the predicted texture edges, and the detectors from the fitness functions F_b and F_{nn} have a single response for the boundaries.

5.3 Discussion

From the comparisons of the three fitness functions and the detected images, the fitness functions based on FOM are useful to evolve good detectors in the GP system. Although FOM has some disadvantages, the evolved edge detectors from the standard FOM can perform detection well on natural images (see Table 1 and Fig. 4). For natural images, the detection results from most detectors usually contain type I errors and type II errors (see all detected images in Fig. 4) so that the standard FOM does not fail to evaluate detection performance. However, FOM is not sensitive to type II errors, which leads the detectors from FOM to miss some true edge points.

For the FOM variants, F_b and F_{nn} introduce a measure of type II errors so that there is a balance for the abilities of detecting edge points and correctly predicting edge points. However, F_{nn} has the problem of type I errors when the recall is high; but this is not the case in F_b. This is why noise strongly affects the detected images. For the high recall and thin edge lines in F_b and F_{nn}, a potential reason is that the matching direction from the ground truth to predictions is considered in both fitness functions.

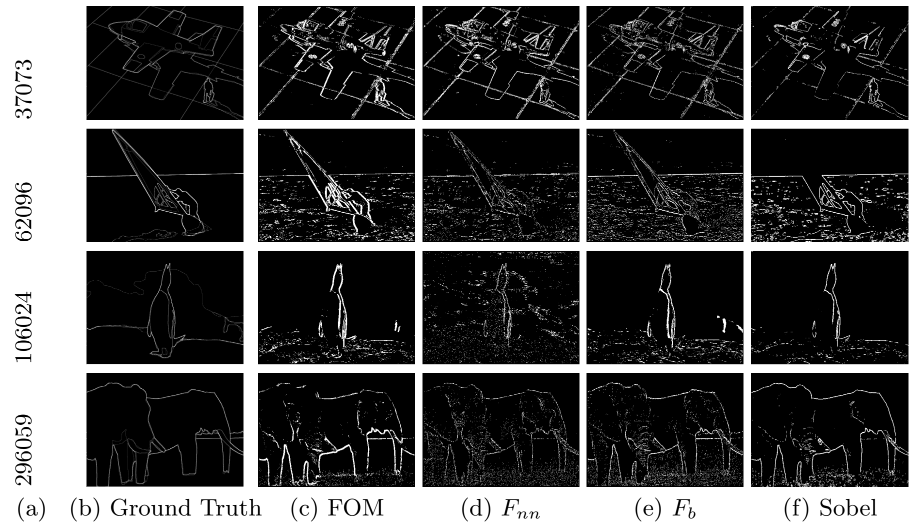


Fig. 4. Detected images by the best detectors from FOM, F_{nn}, F_b and the Sobel detector

6 Conclusions

The goals of this paper were to investigate FOM and its FOM variants as fitness functions for evolving GP edge detectors. Based on experimental results on natural images, FOM can be successfully used to evolve low-level edge detectors which are better than the Sobel detector. The detectors evolved by FOM variants (as fitness functions) can detect more true edge points, and most detected edges are thinner than the results from the standard FOM and the Sobel detector. The proposed FOM variant (F_b) is better than the standard FOM and one of its existing variants (F_{nn}) to detect edges based on the experiment results. The edge detectors from the FOM variants give single response on boundaries but are affected by noise and some textures.

Our future work will further investigate fitness functions based on FOM so that the evolved edge detectors have the ability to suppress noise and some textures. Different image datasets will be used to test evolved edge detectors' performance.

References

1. Baddeley, J.A.: An error metric for binary images. In: Proceedings of the International Workshop on Robust Computer Vision, pp. 59–78 (1992)
2. Bolis, E., Zerbi, C., Collet, P., Louchet, J., Lutton, E.: A GP Artificial Ant for Image Processing: Preliminary Experiments with EASEA. In: Miller, J., Tomassini, M., Lanzi, P.L., Ryan, C., Tetamanzi, A.G.B., Langdon, W.B. (eds.) EuroGP 2001. LNCS, vol. 2038, pp. 246–255. Springer, Heidelberg (2001)
3. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(6), 679–698 (1986)

4. Ebner, M.: On the edge detectors for robot vision using genetic programming. In: Proceedings of Horst-Michael Groβ, Workshop SOAVE 1997 - Selbstorganisation von Adaptivem Verhalten, pp. 127–134 (1997)
5. Fu, W., Johnston, M., Zhang, M.: Genetic programming for edge detection: a global approach. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 254–261 (2011)
6. Fu, W., Johnston, M., Zhang, M.: Genetic programming for edge detection based on accuracy of each training image. In: Proceedings of the 24th Australasian Joint Conference on Artificial Intelligence, pp. 301–310 (2011)
7. Ganesan, L., Bhattacharyya, P.: Edge detection in untextured and textured images: a common computational framework. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 27(5), 823–834 (1997)
8. Golonek, T., Grzechca, D., Rutkowski, J.: Application of genetic programming to edge detector design. In: Proceedings of the International Symposium on Circuits and Systems, pp. 4683–4686 (2006)
9. Harris, C., Buxton, B.: Evolving edge detectors with genetic programming. In: Proceedings of the First Annual Conference on Genetic Programming, pp. 309–314 (1996)
10. Hollingworth, G.S., Smith, S.L., Tyrrell, A.M.: Design of highly parallel edge detection nodes using evolutionary techniques. In: Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing, pp. 35–42 (1999)
11. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6(2), 65–70 (1979)
12. Huttenlocher, D., Klanderman, G., Rucklidge, W.: Comparing images using the Hausdorff distance. IEEE Transactions on Pattern Analysis and Machine Intelligence 15(9), 850–863 (1993)
13. Kadar, I., Ben-Shahar, O., Sipper, M.: Evolution of a local boundary detector for natural images via genetic programming and texture cues. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1887–1888 (2009)
14. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(5), 530–549 (2004)
15. Moreno, R., Puig, D., Julia, C., Garcia, M.A.: A new methodology for evaluation of edge detectors. In: Proceedings of the 16th IEEE International Conference on Image Processing (ICIP), pp. 2157–2160 (2009)
16. Papari, G., Petkov, N.: Edge and line oriented contour detection: state of the art. Image and Vision Computing 29, 79–103 (2011)
17. Pinho, A.J., Almeida, L.B.: Edge detection filters based on artificial neural networks. In: Proceedings of the 8th International Conference on Image Analysis and Processing, pp. 159–164 (1995)
18. Poli, R.: Genetic programming for image analysis. In: Proceedings of the First Annual Conference on Genetic Programming, pp. 363–368 (1996)
19. Pratt, W.K.: Digital Image Processing: PIKS Inside, 3rd edn. Wiley (2001)
20. Wang, J., Tan, Y.: A novel genetic programming based morphological image analysis algorithm. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 979–980 (2010)
21. Zhang, Y., Rockett, P.I.: Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 795–802 (2005)

An Evolutionary Algorithm for the Over-constrained Airport Baggage Sorting Station Assignment Problem

Amadeo Ascó[1], Jason A.D. Atkin[1], and Edmund K. Burke[2]

[1] School of Computer Science, University of Nottingham
a.asco@cs.nott.ac.uk, Jason.Atkin@nottingham.ac.uk
[2] University of Stirling
e.k.burke@stir.ac.uk

Abstract. Airport baggage sorting stations are the places at which the baggage for each flight accumulates for loading into containers or directly onto the aircraft. In this paper the multi-objective and multi-constraint problem of assigning Baggage Sorting Stations in an Airport is defined and an Evolutionary Algorithm is presented, which uses a number of different operators to find good solutions to the problem. The contributions of these different operators are studied, giving insights into how the appropriate choice may depend upon the specifics of the problem at the time.

1 Introduction

Passengers at an airport proceed to the check-in desks assigned to their flight where they leave their baggage to be processed. The baggage enters the baggage system at this point, where it is processed and delivered to baggage sorting stations (BSSs). There it is temporarily stored before being sorted and transported by carts to the aircraft.

This paper considers the task of assigning baggage sorting stations to flights at an airport, which will already have been allocated to stands along piers around the terminals. The aim is to maximise the number of flights which are assigned to sorting stations, using the sorting stations which are most conveniently situated for the stands when possible, while ensuring that the gaps between sorting station usage are at least as large as a specified buffer time, to cope with real world perturbations and uncertainties.

Research into a similar problem was previously performed by [1], who described the problem and applied the activity selection algorithm. In [2], we studied the quality of solutions which could be obtained from a variety of constructive algorithms. Different algorithms were found to prefer different objectives and a selection of these constructed solutions has been used for initial solutions of the evolutionary algorithm which is described in this paper.

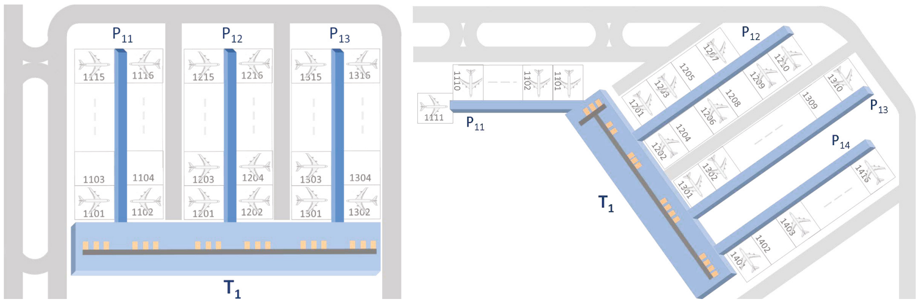
The Airport Baggage Sorting Station Assignment Problem (ABSSP) has many similarities to the Airport Gate Assignment Problem (AGAP), which has had considerable study in the past. The basic gate assignment problem is a quadratic assignment problem, shown in [3] to be NP-hard. The approach presented in [4] used a Genetic Algorithm (GA) for the AGAP with uniform crossover. The method described in [5] applied a GA, using an elitist selection method, with an additional mechanism composed of mutation operators to repair the infeasible solutions which resulted from applying the crossover operator.

We initially applied CPLEX and Gurobi to the Integer Linear Programming (ILP) representation of the ABSSP, both with and without an initial solution chosen as the fittest of the constructed solutions presented in [2]. Both systems were allowed a maximum runtime of 24 hours. Although CPLEX ran out of memory (using over 84GB of disk space) after 16hrs on a 2.4GHz Windows 7, 64bit machine with 4GB RAM. The Results were inadequate, so we developed the evolutionary algorithm which is described in this paper. This has turned out to perform much better, even with a much shorter search time.

2 The Airport Baggage Sorting Station Problem

Aircraft are usually parked at their allocated stand, around an airport terminal. Two example layouts which are considered in this paper are shown in Figure 1, showing aircraft parked around the piers of a terminal, and illustrating the position of baggage sorting stations at the bases of the piers. The aim of this work is to allocate each flight to a sorting station. In general, it is better to allocate a flight to a sorting station on the same pier, but it is not usually important which of the sorting stations in that group the flight is allocated to. Furthermore, a minimum gap should be ensured between flights.

Let $i \in \{0, \ldots, N\}$ denote a sorting station where sorting station 0 represents a dummy sorting station to which flights are assigned, if they cannot be assigned to a real sorting station. Let $j \in \{1, \ldots, M\}$ denote a flight. Let T_j denote the required service time for flight j (1 hour for short haul and $1\frac{1}{3}$ hours for long haul) and let B_j denote the desired buffer time for flight j - the time for which its sorting station should be idle prior to this flight being serviced. B_j is assumed to be 15 minutes for short haul and 30 minutes for long haul flights. Let e_j denote the end service time for flight j and let t_j denote the target starting service



(a) 3 piers and 48 stands. (b) 4 piers and 46 stands.

Fig. 1. Topologies

time for flight j ($t_j = e_j - T_j - B_j$). Let y_{ij} be a decision variable defined to be 1 if flight j is assigned to sorting station i, and 0 otherwise. The aim is to find y_{ij} values such that the objective function (4) is maximised, subject to the constraints expressed by (1), (2) and (3).

$$\sum_{i=0}^{N} y_{ij} = 1 \quad \forall j \in \{1, \ldots, M\} \tag{1}$$

$$y_{ij} + y_{il} \leq 1 \;\; \forall i \in \{1, \ldots, N\}, j \neq l, t_l + B_l < e_j \leq e_l \tag{2}$$

Equation (1) states that each flight can be assigned either to exactly one real sorting station or to the dummy sorting station. Inequality (2) states that flights cannot be assigned to the same real sorting station if their service times overlap.

If the service periods of two flights are closer together than the buffer time of the second flight, then they can still be allocated to the same sorting station, but the buffer time will have to be reduced. The required reduction in buffer time for flight l is denoted r_l and can be calculated using (3), where $0 \leq r_l \leq B_l \; \forall \; l \in \{1, \ldots, M\}$.

$$r_l \geq (y_{ij} + y_{il} - 1) \cdot (e_j - t_l) \;\; \forall i \in \{1, \ldots, N\} \tag{3}$$

Although this is inherently a multi-objective problem, the importance of ensuring maximal assignment of flights to sorting stations (top priority) and the relative importance of keeping reasonable buffer times (second priority) allows these objectives to be combined into a single compound objective (4) with weights W_1, W_2 and W_3 chosen to implement these priorities.

$$\max \left(W_1 \cdot \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} - W_2 \cdot \sum_{j=1}^{M} r_j - W_3 \cdot \sum_{i=1}^{N} \left(C_j \cdot \sum_{j=1}^{M} y_{ij} \cdot d_{ij} \right) \right) \tag{4}$$

The first element in (4) aims to maximise the assignment of flights to sorting stations, the second aims to minimise the reduction in buffer times and the third aims to minimise some distance cost associated with the assignments, where C_j is a factor related to the amount of baggage for flight j (assumed to be 1 in all cases for this paper) and d_{ij} is a measure of the distance or cost incurred from assigning sorting station i to flight j. This aims to ensure that flights are allocated to appropriate sorting stations.

3 Algorithm

In this paper we describe an Evolutionary Algorithm which we have developed for solving the Airport Baggage Sorting Station Assignment Problem (ABSSAP). Our algorithm uses a number of custom problem-specific operators and we have categorised them as either mutation operators, if they modify a single existing solution, or crossover operators, if they combine multiple solutions. We describe the algorithm, selection methods and operators in this section.

3.1 Steady State Evolutionary Algorithm

A steady state GA does not replace all the parent solutions by their children as in generational GAs. In our Steady State Evolutionary Algorithm (SSEA) the next population is obtained by applying the population selector (S_p) to the current population. One of the operators is applied to the required number of individuals which are chosen by applying the member selector (S_m) to the population. This selection and modification is called an iteration and is repeated L times for each population. The newly obtained individuals are then added to the population which will then constitute the new current population and a population selector is used to determine which population members to keep. This is repeated until the termination condition is reached, as shown in Algorithm 1.

Algorithm 1. Steady State Evolutionary Algorithm

```
   input : Initial population $P_0$
   input : Number of iterations in a generation $L \in \mathbb{Z}^+, L > 0$
   input : Operators; $O_1 : p_1, O_2 : p_2, \cdots$ and $O_R : p_R$ where
           $0 < p_k \leq 1 \; \forall \; k \in (1, \cdots, R)$ and $\sum_{k=1}^{R} p_k = 1$
   input : Population selector, $S_p$
   input : Population member(s) selector, $S_m$
1  begin
2      $P \leftarrow P_0$; // initialise population
3      repeat
4          $P \leftarrow S_p(P)$; // get the current population for this generation
5          $P_t \leftarrow \emptyset$; // empty temporary population
6          $i \leftarrow 0$; // initialise the generation
7          repeat
8              Select randomly an operator, $O_k$; // roulette wheel selector
9              $Q \leftarrow S_m(P, O_k)$; // select population member(s)
10             $Q \leftarrow O_k(Q)$; // generate new solution(s)
11             $P_t \leftarrow P_t \cup Q$; // add new solution(s)
12             $i \leftarrow i + 1$;
13         until $i = L$ or Termination Condition;
14         $P \leftarrow P \cup P_t$; // add new solutions to the current population
15     until Termination Condition;
16     return $P$;
17 end
```

3.2 Population Selectors

The population selector (S_p) has the responsibility of selecting the solutions within the current population that will form part in the next generation. A comprehensive analysis of selection schemes used in Evolutionary Algorithms can be found in [6]. We present below an overview of the population selection algorithms used.

Elitist Sampling (ES): Select the fittest μ population members from the current population.

Stochastic Universal Modified Sampling (SUMS): Stochastic Universal Sampling was introduced by [7]. SUMS was used since the order of magnitude of the fitness values are much larger than the differences between the fitness values, so it was not appropriate to use Stochastic Universal Sampling. The population members are mapped by sections, as in Roulette Tournament Selection, $[p_{i-1}, p_i)$ with $p_0 = 0$ for $p_i = \frac{\sum_{j=1}^{i} f_j - F}{\sum_{j=1}^{\lambda} f_j - F}$ where $F = f_\lambda - (f_{\lambda-1} - f_\lambda)$, f_j corresponds to the fitness of solution j and λ is the current population size. μ individuals are selected by obtaining an initial random number (rnd) within $[0, \frac{1}{\mu})$ and subsequent ones spread $\frac{1}{\mu}$ from the previous one. Solution i is selected once per each $p_{i-1} \leq rnd + \frac{j-1}{\mu} < p_i \forall j \in \{1, \ldots, \mu\}$.

Remainder Stochastic Sampling (RSS): This method was also investigated due to its diversity retention properties (see [8]), but provided worse results, so we have not discussed the results in this paper.

3.3 Member Selectors

The member selectors (S_m) distribute the chances that the individuals have to take part in the production of new offsprings in a generation. The roulette wheel selection method was used as the member selector on this paper.

3.4 Operators

Whenever a time has to be determined (for instance for a start or end of a time range) a uniform random variable is used so that any time within the time range of the flights under consideration has an equal probability of being chosen.

Mutation. Guided mutation operators are introduced here which are based upon local search operators which guarantee to generate feasible solutions.

Dummy Single Exchange Mutation Operator (DSEMO). This operator is equivalent to the 'Apron Exchange Move' used by [9] and [10]. A solution is selected from the population by the member selector (S_m) then a new solution is built by moving a flight from the 'dummy' sorting station to a randomly selected sorting station, potentially moving another flight back onto the 'dummy' sorting station if it can no longer fit in. This operator may increase the number of assignments if the operation does not move a flight back onto the 'dummy' sorting station. It requires some flights to be unassigned in the parent solution.

Multi Exchange Mutation Operators. A time period is randomly selected, t_{rs} to t_{re}. All assignments for which the base service times are entirely within the time period are then moved to the next sorting station in the set, as shown in Figure 2,

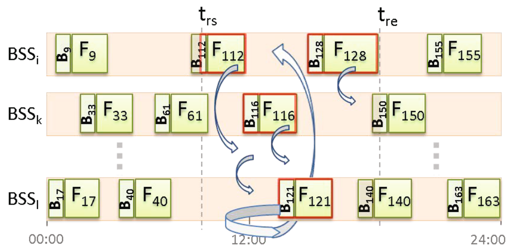


Fig. 2. Multi Exchange Mutation Operator

as long as they will fit. These operators generalise the 'Interval Exchange Move' which was presented by [10], and cannot increase the number of assignments. Two variants have been developed:

1. *Multi Exchange with Fixed Number of Resources (MEFNRn)*: The number of sorting stations to exchange flights between is fixed at n, where $2 \leq n \leq N$.
2. *Multi Exchange with Random Number of Resources (MERNRn)*: The number of sorting stations to exchange flights between is randomly chosen each time, between 2 and n, where $2 \leq n \leq N$.

Multi Exchange By Pier Mutation Operators. These operators are a specialised case of the Multi Exchange Mutation Operators, where the sorting station selection element ensures that no two consecutive sorting stations in the set are on the same pier. The idea is to improve the distance objective by encouraging the movement of assignments between piers. Again, this operator cannot increase the number of assignments. As for the Multi Exchange Mutation Operators, two variants have been created: *Multi Exchange By Pier with Fixed Number of Resources* (MEBPFNRn) and with Random Number of Resources (MEBPRNRn).

Range Multi Exchange Mutation Operators. These are the same as the Multi Exchange Mutation Operators. However they add an additional feasibility recovery step when flights cannot be moved. Such flights are added to the set of flights which will be considered for assignment to the next sorting station, potentially reducing this number. Again, this operator cannot increase the number of assignments. As for the Multi Exchange Mutation Operators, two variants have been created: *Multi Exchange with Fixed Number of Resources* (RMEFNRn) and with Random Number of Resources (RMERNRn).

Crossover (reproduction). The crossover operators involve the generation of new solutions from multiple parents. Each parent will be chosen using the Population Member Selector (S_m) and multiple child solutions will be generated.

2-Point Crossover (C2P): Two points in time are randomly selected to generate a time window. A child solution is created from each parent by re-assigning all flights within the time window to the sorting station they were assigned to in the other parent. Although the flight timings are identical across all solutions, flights in the swapped region may overlap flights which are not swapped for some sorting stations. Such overlapping flights in the swapped region are re-assigned to other sorting stations, if possible, or assigned to the dummy sorting station otherwise.

1-Point Crossover (C1P): This is a specific case of the above 2-Point Crossover, where the window extends to the end time of the solution.

4 Results

Since it would be unrealistic to assume that baggage from a flight at a stand in one terminal is serviced by a baggage sorting station in another terminal (e.g. passengers usually go through security and board flights from the same terminal at which their baggage was checked in), this paper is centred on a single terminal.

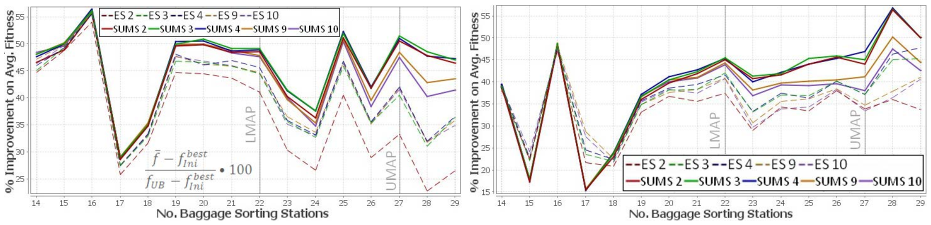
For this paper, we assumed that all flights require only one sorting station. Two datasets, provided by NATS (formerly National Air Traffic Services) were used, from 16^{th} December 2009 (194 flights) and from 1^{st} March 2010 (163 flights) for Heathrow terminal 1. All the experiments were executed for 800,000 iterations. The same random number generator was used throughout.

The Steady State Evolutionary Algorithm (SSEA) which was used in these experiments had $L = 1$ and a population size of 30. The value of W_1 was determined by running initial experiments with our Evolutionary Algorithm (EA) using different values, from 15 to 100, for the data set of 16^{th} December 2009, 3-pier topology, a fixed W_2 of 0.008 and W_3 of 1. A value of $W_1 = 90$ appeared to give an appropriate balance between the objectives and was adopted. The SSEAs were run with just one of the presented operators at a time, to show their individual behaviour in the considered landscapes. A distance of $1unit$ is assumed between sides of the same pier, $2units$ to move between piers and a distance of zero is assumed for those assignments that are as close as possible.

The Lower Maximum Assignment Point (LMAP) is the minimum number of sorting stations needed to assign all flights without buffer times. The Upper Maximum Assignment Point (UMAP) is the minimum number of sorting stations needed to assign all flights without reducing the service time. The LMAP and UMAP points will be observed to be useful when interpreting the results.

4.1 Experiments

Figure 3(a) shows the percentage improvement in average fitness of the results for the Multi Exchange with Fixed Number of Resources (MEFNRn) operator, $n \in \{2, 3, 4, 9, 10\}$, with the population selectors Elitist (ES) and Stochastic Universal Modified Sampling (SUMS), applied to different numbers of baggage sorting stations (BSSs) with respect to the best initial solution (f_{Ini}^{best}, 0%) and the Upper Bound obtained by CPLEX (f_{UB}, 100%). The achieved improvements range from 25% to over 55%, depending on the number of BSSs. It should be



(a) 4-pier and 46 stands. (b) 3-pier and 48 stands.

Fig. 3. Improvement on average fitness for 194 flights, MEFNRn, $n \in \{2, 3, 4, 9, 10\}$

noted that 100% improvement corresponds to solutions which reach the upper bound for their specific case, whereas 0% corresponds to no improvement over the best initial (constructed) solution.

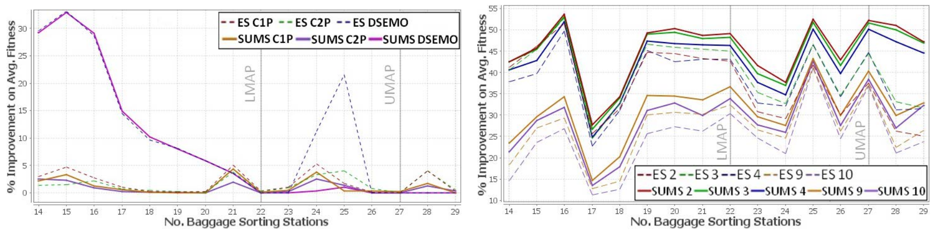
When the number of sorting stations is lower than the Lower Maximum Assignment Point (LMAP), it is impossible to fully assign all of the flights. Since the initial solutions already provide at least one solution which is guaranteed to achieve the maximum number of assignments for the given number of baggage sorting stations ([2]), the EA can only improve upon this by improving the 'distance' and 'reduction in service' objectives. For example, it may be possible to swap out flights in order to lower the reduction in service time. The results show that the largest benefits in this region are due to the ability to swap which flights are assigned, however, in practice it is unlikely for an airport to operate in this mode - where some flights cannot actually be serviced.

In contrast, full assignment can be achieved between the LMAP and Upper Maximum Assignment Point (UMAP), but only by reducing the service times. As long as there are solutions with unassigned flights, it may be advantageous to use the operators which use the dummy sorting station. These operators do not contribute once all solutions have full assignment, and should no longer be used.

Finally, after the UMAP full assignment can be achieved without reduction in service time so the only objective that contributes to the fitness is the distance. The fitness is still improved by the EA but by a much smaller amount since the weights and the values of this objective are much lower than in the others.

All of the operators always improve upon the initial solutions by at least 10% except for the crossover and DSEMO operators (Figure 4(a)). Crossover alone could not always find improvements over the constructed initial solutions, since crossover operators are dependent upon the quality of the building blocks which are already present in the population. If the current solutions do not contain many useful differences, then the crossover will not be able to improve the solution. Similarly initial solutions of lower fitness may contain better building blocks from the point of view of the crossover operator, but may be removed too early in the search.

When $N < LMAP$, DSEMO provided better solutions in most cases than all of the crossover operators which are considered in this paper, irrespective of the selection method used, but the solutions are of lower fitness than those obtained by the other considered mutation operators.



(a) C1P, C2P and DSEMO. (b) RMEFNRn, $n \in \{2, 3, 4, 9, 10\}$.

Fig. 4. Improvement on average fitness for 4-pier and 194 flights

For $N \geq LMAP$ the solutions obtained by DSEMO are generally no better than the initial solution and the main reason for this is that the highly fit solutions in this area will already have all of their flights assigned. However, not all individuals in the population will have maximal assignments, so in some of the instances, where the operator is applied to a lower fitness solution, doing so can reach better areas of the search space and achieve a better fitness. Thus, the use of DSEMO may help the algorithm to reach different parts of the search space when the initial population is composed of more diverse solutions. These results imply that it may be wise to adjust the use of DSEMO dynamically to accommodate to the changing structure of the population.

With the Multi Exchange with Fixed Number of Resources (MEFNRn), the results observed for the different population selectors do not differ much on average for different values of n. This is not the case with the Range Multi Exchange with Fixed Number of Resources (RMEFNRn), where the obtained solutions deteriorate as n increases, so $n = 2$ and $n = 3$ provided the best solutions, as shown in Figure 4(b). When considering the execution speed of using these operators, lower values of n are also faster, as is also true for MEFNRn and MEBPFNRn.

Figures 3(a) and 4(b) show that SUMS provides better average results than an elitist strategy, which could be considered to be a consequence of SUMS preserving the diversity better than the ES.

In general the results did not differ across airport topologies (i.e. 3 or 4 pier models) or days (i.e. number of flights), however there were some minor differences. For example, in the 3-pier topology, with 194 flights, shown in Figure 3(b), Elitist selection performed better than SUMS for 17 sorting stations and 15 sorting stations (163 flights). We believe that this is because the constructive algorithms perform well on these problems, so that the initial population contains solutions which are much better, and the diversity which SUMS introduces is not beneficial for the search. This behaviour is not present in the 4-pier topology, for either of the datasets for the range of sorting stations studied, shown in Figure 3(a).

5 Conclusions

The aim of this research was to gain more general insights into the appropriate operator choices, especially since some operators (such as crossover) are slower than others to apply, and the appropriate operator percentages may differ depending upon the situation.

The Dummy Single Exchange Mutation Operator (DSEMO) extends the search to other areas of the search space which may help to improve the solutions, but it is only useful when there are unassigned flights, eg. for N $<$ LMAP. Whereas for N $\geq$ LMAP, the DSEMO should only be used when the selected solution from the population has unassigned flights, most commonly nearer the start of the search.

Given the diverse ways in which the studied operators work it is expected that their combination will further improve the solutions. Furthermore, the combination of different operators together with an adaptive method of selecting operators seems to be the most promising approach for future work.

In further research we plan to shed some light upon the effects of changing the value of L, both statically and dynamically, as well as to consider statically and dynamically changing the proportions of usage of different operators.

Acknowledgments. We are grateful to NATS Ltd and to EPSRC for providing the funding for this project, and especially to John Greenwood (NATS) for his continuous support.

References

1. Abdelghany, A., Abdelghany, K., Narasimhan, R.: Scheduling baggage-handling facilities in congested airports. Journal of Air Transport Management 12(2), 76–81 (2006)
2. Ascó, A., Atkin, J.A.D., Burke, E.K.: The airport baggage sorting station allocation problem. In: Proceedings of the 5th Multidisciplinary International Scheduling Conference, MISTA 2011, Phoenix, Arizona, USA (August 2011)
3. Obata, T.: The quadratic assignment problem: Evaluation of exact and heuristic algorithms. Technical report, Rensselaer Polytechnic Institute, Troy, New York (1979)
4. Hu, X., Di Paolo, E.: An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 55–62 (September 2007)
5. Bolat, A.: Models and a genetic algorithm for static aircraft-gate assignment problem. The Journal of the Operational Research Society 52, 1107–1120 (2001)
6. Blickle, T., Thiele, L.: A comparison of selection schemes used in evolutionary algorithms. Evolutionary Computation 4(4), 361–394 (1996)
7. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and Their Application, pp. 14–21. L. Erlbaum Associates Inc., Hillsdale (1987)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
9. Ding, H., Lim, A., Rodrigues, B., Zhu, Y.: Aircraft and gate scheduling optimization at airports. Hawaii International Conference on System Sciences 3, 1530–1605 (2004)
10. Ding, H., Rodrigues, A.L., Zhu, Y.: The over-constrained airport gate assignment problem. Computers and Operations Research 32(7), 1867–1880 (2005)

A Non-parametric Statistical Dominance Operator for Noisy Multiobjective Optimization

Dung H. Phan and Junichi Suzuki

Deptartment of Computer Science, University of Massachusetts, Boston, USA
{phdung,jxs}@cs.umb.edu

Abstract. This paper describes and evaluates a new noise-aware dominance operator for evolutionary algorithms to solve the multiobjective optimization problems (MOPs) that contain noise in their objective functions. This operator is designed with the Mann-Whitney U-test, which is a non-parametric (i.e., distribution-free) statistical significance test. It takes objective value samples of given two individuals, performs a U-test on the two sample sets and determines which individual is statistically superior. Experimental results show that it operates reliably in noisy MOPs and outperforms existing noise-aware dominance operators particularly when many outliers exist under asymmetric noise distributions.

Keywords: Evolutionary multiobjective optimization algorithms, noisy optimization, uncertainties in objective functions.

1 Introduction

This paper focuses on noisy multiobjective optimization problems (MOPs):

$$\left.\begin{aligned} \min\ F(\boldsymbol{x}) = [f_1(\boldsymbol{x}) + \epsilon_1, \cdots, f_m(\boldsymbol{x}) + \epsilon_m]^T \in \mathcal{O} \\ \text{subject to } \boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T \in \mathcal{S} \end{aligned}\right\} \tag{1}$$

$\mathcal{S}$ denotes the decision variable space. $\boldsymbol{x} \in \mathcal{S}$ is a solution candidate that consists of n decision variables. It is called an *individual* in evolutionary multiobjective optimization algorithms (EMOAs). $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ consists of m real-value objective functions, which produce the objective values of $\boldsymbol{x}$ in the objective space $\mathcal{O}$. In MOPs, objective functions often conflict with each other. Thus, there rarely exists a single solution that is optimum with respect to all objectives. As a result, EMOAs often seek the optimal trade-off solutions, or *Pareto-optimal* solutions, by balancing the trade-offs among conflicting objectives. A notion of *Pareto dominance* plays an important role to seek Pareto optimality in EMOAs. An individual $\boldsymbol{x} \in \mathcal{S}$ is said to *dominate* another individual $\boldsymbol{y} \in \mathcal{S}$ (denoted by $\boldsymbol{x} \succ \boldsymbol{y}$) iif the both of the following two conditions are hold: (1) $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})\ \forall\ i\ = 1, \cdots, m$ and (2) $f_i(\boldsymbol{x}) < f_i(\boldsymbol{y})\ \exists\ i\ =\ 1, \cdots, m$.

In Eq. 1, ϵ_i is a random variable that represents noise in the i-th objective function. Given noise, each objective function can yield different objective values for the same individual from time to time. Noise in objective functions often interferes with a dominance operator, which determines dominance relationships among individuals. For example, a dominance operator may mistakenly judge that an inferior individual dominates an superior one. Defects in a dominance operator significantly degrade the performance to solve MOPs [2,11].

In order to address this issue, this paper proposes a new noise-aware dominance relationship, called *U-dominance*, which extends the classical Pareto dominance. It is designed with the Mann-Whitney U-test (U-test in short), which is a non-parametric (i.e., distribution free) statistical significance test [12]. The U-dominance operator takes objective value samples of given two individuals, performs a U-test on the two sample sets to examine whether it is statistically confident enough to judge a dominance relationship between the two individuals, and determines which individual is statistically superior/inferior. This paper evaluates the U-dominance operator by integrating it with NSGA-II [4], a well-known EMOA. Experimental results demonstrate that the U-dominance operator reliably performs dominance ranking operation in noisy MOPs and outperforms existing noise-aware dominance operators particularly when many outliers exist under asymmetric noise distributions.

2 Related Work

There exist various existing work to handle uncertainties in objective functions by modifying the classical Pareto dominance operator [2,11]. Most of them assume particular noise distributions in advance; for example, normal distributions [1,8–10,13], uniform distributions [14] and Poisson distributions [6,17]. Given a noise distribution, existing noise-aware dominance operators collect objective value samples from each individual [6,8,9,14,17], or each cluster of individuals [1], in order to determine dominance relationships among individuals. Those existing operators are susceptible to noisy MOPs in which noise follows unknown distributions. In contrast, the U-dominance operator assumes no noise distributions in advance because, in general, it is hard to predict and model them in many (particularly, real-world) MOPs. Instead of estimating each individual's objective values based on a given noise distribution, the U-dominance operator estimates the impacts of noise on objective value samples and determines whether it is statistically confident enough to compare individuals.

Voß et al. [16] and Boonma et al. [3,18] study similar dominance operators to the U-dominance operator in that they assume no noise distributions in objective value samples of each individual and statistically examine the impacts of noise on those samples. Voß et al. propose an operator that considers an uncertainty zone around the median of samples on a per-objective basis. A dominance decision is made between two individuals only when their uncertainty zones do not overlap in all objectives. The uncertainty zone can be the inter-quartile range (IQR) of samples or the bounding box based on the upper and lower quartiles (BBQ) of samples. Unlike the U-dominance operator, this operator is susceptible to high noise strength and asymmetric heavy-tailed noise distributions. Boonma et al. propose an operator that classifies objective value samples with a support vector machine, which can be computationally expensive. In contrast, the U-dominance operator is designed lightweight; it requires no classification and learning.

Algorithm 1. The U-Dominance Operator: `uDominance(`$\mathcal{A}$`, `$\mathcal{B}$`, `α`)`

```
Input: $\mathcal{A}$ and $\mathcal{B}$, Objective value samples of individuals $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively
Input: $\alpha$, The confidence level used in a $U$-test
Output: Dominance relationship between $\boldsymbol{a}$ and $\boldsymbol{b}$
1: $p_{\mathcal{A}}$, $p_{\mathcal{B}} = 1$
2: for each (the $i$-th) objective do
3:     Perform a $U$-test on $\mathcal{A}$ and $\mathcal{B}$ in the $i$-th objective.
4:     if $\mathcal{A}$ is superior than $\mathcal{B}$ with the confidence level $\alpha$ in the $i$-th objective then
5:         $p_{\mathcal{A}} = 1 * p_{\mathcal{A}}$
6:         $p_{\mathcal{B}} = 0$
7:     else if $\mathcal{B}$ is superior than $\mathcal{A}$ with the confidence level $\alpha$ in the $i$-th objective then
8:         $p_{\mathcal{A}} = 0$
9:         $p_{\mathcal{B}} = 1 * p_{\mathcal{B}}$
10:    end if
11: end for
12: if $p_{\mathcal{A}} = 1$ and $p_{\mathcal{B}} = 0$ then
13:     return 1 /*** $\boldsymbol{a}$ U-dominates $\boldsymbol{b}$. ***/
14: else if $p_{\mathcal{A}} = 0$ and $p_{\mathcal{B}} = 1$ then
15:     return $-1$ /*** $\boldsymbol{b}$ U-dominates $\boldsymbol{a}$. ***/
16: else
17:     return 0 /*** $\boldsymbol{a}$ and $\boldsymbol{b}$ are non-U-dominated. ***/
18: end if
```

3 The *U*-Dominance Operator

U-dominance is a new noise-aware dominance that is designed with the Mann-Whitney U-test (U-test in short), which is a non-parametric (i.e., distribution-free) statistical significance test [12]. An individual $\boldsymbol{a}$ is said to *U-dominate* an individual $\boldsymbol{b}$ (denoted by $\boldsymbol{a} \succ_U \boldsymbol{b}$) with the confidence level α iif:

– In all objectives, $\boldsymbol{b}$ is not superior than $\boldsymbol{a}$ using a U-test with the confidence level of α.
– In at least one objective, $\boldsymbol{a}$ is superior than $\boldsymbol{b}$ using a U-test with the confidence level of α.

The U-dominance operator takes objective value samples of given two individuals, estimates the impacts of noise on the samples through a U-test, examines whether it is statistically confident enough to judge a dominance relationship between the two individuals, and determines which individual is statistically superior/inferior (Algorithm 1).

Given two sets of objective value samples, $\mathcal{A}$ and $\mathcal{B}$, which are collected from two individuals $\boldsymbol{a}$ and $\boldsymbol{b}$, a U-test is performed on $\mathcal{A}$ and $\mathcal{B}$ in each objective (Algorithm 1). First, $\mathcal{A}$ and $\mathcal{B}$ are combined into a set $\mathcal{S} = \mathcal{A} \cup \mathcal{B}$. The samples in $\mathcal{S}$ are sorted in ascending order based on their objective values in an objective in question. Then, each sample obtains its *rank*, which represents the sample's position in $\mathcal{S}$. The rank of one is given to the first sample in $\mathcal{S}$ (i.e., the best sample that has the minimum objective value). If multiple samples tie, they receive the same rank, which is equal to the mean of their positions in $\mathcal{S}$. For example, if the first two samples tie in $\mathcal{S}$, they receive the rank of 1.5 ($\frac{1+2}{2}$).

Once ranks are assigned to samples, the *rank-sum* values, $R_{\mathcal{A}}$ and $R_{\mathcal{B}}$, are computed for $\mathcal{A}$ and $\mathcal{B}$, respectively. ($R_{\mathcal{A}}$ sums up the ranks for the samples in $\mathcal{A}$.) For a large sample size (> 10), the sampling distributions of $R_{\mathcal{A}}$ and $R_{\mathcal{B}}$

are approximately normal [12]. Therefore, the standardized value of a rank-sum is a standard normal deviate whose significance can be tested under the standard normal distribution. The standardized value of $R_{\mathcal{A}}$ is given as follows.

$$z_{R_{\mathcal{A}}} = \frac{R_{\mathcal{A}} - \mu_{R_{\mathcal{A}}}}{\sigma_{R_{\mathcal{A}}}} \tag{2}$$

$\mu_{R_{\mathcal{A}}}$ and $\sigma_{R_{\mathcal{A}}}$ denote the mean and standard deviation of $R_{\mathcal{A}}$, respectively.

$$\mu_{R_{\mathcal{A}}} = \frac{|\mathcal{A}| \times (|\mathcal{A}| + |\mathcal{B}| + 1)}{2} \tag{3}$$

$$\sigma_{R_{\mathcal{A}}} = \sqrt{\frac{|\mathcal{A}| \times |\mathcal{B}| \times (|\mathcal{A}| + |\mathcal{B}| + 1)}{12}} \tag{4}$$

With the confidence level of α, the U-test determines that $\mathcal{A}$ and $\mathcal{B}$ are not drawn from the same distribution if $F(z_{R_{\mathcal{A}}}) \leq (1 - \alpha)$ or $F(z_{R_{\mathcal{A}}}) \geq \alpha$. ($F(z)$ is the cumulative distribution function of the standard normal distribution.) This means that $\mathcal{A}$ and $\mathcal{B}$ are significantly different with the confidence level of α. The U-test concludes that $\boldsymbol{a}$ is superior than $\boldsymbol{b}$ with respect to an objective in question if $F(z_{R_{\mathcal{A}}}) \leq (1 - \alpha)$ and that $\boldsymbol{b}$ is superior than $\boldsymbol{a}$ if $F(z_{R_{\mathcal{A}}}) \geq \alpha$.

This paper integrates the U-dominance operator with NSGA-II [4], a well-known EMOA. It is integrated with a binary tournament operator and a dominance ranking operators in NSGA-II. NSGA-II uses binary tournament in its parent selection process, which selects a parent individual to be used in crossover, and uses dominance ranking in its environmental selection process, which selects the next-generation population from the union of the current population and its offspring [4]. Fig. 1 shows how to perform binary tournament with with the U-dominance operator. In Lines 1 and 2, two individuals $\boldsymbol{a}$ and $\boldsymbol{b}$ are randomly drawn from the population $\mathcal{P}$. Then, in Lines 3 and 4, their objective value samples are obtained to invoke the U-dominance operator at Line 5. Based on the U-dominance relationship between $\boldsymbol{a}$ and $\boldsymbol{b}$, one of them is returned as a parent individual (Lines 6 to 16).

Fig. 2 shows how to rank individuals with the U-dominance operator. From Line 1 to 12, U-dominance relationships are determined among N individuals in the population $\mathcal{P}$. The U-dominance operator is invoked in Line 5. Unlike the classical Pareto dominance, U-dominance relationships are not transitive. When $\boldsymbol{a} \succ_U \boldsymbol{b}$ and $\boldsymbol{b} \succ_U \boldsymbol{c}$, $\boldsymbol{a} \succ_U \boldsymbol{c}$ is not guaranteed. When objective functions contain high-level noise, $\boldsymbol{c}$ might even U-dominate $\boldsymbol{a}$. If a loop exists in U-dominance relationships (e.g., $\boldsymbol{a} \succ_U \boldsymbol{b}$, $\boldsymbol{b} \succ_U \boldsymbol{c}$ and $\boldsymbol{c} \succ_U \boldsymbol{a}$), the U-dominance operator deletes the U-dominance relationships among $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$, and concludes that they are non-U-dominated with each other (Line 13 to 15 in Algorithm 2).

4 Experimental Evaluation

This section evaluates the U-dominance operator by integrating it with NSGA-II. This variant of NSGA-II is called NSGA-II-U in this paper. It is compared with the following five other variants of NSGA-II.

```
Input: $\mathcal{P}$, The population of $N$ individuals
Output: A parent individual to be used in crossover
1: $\boldsymbol{a}$ = randomSelection($\mathcal{P}$)
2: $\boldsymbol{b}$ = randomSelection($\mathcal{P}$)
3: $\mathcal{A}$ = samplesOf($\boldsymbol{a}$)
4: $\mathcal{B}$ = samplesOf($\boldsymbol{b}$)
5: $r$ = uDominance($\mathcal{A}$, $\mathcal{B}$, $\alpha$)
6: if $r = 1$ then
7:     return $\boldsymbol{a}$
8: else if $r = -1$ then
9:     return $\boldsymbol{b}$
10: else
11:     if random() $> 0.5$ then
12:         return $\boldsymbol{a}$
13:     else
14:         return $\boldsymbol{b}$
15:     end if
16: end if
```

Fig. 1. Binary Tournament

```
Input: $\mathcal{P}$, The population of $N$ individuals
Output: $\mathcal{F}$, Ranked and sorted $N$ individuals
1: for each $\boldsymbol{p} \in \mathcal{P}$ do
2:     for each $\boldsymbol{q} \in \mathcal{P}$ do
3:         $\mathcal{P}$ = samplesOf($\boldsymbol{p}$)
4:         $\mathcal{Q}$ = samplesOf($\boldsymbol{q}$)
5:         $r$ = uDominance($\mathcal{P}$, $\mathcal{Q}$, $\alpha$)
6:         if $r = 1$ then
7:             $\mathcal{S}_p = \mathcal{S}_p \cup \{\boldsymbol{p}\}$
8:         else if $r = -1$ then
9:             $n_p = n_p + 1$
10:        end if
11:    end for
12: end for
13: for each $\boldsymbol{p} \in \mathcal{P}$ do
14:     clearDominanceRelationLoop($\boldsymbol{p}$)
15: end for
16: for each $\boldsymbol{p} \in \mathcal{P}$ do
17:     if $n_p = 0$ then
18:         $\mathcal{F}_1 = \mathcal{F}_1 \cup \{\boldsymbol{p}\}$
19:     end if
20: end for
21: $i = 1$
22: while $\mathcal{F}_i \neq \varnothing$ do
23:     $\mathcal{H} = \varnothing$
24:     for each $\boldsymbol{p} \in \mathcal{F}_i$ do
25:         for each $\boldsymbol{q} \in S_p$ do
26:             $n_q = n_q - 1$
27:             if $n_q = 0$ then
28:                 $\mathcal{H} = \mathcal{H} \cup \{\boldsymbol{q}\}$
29:             end if
30:         end for
31:     end for
32:     $i = i + 1$
33:     $\mathcal{F}_i = \mathcal{H}$
34: end while
35: return $\mathcal{F}$
```

Fig. 2. U-dominance Ranking

- NSGA-II: The original NSGA-II. It takes only one sample and uses its objective values in the default Pareto dominance operator. It never considers noise in its dominance operator.
- NSGA-II-Median: takes multiple samples, obtains median values in different objectives and use them in NSGA-II's default dominance operator.
- NSGA-II-Mean: takes multiple samples, obtains mean values in different objectives and use them in NSGA-II's default dominance operator.
- NSGA-II-N: replaces NSGA-II's default dominance operator with a noise-aware dominance operator proposed in [8]. This noise-aware operator assumes Gaussian noise in objective functions in advance (c.f. Section 2).
- NSGA-II-IQR: replaces NSGA-II's default dominance operator with a noise-aware dominance operator proposed in [16] (c.f. Section 2).

All NSGA-II variants are evaluated with ZDT and DTLZ family problems (10 problems in total) [5,19]. Experiments were configured as shown in Table 1 and conducted with jMetal [7]. The total number of generations in each experiment is 200 in ZDT problems, 500 in DTLZ3 and 250 in the other DTLZ problems. Every experimental result is obtained from 20 independent experiments.

Table 1. Experimental Configurations

Parameter	Value	Parameter	Value
Confidence level α	0.55	Population size N	100
# of samples per individual	20	Crossover rate	0.9
γ (Eq. 8)	3	Mutation rate	1/(# of decision variables)
Noise strength β (Eq. 6–9)	0.1 or 0.5	Total # of generations	200, 250 or 500

This paper uses the hypervolume ratio (HVR) metric to compare NSGA-II variants and evaluate the U-dominance operator. HVR is calculated as the ratio of the hypervolume (HV) of non-dominated individuals (D) to the hypervolume of Pareto-optimal solutions (P^*) [15].

$$HVR(D) = \frac{HV(D)}{HV(P^*)} \tag{5}$$

HV measures the union of the volumes that non-dominated individuals dominate. Thus, HVR quantifies the optimality and diversity of non-dominated individuals D. A higher HVR indicates that non-dominated individuals are closer to the Pareto-optimal front and more diverse in the objective space.

In order to turn ZDT and DTLZ family problems to be noisy problems, this paper defines four kinds of additive noise in objective functions (c.f. Eq. 1).

- **Gaussian noise:** This noise distribution is characterized with a symmetric shape and a very limited number of outliners.

$$\epsilon_i = \beta\mathcal{N}(0, 1) \tag{6}$$

  $\mathcal{N}(0, 1)$ is the standard normal (or Gaussian) distribution.
- **Cauchy noise:** This noise distribution is used to generate more outliers than the Gaussian distribution does.

$$\epsilon_i = \beta\frac{\mathcal{N}(0, 1)}{\mathcal{N}(0, 1) + e} \tag{7}$$

  e is set to be a very small value in order to prevent division by zero.
- **Chi-squared noise:** This distribution is asymmetric and heavy-tailed in contrast to Gaussian and Cauchy distributions. It contains outliers.

$$\epsilon_i = \beta\sum_{i=1}^{\gamma}\mathcal{N}_i(0, 1)^2 \tag{8}$$

- **Log-normal noise:** This distribution is characterized with an asymmetric and heavy-tailed shape and outliers.

$$\epsilon_i = \beta \times exp(\mathcal{N}(0, 1)) \tag{9}$$

4.1 Experimental Results

Tables 2 to 5 show the average HVR values that six EMOAs yield at the last generation in 10 different test problems with different noise distributions. In each table, a number in parentheses indicates a standard deviation among different experiments. A bold number indicates the best result among six algorithms on a per-row basis. A double star (**) or a single star (*) is placed for an average HVR result when the result is significantly different from NSGA-II-U's result based on a single-tail *t*-test. A double star is placed with the confidence level of 99% while a single star is placed with the confidence level of 95%.

Table 2 shows the experimental results under Gaussian noise. NSGA-II-U clearly outperforms NSGA-II, NSGAII-Median and NSGA-II-IQR in all problems except ZDT2. NSGA-II-Mean and NSGA-II-N are more competitive against NSGA-II-U because noise follows a normal distribution and the distribution is symmetric. NSGA-II-U significantly outperforms NSGA-II-Mean and NSGA-II-N in DTLZ1 and DTLZ3 with the confidence level of 99% while the three EMOAs perform similarly in the other problems.

Table 3 shows the results under Cauchy noise. NSGA-II-U clearly outperforms NSGA-II, NSGA-II-Mean, NSGA-II-N and NSGA-II-IQR. In contrast to Table 2, NSGA-II-Median outperforms NSGA-II-Mean because Cauchy noise contains a lot of outliers. NSGA-II-U significantly outperforms NSGA-II-Median in DTLZ1 and DTLZ3 with the confidence level of 99% while NSGA-II-Median yields similar or better performance than NSGA-II-U in the other problems.

Under chi-squared noise (Table 4) and lognormal noise (Table 5), NSGA-II-U significantly outperforms five other EMOAs in almost all problems except ZDT2. In ZDT2, NSGA-II-Median performs better than NSGA-II-U when noise strength is 0.5. However, there is no significant difference between the two algorithms when noise strength is 0.1. Tables 4 and 5 demonstrate that the *U*-dominance operator reliably operates when many outliers exist in objective value samples under asymmetric noise distributions.

Table 2. HVR Results under Gaussian Noise

	β	NSGA-II	NSGA-II-Median	NSGA-II-Mean	NSGA-II-N	NSGA-II-IQR	NSGA-II-U
ZDT1	0.1	0.732(0.050)**	0.922(0.007)**	0.931(0.007)	0.923(0.006)**	0(0)**	**0.932**(0.006)
	0.5	0.004(0.015)**	0.604(0.066)**	**0.690**(0.045)	0.652(0.050)	0(0)**	0.684(0.056)
ZDT2	0.1	0.086(0.177)	0.716(0.264)**	**0.813**(0.150)**	0.347(0.404)	0(0)**	0.256(0.386)
	0.5	0(0)	0.010(0.044)	**0.024**(0.059)	0(0)	0(0)	0(0)
ZDT3	0.1	0.814(0.045)**	0.948(0.006)	0.951(0.018)	0.945(0.022)	0.003(0.004)**	**0.952**(0.006)
	0.5	0.066(0.066)**	0.679(0.108)	**0.757**(0.081)	0.731(0.077)	0(0)**	0.716(0.087)
ZDT4	0.1	0.764(0.139)**	0.908(0.105)	**0.938**(0.016)	0.839(0.134)*	0(0)**	0.917(0.053)
	0.5	0.004(0.019)**	0.558(0.197)*	0.679(0.163)	0.672(0.191)	0(0)**	**0.685**(0.181)
ZDT6	0.1	0.233(0.077)**	0.698(0.028)**	0.721(0.031)	0.691(0.033)**	0(0)**	**0.732**(0.023)
	0.5	0(0)**	0.067(0.055)*	**0.181**(0.094)*	0.071(0.054)*	0(0)**	0.112(0.073)
DTLZ1	0.1	0.022(0.056)**	0.397(0.390)**	0.477(0.369)**	0.546(0.353)**	0(0)**	**0.895**(0.010)
	0.5	0(0)**	0(0)**	0.007(0.032)**	0.014(0.055)**	0(0)**	**0.519**(0.138)
DTLZ2	0.1	0.319(0.101)	0.756(0.0141)	0.775(0.011)	0.736(0.020)	0(0)**	**0.780**(0.014)
	0.5	0(0)	0.006(0.015)	0.070(0.091)**	**0.204**(0.132)**	0(0)	0.002(0.008)
DTLZ3	0.1	0.015(0.069)**	0.007(0.024)**	0.059(0.178)**	0(0)**	0(0)**	**0.725**(0.151)
	0.5	0(0)**	0.002(0.011)**	0.009(0.030)**	0(0)**	0(0)**	**0.417**(0.151)
DTLZ4	0.1	0.469(0.122)**	0.801(0.131)	**0.862**(0.010)	0.810(0.091)	0(0)**	0.819(0.134)
	0.5	0.046(0.058)**	0.354(0.106)	**0.407**(0.093)	0.474(0.088)**	0(0)**	0.373(0.137)
DTLZ7	0.1	0.234(0.050)**	0.733(0.037)**	0.764(0.032)	0.728(0.035)**	0(0)**	**0.770**(0.030)
	0.5	0(0)**	0.153(0.063)**	**0.229**(0.059)**	0.182(0.059)**	0(0)**	0.069(0.084)

Table 3. HVR Results under Cauchy Noise

	β	NSGA-II	NSGA-II-Median	NSGA-II-Mean	NSGA-II-N	NSGA-II-IQR	NSGA-II-U
ZDT1	0.1	0.682(0.054)**	**0.924**(0.005)**	0.700(0.040)**	0.719(0.039)**	0.0(0.001)**	0.917(0.009)
	0.5	0.160(0.086)**	**0.727**(0.031)**	0.174(0.068)**	0.195(0.085)**	0(0)**	0.676(0.043)
ZDT2	0.1	0.307(0.197)	**0.819**(0.193)**	0.214(0.228)*	0.039(0.122)**	0(0)**	0.459(0.426)
	0.5	0(0)	**0.300**(0.255)**	0(0)	0(0)	0(0)**	0.015(0.069)
ZDT3	0.1	0.675(0.054)**	**0.936**(0.009)*	0.647(0.062)**	0.629(0.060)**	0.007(0.017)**	0.929(0.007)
	0.5	0.041(0.025)**	**0.672**(0.045)**	0.042(0.030)**	0.035(0.038)**	0(0)**	0.569(0.094)
ZDT4	0.1	0.686(0.134)**	**0.931**(0.012)	0.670(0.152)**	0.658(0.199)**	0(0)**	0.907(0.061)
	0.5	0.245(0.188)**	0.645(0.146)**	0.226(0.158)**	0.230(0.203)**	0(0)**	**0.777**(0.091)
ZDT6	0.1	0.311(0.058)**	**0.740**(0.027)	0.308(0.070)**	0.349(0.046)**	0(0)**	0.736(0.025)
	0.5	0.001(0.005)**	**0.406**(0.048)**	0.002(0.006)**	0(0)**	0(0)**	0.300(0.068)
DTLZ1	0.1	0(0)**	0.444(0.358)**	0(0)**	0.364(0.156)**	0(0)**	**0.873**(0.019)
	0.5	0.011(0.050)**	0.102(0.174)**	0(0)**	0.048(0.114)**	0(0)**	**0.492**(0.120)
DTLZ2	0.1	0.026(0.049)**	**0.780**(0.012)	0.007(0.021)**	0.019(0.039)**	0(0)**	0.773(0.009)
	0.5	0.000(0.001)	**0.346**(0.074)**	0.003(0.011)	0.010(0.025)	0(0)**	0.014(0.031)
DTLZ3	0.1	0(0)**	0.027(0.110)**	0(0)**	0.006(0.027)**	0(0)**	**0.714**(0.056)
	0.5	0(0)**	0.004(0.020)**	0(0)**	0(0)**	0(0)**	**0.008**(0.038)
DTLZ4	0.1	0.370(0.097)**	**0.825**(0.135)	0.370(0.117)**	0.297(0.101)**	0(0)**	0.795(0.161)
	0.5	0.087(0.060)**	**0.496**(0.162)	0.092(0.075)**	0.014(0.030)**	0(0)**	0.416(0.134)
DTLZ7	0.1	0.490(0.068)**	**0.830**(0.019)	0.500(0.050)**	0.552(0.037)**	0(0)**	0.820(0.016)
	0.5	0.129(0.033)**	**0.636**(0.034)	0.123(0.056)**	0.037(0.045)**	0(0)**	0.612(0.044)

Table 4. HVR Results under Chi-squared Noise

	β	NSGA-II	NSGA-II-Median	NSGA-II-Mean	NSGA-II-N	NSGA-II-IQR	NSGA-II-U
ZDT1	0.1	0.689(0.045)**	0.893(0.009)**	0.898(0.012)**	0.897(0.009)**	0.001(0.002)**	**0.922**(0.010)
	0.5	0.173(0.059)**	0.532(0.066)**	0.587(0.053)**	0.615(0.061)**	0(0)**	**0.713**(0.037)
ZDT2	0.1	0.46(0.169)	**0.727**(0.256)	0.733(0.258)	0.279(0.374)	0(0)**	0.603(0.370)
	0.5	0(0)	**0.208**(0.085)**	0.120(0.149)	0(0)	0(0)	0.027(0.087)
ZDT3	0.1	0.675(0.059)**	0.915(0.010)**	0.920(0.008)**	0.914(0.009)**	0.002(0.003)**	**0.930**(0.005)
	0.5	0.071(0.091)	0.490(0.081)	0.521(0.085)	0.550(0.081)	0(0)**	**0.556**(0.066)
ZDT4	0.1	0.610(0.204)**	0.891(0.086)	0.884(0.081)	0.866(0.056)**	0(0)**	**0.926**(0.027)
	0.5	0.218(0.150)**	0.685(0.115)	0.569(0.205)	0.623(0.154)	0(0)**	**0.732**(0.156)
ZDT6	0.1	0.333(0.070)**	0.687(0.032)**	0.684(0.033)**	0.657(0.047)**	0(0)**	**0.740**(0.022)
	0.5	0(0)**	0.239(0.060)**	0.228(0.098)**	0.205(0.063)**	0(0)**	**0.337**(0.066)
DTLZ1	0.1	0(0)**	0.541(0.292)**	0.452(0.390)**	0.265(0.230)**	0(0)**	**0.878**(0.013)
	0.5	0(0)**	0(0)**	0.014(0.047)**	0.009(0.019)**	0(0)**	**0.513**(0.134)
DTLZ2	0.1	0.451(0.058)**	0.745(0.015)**	0.749(0.011)**	0.736(0.013)**	0(0)**	**0.787**(0.004)
	0.5	0.007(0.024)	0.002(0.004)	**0.012**(0.020)	0.026(0.024)	0(0)**	0.007(0.013)
DTLZ3	0.1	0(0)**	0(0)**	0(0)**	0(0)**	0(0)**	**0.742**(0.063)
	0.5	0(0)**	0(0)**	0(0)**	0(0)**	0(0)**	**0.011**(0.041)
DTLZ4	0.1	0.578(0.094)**	0.789(0.129)	0.840(0.012)**	0.827(0.014)**	0(0)**	**0.873**(0.011)
	0.5	0.120(0.093)**	0.285(0.099)**	0.323(0.091)**	0.326(0.079)**	0(0)**	**0.491**(0.100)
DTLZ7	0.1	0.658(0.036)**	0.798(0.013)**	0.787(0.018)**	0.782(0.026)**	0(0)**	**0.828**(0.008)
	0.5	0.138(0.042)**	0.535(0.051)**	0.553(0.053)*	0.558(0.045)*	0(0)**	**0.607**(0.037)

Table 5. HVR Results under Log-normal Noise

	β	NSGA-II	NSGA-II-Median	NSGA-II-Mean	NSGA-II-N	NSGA-II-IQR	NSGA-II-U
ZDT1	0.1	0.822(0.019)**	0.937(0.007)**	0.917(0.006)**	0.917(0.009)**	0.048(0.04)**	**0.951**(0.005)
	0.5	0.323(0.103)**	0.791(0.036)**	0.667(0.052)**	0.683(0.057)**	0.0(0.0)**	**0.857**(0.013)
ZDT2	0.1	0.689(0.164)	**0.882**(0.011)	0.776(0.195)	0.428(0.394)**	0.0(0.0)**	0.753(0.321)
	0.5	0.046(0.074)**	**0.561**(0.196)**	0.278(0.206)	0.043(0.134)**	0.0(0.0)**	0.311(0.353)
ZDT3	0.1	0.873(0.019)**	0.949(0.008)	0.934(0.007)**	0.931(0.008)**	0.086(0.056)**	**0.954**(0.016)
	0.5	0.27(0.118)**	0.806(0.032)**	0.626(0.068)**	0.63(0.094)**	0.0(0.0)**	**0.874**(0.014)
ZDT4	0.1	0.739(0.145)**	0.905(0.069)*	0.898(0.067)*	0.86(0.096)**	0.0(0.0)**	**0.940**(0.016)
	0.5	0.178(0.15)**	0.808(0.089)	0.673(0.199)*	0.596(0.194)**	0.0(0.0)**	**0.816**(0.130)
ZDT6	0.1	0.531(0.052)**	0.784(0.029)**	0.709(0.024)**	0.694(0.026)**	0.0(0.0)**	**0.808**(0.020)
	0.5	0.062(0.034)**	0.471(0.059)**	0.301(0.075)**	0.315(0.062)**	0.0(0.0)**	**0.589**(0.032)
DTLZ1	0.1	0.036(0.158)**	0.561(0.334)**	0.426(0.373)**	0.556(0.265)**	0.0(0.0)**	**0.906**(0.013)
	0.5	0.0(0.0)**	0.146(0.223)**	0.063(0.129)**	0.429(0.167)**	0.0(0.0)**	**0.795**(0.027)
DTLZ2	0.1	0.675(0.016)**	0.779(0.014)**	0.781(0.007)**	0.747(0.013)**	0.0(0.0)**	**0.792**(0.011)
	0.5	0.004(0.012)**	0.589(0.045)**	0.27(0.101)**	0.313(0.112)**	0.0(0.0)**	**0.676**(0.024)
DTLZ3	0.1	0.0(0.0)**	0.0(0.0)**	0.0(0.0)**	0.0(0.0)**	0.0(0.0)**	**0.763**(0.035)
	0.5	0.0(0.0)**	0.008(0.036)**	0.0(0.0)**	0.0(0.0)**	0.0(0.0)**	**0.547**(0.107)
DTLZ4	0.1	0.657(0.178)**	**0.844**(0.094)	0.802(0.161)	0.71(0.192)	0.0(0.0)**	0.816(0.156)
	0.5	0.255(0.094)**	0.653(0.138)	0.516(0.089)**	0.509(0.109)**	0.0(0.0)**	**0.703**(0.206)
DTLZ7	0.1	0.763(0.014)**	**0.840**(0.012)	0.83(0.015)	0.799(0.024)**	0.001(0.003)**	**0.840**(0.030)
	0.5	0.341(0.065)**	0.690(0.030)**	0.631(0.033)**	0.62(0.034)**	0.0(0.0)**	**0.742**(0.028)

5 Conclusions

This paper proposes and evaluates a new noise-aware dominance operator, called the U-dominance operator, which never assumes noise distributions in advance by leveraging the Mann-Whitney U-test. Experimental results show that it operates reliably in noisy MOPs and outperforms existing noise-aware dominance operators particularly when many outliers exist in objective value samples under asymmetric noise distributions.

References

1. Babbar, M., Lakshmikantha, A., Goldberg, D.: A modified NSGA-II to solve noisy multiobjective problems. In: Proc. ACM Genet. Evol. Computat. Conf. (2003)
2. Bianchi, L., Dorigo, M., Gambardella, L., Gutjahr, W.J.: A survey on metaheuristics for stochastic combinatorial optimization. Nat. Comput. 8(2) (2009)
3. Boonma, P., Suzuki, J.: A confidence-based dominance operator in evolutionary algorithms for noisy multiobjective optimization problems. In: Proc. IEEE Int'l Conference on Tools with Artificial Intelligence (2009)
4. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
5. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Abraham, A., Jain, R., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization. Springer (2005)
6. Delibrasis, K., Undrill, P., Cameron, G.: Genetic algorithm implementation of stack filter design for image restoration. In: Proc. Vis., Image, Sign. Proc. (1996)

7. Durillo, J., Nebro, A., Alba, E.: The jMetal framework for multi-objective optimization: Design and architecture. In: Proc. IEEE Congress on Evol. Computat. (2010)
8. Eskandari, H., Geiger, C., Bird, R.: Handling uncertainty in evolutionary multiobjective optimization: SPGA. In: Proc. IEEE Congress Evol. Computat. (2007)
9. Goh, C.K., Tan, K.C.: Noise handling in evolutionary multi-objective optimization. In: Proc. of IEEE Congress on Evolutionary Computation (2006)
10. Hughes, E.: Evolutionary multi-objective ranking with uncertainty and noise. In: Proc. Int'l Conf. on Evolutionary Multi-Criterion Optimization (2001)
11. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. IEEE Trans. Evol. Computat. 9(3) (2005)
12. Mann, H., Whitney, D.: On a test of whether one of two random variables is stochastically larger than the other. Annals of Math. Stat. 18(1) (1947)
13. Park, T., Ryu, K.: Accumulative sampling for noisy evolutionary multi-objective optimization. In: Proc. of ACM Genetic and Evol. Computat. Conference (2011)
14. Teich, J.: Pareto-front exploration with uncertain objectives. In: Proc. of Int'l Conf. on Evol. Multi-Criterion Optimization (2001)
15. Veldhuizen, D.A.V., Lamont, G.B.: Multiobjective evolutionary algorithm test suites. In: Proc. ACM Symposium on Applied Computing (1999)
16. Voß, T., Trautmann, H., Igel, C.: New Uncertainty Handling Strategies in Multiobjective Evolutionary Optimization. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 260–269. Springer, Heidelberg (2010)
17. Wormington, M., Panaccione, C., Matney, K.M., Bowen, D.K.: Characterization of structures from x-ray scattering data using genetic algorithms. Phil. Trans. R. Soc. Lond. A 357(1761) (1999)
18. Zhu, B., Suzuki, J., Boonma, P.: Solving the probabilistic traveling salesperson problem with profits (pTSPP) with a noise-aware evolutionary multiobjective optimization algorithm. In: Proc. IEEE Congress on Evol. Computat. (2011)
19. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evol. Computat. 8(2) (2000)

The Emergence of New Genes in EcoSim and Its Effect on Fitness

Marwa Khater, Elham Salehi, and Robin Gras

School of Computer Science, University of Windsor
ON, Canada
{khater,salehie,rgras}@uwindsor.ca

Abstract. The emergence of complex adaptive traits and behaviors in artificial life systems requires long term evolution with continuous emergence governed by natural selection. We model organism's genomes in an individual-based evolutionary ecosystem simulation (EcoSim), with fuzzy cognitive maps (FCM) representing their behavioral traits. Our system allows for the emergence of new traits and disappearing of others, throughout a course of evolution. We show how EcoSim models evolution through the behavioral model of its individuals governed by natural selection. We validate our model by examining the effect, the emergence of new genes, has on individual's fitness. Machine learning tools showed great interest lately in modern biology, evolutionary genetics and bioinformatics domains. We use Random Forest classifier, which has been widely used lately due to its power of dealing with large number of attributes with high efficiency, to predict fitness value knowing only the values of new genes. Furthermore discovering meaningful rules behind the fitness prediction encouraged us to use a pre processing step of feature selection. The selected features were then used to deduce important rules using the JRip learner algorithm.

Keywords: artificial life modeling, individual-based modeling, evolution, fitness.

1 Introduction

Charles Darwin's theory of adaptation through natural selection came to be widely seen as the primary explanation of the process of evolution and forms the basis of modern evolutionary theory. Darwin's principle of natural selection relies on a number of propositions. The individuals in a population are not identical but vary in certain traits. This variation, at least partly, is heritable. Individuals vary in the number and the quality of their descendants which depends critically on the interactions of the individual's trait and its environment. Populations with these characteristics, over generations, become more adapted to their environment. The key to adaptation by natural selection is the effect of a multitude of small but cumulative changes. But while the heritable variation causing these changes might be random, most of those that are preserved do not damage the fitness of the individuals. These variations have turned out to be somehow beneficial to the reproductive success of their carrier. From the genetic perspective, mutations and natural selection, through the course of evolution enforce the emergence of new traits and disappearing of others.

Darwinian evolution governed by natural selection is modeled in EcoSim, an evolutionary predator - prey ecosystem simulation. In this paper the evolutionary machinery in EcoSim was studied by examining the emergence of new genes and their effect on fitness. Random Forest (RF) [2] was used to build a classifier that was able to predict the values of fitness based on the values of new developed genes. This is considered to be a validation step to ensure the validity of the behavior model and its ability to cope with changes in the environment. A feature selection step is then presented along with rule learning using JRip learner [4]. These rules allow us to discover the most important features that increase fitness, and help us to understand the semantics of the behavior model. The rest of the paper is organized as follows: A brief description of our model is presented in Section 2. Section 3 depicts the details of emergence of new genes. Building a Random Forest classifier for inference is illustrated in Section 4. Furthermore, the JRip rule learner along with the two feature selection steps are presented in Section 5, followed by a summed up conclusion in Section 6.

2 EcoSim

In order to investigate several open theoretical ecology questions we have designed the individual-based evolving predator-prey ecosystem simulation platform EcoSim [5] [6] [1]. Our objective is to study how individual and local events can affect high level mechanisms such as community formation, speciation or evolution. EcoSim uses a fuzzy cognitive map (FCM) as a behavior model [7] which allows a combination of compactness with a very low computational requirement while having the capacity to represent complex high level notions. The complex adaptive agents (or individuals) of this simulation are either prey or predators which act in a dynamic 2D environment of 1000 x 1000 cells. Each cell may contain several individuals and some amount of food from which individuals gain energy. Each individual possesses several physical characteristics including age, minimum age for breeding, speed, vision distance, levels of energy, and the amount of energy transmitted to the offspring. Preys consume grass, whereas predators predate on prey individuals. Grass distribution is dynamic, as it diffuses throughout the world and disappears when consumed by prey. An individual consumes some energy each time it performs an action such as evasion, search for food, eating or breeding. Each individual performs one action during a time step based on its perception of the environment.

FCM [5] is used to model the individual's behavior and to compute the next action to be performed. The individual's FCM is analogous to a genome and therefore can be subjected to evolution. Each agent possesses a unique proper FCM, and the system can still manage several hundreds of thousands of such

[1] `http://sites.google.com/site/ecosimgroup/research/ecosystem-simulation`

agents simultaneously into the world with reasonable computational requirements. A typical run lasts dozens of thousands of time steps during which more than a billion agents are born and several thousands of species are generated, allowing evolutionary process to take place and new behaviors to emerge in a constantly changing environment. The simulation operationalizes each 'species' as a set of individuals sharing similar genomes [8]. Every member of a species has a genome that is within a threshold genetic distance away from the average species genome - an average of the FCMs of all the species' members. For computational consideration, species are also associated with a 'genome' representing the average genome of all their population of individuals. As species evolve due to birth and death of their individuals, the changes they go through are mapped to their FCMs. The genomic distance between individuals, the average species genome. and genetic distance threshold are used to decide on the creation of a new species. Formally a FCM is a graph which contains a set of nodes C, each node C_i being a concept, and a set of edges I, each edge I_{ij} representing the influence of the concept C_i on the concept C_j see Figure 1. We use a FCM to model an agent's behavior (i.e. the structure of the graph) and to compute the next action of the agent (i.e. dynamic of the map). In each FCM, three kinds of concepts are defined: sensitive (such as distance to foe or food, amount of energy, etc), internal (fear, hunger, curiosity, satisfaction, etc) and motor (evasion, socialization, exploration, breeding, etc). The genome has a maximum of 390

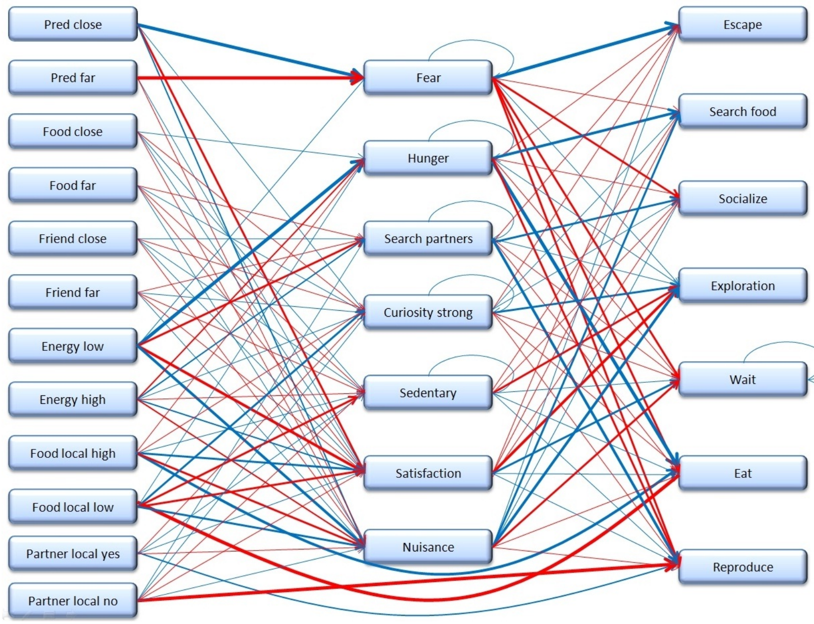


Fig. 1. The initial map of a prey. Blue edges represent positive influence and red edges negative influence of one concept on another. The width of the edge represents the strength of the influence.

sites, where each site corresponds to a possible edge between two concepts of the FCM. A breeding event occurs when two individuals in the same cell choose the reproduction action at the same time step, when their genome dissimilarity is below a threshold (i.e. half of the threshold for speciation) and when they have enough energy to reproduce. During this breeding event the FCMs of the two parents are combined and transmitted to their offspring with the possible addition of some random mutations. The behavioral model of each individual is therefore unique.

3 Emergence of New Genes

The FCM of each individual plays the role of its genome and has a maximum size of 390 sites. Every site is a real number which measures the level of influence from one concept to another. Initially all prey and predator individuals are given the same value for their genomes respectively see Figure 1. This initial map has 125 existing edges between concepts. These values were carefully chosen and tested. Furthermore, a new offspring carries a combination of its parent's two genomes, along with some possible mutations. Step after step as more individuals are created changes in the FCM occur due to the formation of new edges (with probability of 0.001), removal of existing edges (with probability of 0.0005) and changes in the weights associate to existing edges (with probability of 0.005). New genes may emerge from among the 265 possible new edges. Furthermore, we calculate the average fitness for every species as the average fitness of its individuals. We define fitness of an individual as the age of death of the individual plus the sum of the age of death of its direct offspring. Accordingly, the fitness value mirrors the individual's capability to survive longer and produce high number of strong adaptive offspring.

The values of the genome determine how the organism behaves in its current environment. Thus, this information determines the capability of the organism to survive, reproduce and transmit its genome. The environment changes from one place to another and from a time step to another. Furthermore, as we model a predator-prey system, we have co-evolution; the strategies (behavior) of each individual (predator/prey) are continuously changing as they try to adapt to the other kind. In a constantly changing environmnet individuals must continuously learn. This fact drives the individulas evolve survival strategies that helps them adapt to their changing environmnet. Prey individuals die due to several reasons: reaching maximum age, lack of energy or being eaten by predators. We do not force natural selection by limiting the number of existing species or fixing population size, but rather selection acts through the behavioral model. Individual that are not able to gain energy from food, reproduce and escape from predators will be eliminated by the evolutionary process. Thus, fitness levels are not fixed and do not always increase; rather, they vary over time. The evolutionary process of EcoSim governs the emergence of new genes and disappearing of others. By the process of natural selection only the fittest will be able to survive, and therefore the emergence of new genes is not random but adaptive,adding to the intelligence and complexity of the individuals.

4 Building a Random Forest Classifier for Inference

Recent work in computational biology has shown an increased use of Random Forest[2], owing to its unique advantages in dealing with small sample size, high-dimensional feature space, and complex data structures [1][3]. Random forests have higher prediction accuracy as compared to individual classification trees because the ensemble adjusts for the instability of individual trees induced by small changes in the learning sample, which would otherwise impair the prediction accuracy in test samples. A new gene is a gene that had the value of zero in the initial FCM map of prey, and then mutates to and preserves a non-zero value later on in the simulation. The genes that were initially zero and then changed are monitored and extracted. We calculate the average FCM for every existing species in every time step in the simulation. From 390 possible genes there are 125 initial genes and 265 possible new genes, which can appear gradually. Each of these 265 new genes represents an attribute in the classifier, making the feature space a high dimensional feature space. Each instance of the learning process consists of the set of 265 average gene values in a given species at a given time step. We have 4 different runs of the simulation, each having around 20000 time steps. We neglect the first couple of thousand of time steps in our calculations to overcome any misleading results due to the initial similarity between individual genomes. Consequently, all genes would have obtained non-zero values. We extract randomly around 150000 instances from every run, to build 4 classifiers, one for each run. The class variable to predict at a given time step is the average fitness value of the species 50 time steps later. The effect of these new genes on the individual fitness is not immediate so used a 50 time steps shift to give time for the values of new genes to affect the fitness. We have three classes for the average fitness; LOW which represent values less than 85. HIGH, which is between 85 and 110, and very high VHIGH, for values higher than 110. Table1 presents the percentage of instances for every class in 4 different runs of the simulation.

The Random Forest classifier implemented in the weka environment [10] was used. Instances for every run are split into two sets: train and test. Using 10 fold cross validation and 10 classification trees, 94.7% average train accuracy of 4 runs with a standard deviation (std) of 0.33 and 95% test accuracy with std of 0.3 was found. Although there are many factors affecting the fitness, it

Table 1. Percentage of low fitness (LOW), high fitness (HIGH), and very high fitness (VHIGH) prey instances for 4 different runs

Run	Percentage LOW	Percentage HIGH	Percentage VHIGH
Run 1	53%	36%	10%
Run 2	45%	39.5%	15.5%
Run 3	41%	42%	17%
Run 4	40%	46%	14%

was still predicted with high accuracy knowing only the values of the newly developed genes. This high accuracy supports our assumption that the values of the new evolved genes could affect the well being of the individuals. We also tested the generality of our finding by training the classifier on one data set from one run and testing it on a data set from another run of the simulation. Although the classifier was able to learn some general rules for prediction in different runs, the average accuracy of 43.5% with std of 1.2 was not very high. One reason is that the simulation varies, from one run to another and each run has unique conditions in which survival strategies of the individuals vary. This leads to different behaviors, and thus different values of the genes that affect the fitness. These values did not evolve randomly, but were preserved by the evolutionary process thereby adding a higher level of complexity and intelligence to the individuals. It is worth noting that this increase in complexity adds an extra cost for the individual as any new gene added to this individual increase the amount of energy the individual needs to consume at any time step. Therefore, in order for new genes to be beneficial in terms of fitness, the benefit in term of advantageous behavior should be higher than the cost in energy. This supports the validity of our behavior model and demonstrates how evolutionary processes lead to adaptation and intelligence. This finding also emphasizes the role of natural selection in our simulation. Genomes that participate in the well being of its carrier host persist and survive.

5 Rule Learning Using JRip

We are interested in a better understanding of the semantics behind the evolution of the new genes. Which genes have a stronger influence on fitness and with which values? In order to study this phenomenon we extracted rules from the learned model predicting the fitness. The interpretability of a random forest is not as straightforward as that of an individual classification tree, where the influence of a predictor variable directly corresponds to its position in the tree. The model generated by the Random Forest can be challenging to interpret. To by-pass this limitation we use the JRip rule learner [4] to extract more semantics from the prediction model. JRip implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which is an optimized version of IREP. JRip learn rules that are easy to understand and provide informative feedback about the problem. In order to improve the model performance and gain a deeper insight into the underlying processes affecting the results, we used a feature selection step. This pre-processing step highlights the most important genes affecting the fitness and eases the process of rule interpretation in addition to minimizing the number of rules. We use two different feature selection methods, CfsSubsetEval [11] using Best First searcher, and CMSS-EDA [12] and present both their results.

The first feature selection method used CfsSubsetEval evaluates the worth of a subset of features by considering the individual predictive ability of each feature along with the degree of redundancy between them. BestFirst searches the space of feature subsets with a greedy hill-climber augmented with a backtracking facility. This was implemented under weka environment. The other method, which was previously presented by Salehi E, is a wrapper feature selection methods [13] [14] based on an estimation of distribution algorithm (EDA) called CMSS-EDA [12]. Since CMSS-EDA does not consider a small fix upper bound on the number of variables on which each variable depends, the most relevant variables using this approach were found even when there were many dependencies between them. Each subset of variables is encoded as a bit-string and the subset of variables which maximizes the AUC (Area Under ROC Curve) obtained by a Bayesian network classifier was found.

Table 2. Accuracy percentages for training and testing with JRip after CMSS-EDA and CfsSubsetEval feature selection, for 4 runs of the simulation

	CMSS-EDA				CfsSubsetEval			
	Train Acc.	Test Acc.	Selected Features	No. Rules	Train Acc.	Test Acc.	Selected Features	No. Rules
Run 1	69.5%	70.9%	41	78	69.3%	70.5%	28	53
Run 2	72.2%	74.4%	35	119	73%	74.7%	41	101
Run 3	71.3%	71.5%	63	109	70.3%	71.6%	41	113
Run 4	73.7%	76.8%	47	62	73.8%	74.3%	38	55

First, samples from the data set were used to extract features using both techniques. Then only these features were used with JRip rule extractor using separate training and testing sets with 10 fold cross validation. Different features have been selected from different runs of the simulation. This is due to the complexity of the behavior model. Survival techniques vary based on the different circumstances of each run and environment dynamics. Although the two feature selection techniques selected different features, their prediction accuracy using JRip was very similar. This shows the strong dependencies among the genes and how they collaborate with each other to adapt to their dynamic environment. Also some genes might have redundant information which could be replaced with some other set of genes. Table2 shows results of JRip rule learner along with the number of rules it produces for each different run of the simulation for both CMSS-EDA and CfsSubsetEval respectively. Different IF THEN rules are learned from JRip to predict the three fitness classes. We were mainly interested in the rules that predict the very high fitness class, VHIGH because understanding the conditions that increase the individual's species fitness is highly informative about the simulation properties. The number of rules, that predict VHIGH ranges from 2 to 25, in all runs using both feature selection methods. It should be noted that each gene can have a real positive value(for a new edge from one concept that positively influence another) or a real negative value (for a new edge from one concept that negatively influence another). Due to space limitations we present rules with highest ranking and highest weight of instances per rule. Rules are in the form of "IF edges emerge between the following concepts THEN the average fitness of the species will be very high after 50 time steps".

– IF Satisfaction decreases sedentary, AND escape decreases socialize, AND search for partner decreases nuisance, AND food local low increases fear, AND satisfaction increases satisfaction, THEN fitness is VHIGH.

Explanation for the previous rule is as follows. Satisfaction is an internal concept which is initially decreased by a low local (same cell as the individual) food level, or by a low energy level of the individual, or by a predator being detected in a close range (see Figure 1 for the initial prey map). A new edge corresponding to 'satisfaction decreasing sedentary' has evolved. The desire to escape lowers the desire to socialize because preference should be given to escaping from a close by predator. Searching for a partner decreases the nuisance, and having a low local food levels increases the internal concept of fear. 'Satisfaction increases satisfaction' is an internal loop which gives persistence to the sensation of satisfaction. The combination of these new emerging genes within population tends to increase the population's fitness.

– IF curiosity decreases sedentary, AND friend close decreases exploration, AND energy high decreases exploration, AND food far increases reproduction, AND energy high increases wait, AND explore decreases explore, THEN fitness is VHIGH.

Curiosity and sedentary are both internal concepts. Initially curiosity increases exploration which encourages the individual to move. Sedentary, in the initial prey map, decreases exploration. A new edge was developed between these two internal concepts, meaning that curiosity decreases sedentary, which enforce the initial semantic. Having friends close-by decreases the desire to explore and to move because having close-by friends encourages the individual to search for a partner. Having a high level of energy decreases exploration because it might be better to reproduce and search for a partner. Having no close-by food, increases the will to reproduce instead of wasting energy by searching for distant food. Having high levels of energy increases the wait action is less obvious to interpret, but can mean that the individual has no need to move. Finally, exploration decreases the internal desire for further exploration, is also an internal loop that reduce the persistence of exploration, which is mostly a random movement. These factors also increase the fitness.

– IF no local partner decreases fear, AND food local high increases wait, AND food far increases exploration, AND friend close increases eat, AND partner local yes decreases search partner, AND reproduce decreases socialize, THEN fitness is VHIGH.

Fear is an internal concept that initially decreases all motion actions except escape and explore. A negative edge has been established between the perception of having no local partner and fear. If high levels of local food have been found, waiting is increased. If the food is far the need to explore increases as well.

Having close-by friends increases the desire to eat and gain more energy. Local close by partners detected decreases the need of searching for partners. The desire of reproduction lowers also the socialization concept. The combination of the emergence of these factors also increases the fitness.

– IF no local partner decreases fear, AND predator close increases escape, AND socialize decreases reproduce, AND predator far increases eat, THEN fitness is VHIGH.

Initially detecting close-by predators increases the internal concept of fear and an increased level of fear also increases the desire of escaping. A new edge emerged which directly encourages escape action if predators are detected without using internal concept. The need to socialize decreases the need to reproduce, which means that when there are no local partners (need to socialize), there is no interest to try to breed. Also a meaningful edge between increasing the eat action if predators are far was developed all leading to increasing fitness. The logical soundness of most of the produced rules shows both high semantics in the initial behavioral model of prey individuals, and the self organizing capability of our system. Some rules seems less obvious to interpret, but as the global model is a highly complex non-linear system with feedback loops, some modifications can have effects on other parts of the system and are therefore difficult to understand. The behavior model (genome) was able to evolve without any external interference besides the natural selection forces. These rules emphasize the importance of certain new genes and the strong dependencies found among them. The logical correctness of some of these rules is also a major interesting discovery.

6 Conclusion

The correlation between the fitness of a population and the emergence of new behaviors was studied. We analyzed the emergence of new genes which affect the behavioral model of the agents, through the evolutionary process in EcoSim. The value of these new genes served as attributes for fitness prediction using Random Forest classifier implemented in weka. The high accuracy obtained from the Random Forest classifier shows the capacity of our behavior model to capture relevant information from its environment giving to the successful individual's ability to survive and to adapt to its dynamic environment. In a second step, rule learning technique was applied to extract semantics information from the prediction model. This enabled better understanding of the logical rules in the new evolved behavioral models that led to an increase in fitness. The JRip rule learner was used after a pre-processing feature selection step. The soundness of the rules obtained is very encouraging as they help us to understand what kind of new behaviors can be useful in such dynamic and competitive ecosystem. The next step of our study will be to correlate, with the help of biologists, the semantic rules we obtained with data coming from real natural ecosystems. As this simulation allows very long runs, it will be possible to study the dynamics of co-evolution by analyzing successive changes in the behavioral models during periods of prey adaptation to predators and vise versa.

Acknowledgements. This work is supported by the NSERC grant ORGPIN 341854, the CRC grant 950-2-3617 and the CFI grant 203617 and is made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET:www.sharcnet.ca).

References

1. Cutler, D.R., Edwards Jr., T.C., Beard, K.H., Cutler, A., Hess, K., Gibson, J., Lawler, J.: Random forests for classification in ecology. Ecology 88, 2783–2792 (2007)
2. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
3. Diaz-Uriarte, R., de Andres, S.A.: Gene selection and classification of microarray data using random forest. BMC Bioinformatics 7(1), 3 (2006)
4. Cohen, W.: Fast effective rule induction. In: 12th International Conference on Machine Learning, pp. 115–123 (1995)
5. Gras, R., Devaurs, D., Wozniak, A., Aspinall, A.: An individual-based evolving predator-prey ecosystem simulation using fuzzy cognitive map as behavior model. Artificial Life 15(4), 423–463 (2009)
6. Gras, R., Golestani, A., Hosseini, M., Khater, M., Farahani, Y.M., Mashayekhi, M., Ibne, S.M., Sajadi, A., Salehi, E., Scott, R.: Ecosim: an individual-based platform for studying evolution. In: European Conference on Artificial Life, pp. 284–286 (2011)
7. Kosko, B.: Fuzzy cognitive maps. Int. Jornal of Man-Machine Studies, 65–75 (1986)
8. Aspinall, A., Gras, R.: K-Means Clustering as a Speciation Mechanism within an Individual-Based Evolving Predator-Prey Ecosystem Simulation. In: An, A., Lingras, P., Petty, S., Huang, R. (eds.) AMT 2010. LNCS, vol. 6335, pp. 318–329. Springer, Heidelberg (2010)
9. Qi, Y., Bar-Joseph, Z., Klein-Seetharman, J.: Evaluation of different biological data and computational classification methods for use in protein interaction predection. Proteins 63(3), 490–500 (2006)
10. Witten, I., Frank, E.: Data Mining- Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, USA (2000)
11. Hall, M.A.: Correlation-based Feature Subset Selection for Machine Learning. PhD thesis, University of Waikato, Hamilton, New Zealand (1998)
12. Salehi, E., Gras, R.: Efficient eda for large optimization problem via constraining the search space of models. In: GECCO 2011, pp. 73–74. ACM (2011)
13. Yang, Q., Salehi, E., Gras, R.: Using Feature Selection Approaches to Find the Dependent Features. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010, Part I. LNCS (LNAI), vol. 6113, pp. 487–494. Springer, Heidelberg (2010)
14. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics 23(19), 2507–2517 (2007)

Mass-Dispersed Gravitational Search Algorithm for Gene Regulatory Network Model Parameter Identification

Mohsen Davarynejad[1], Zary Forghany[2], and Jan van den Berg[1]

[1] Faculty of Technology, Policy and Management, Delft University of Technology, The Netherlands
{m.davarynejad,j.vandenberg}@tudelft.nl
[2] Department of Molecular Cell Biology, Leiden University Medical Center (LUMC), The Netherlands
z.forghany@lumc.nl

Abstract. The interaction mechanisms at the molecular level that govern essential processes inside the cell are conventionally modeled by nonlinear dynamic systems of coupled differential equations. Our implementation adopts an S-system to capture the dynamics of the gene regulatory network (GRN) of interest. To identify a solution to inverse problem of GRN parameter identification the gravitational search algorithm (GSA) is adopted here. Contributions made in the present paper are twofold. Firstly the bias of GSA toward the center of the search space is reported. Secondly motivated by observed center-seeking (CS) bias of GSA, mass-dispersed gravitational search algorithm (mdGSA) is proposed here. Simulation results on a set of well-studied mathematical benchmark problems and two gene regulatory networks confirms that the proposed mdGSA is superior to the standard GSA, mainly duo to its reduced CS bias.

Keywords: Gravitational search algorithm, Center-seeking bias, Mass-dispersed gravitational search algorithm, Gene regulatory network model identification.

1 Introduction

Many diseases are the result of polygenic and pleiotropic effects controlled by multiple genes. Genome-wide interaction analysis (GWIA), by providing insight into the biological and biochemical pathways of disease is a natural evolution of single locus study.

The level of activation and inhibition of genes are governed by factors within a cellular environment and outside of the cell. The activation and inhibition relationship between genes are integrated by gene regulatory networks (GRNs), forming an organizational level in the cell with complex dynamics [4]. Mathematical modeling of GRNs provides a powerful tool not only to better understand such a complex system but also to develop new hypotheses on underlying mechanisms. Model parameter identification is a challenging optimization problem [21] with the objective function representing data misfit in a given norm. In this study, S-systems [22], a set of non-linear differential equations of a special form belonging to the power-law formalism are adopted as model. To estimate the model parameters and to capture the dynamics in gene expression data, Tominaga et al. [22] used standard evolutionary algorithms (EAs). Evolutionary computation is becoming a popular approach for solving S-system parameter identification [3, 10, 11, 15], mainly due to the multimodality and strong nonlinear parameter-dependencies in the problem.

To the best of our knowledge, this is the first attempt to adopt the gravitational search algorithm (GSA) for S-system parameter optimization for GRN. This study provides a conceptual analysis of the search behavior of the GSA. The analysis suggests a center-seeking (CS) bias in the search process of the GSA confirmed by a setting up a test similar to one proposed by Angeline [1] on a set of several widely used numerical benchmark problems with various optimization characteristics. To partially dilute the search bias of GSA, a solution inspired from the Simulated Big Bounce [6] algorithm is proposed.

The remainder of this paper is organized as follows. A short introduction to GRNs and to S-systems is provided in Section 2. The population based model for S-system parameter identification is presented in Section 3. A brief tour of GSA followed by a conceptual analysis of its CS behavior are presented in Section 3.1. Motivated by the observed CS bias of the standard GSA, Section 3.2 presents the mdGSA, a solution borrowed from the Simulated big bounce (SBB) algorithm [6]. The experimental setup adopted to check the CS bias of GSA and mdGSA and their suitability for gene network parameter identification are presented in Section 4. The final section draws conclusions and considers implications for future research.

2 Gene Regulatory Networks

GRNs in the cell are a complex dynamic network of interactions between the products of genes (mRNAs) and the proteins they produce, some of which in return act as regulators of the expression of other genes (or even their own gene) in the production of mRNA. While low cost methods to monitor gene expression through microarrays exist, we still know little about the complex interactions of these cellular components. Usually, sets of ordinary differential equations (ODEs) are used as mathematical models for these systems [23, 24], however they suffer from many assumptions critical to the equations themselves. S-system approaches, on the other hand, use time-independent variables to model these processes. Assuming the concentration of N proteins, mRNAs, or small molecules at time t is given by $y_1^t, y_2^t, \ldots, y_i^t, \ldots, y_N^t$, S-systems model the temporal evolution of the ith component at time t by power-law functions of the form (1).

$$\frac{dy_i^t}{dt} = \alpha_i \left(\prod_{j=1}^{N} y_j^{g_{ij}} \right) - \beta_i \left(\prod_{j=1}^{N} y_j^{h_{ij}} \right), \tag{1}$$

where N represents the number of genes involved in the GRN. The first term represents all factors that promote the expression of component i, y_i, whereas the second term represents all factors that inhibit its expression. In a biochemical engineering context, the non-negative parameters α_i , β_i are called rate constants, and real-valued exponents g_{ij} (G matrix, $[G]$) and h_{ij} (H matrix, $[H]$) are, respectively, referred to as kinetic order for synthesis and kinetic order for degradation.

$\mathbf{x} = \{\alpha, \beta, [G], [H]\}$ are the parameters that define the S-system. The total number of parameters in the S-system is $2N(N+1)$, meaning the number of parameters increases quadratically and can quickly become very large. The parameter estimation task is to determine model parameters such that the dynamic profiles fit the observation.

3 Population Based S-Systems Model Parameter Identification

Let us define the search space as the following:

$$E = \bigotimes_{d=1}^{D} [\mathrm{L}_x^d, \mathrm{U}_x^d], \tag{2}$$

with the objective of locating $\mathbf{x}^* \in E$, where $f(\mathbf{x}^*)$ is the extremum of a function $f(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$ and where L_x^d and U_x^d are respectively the lower and upper bound of the search domain at dimension d.

To guide the population in the search space, some measure of discrimination is needed. The most commonly used quality assessment criterion is the mean quadratic discrepancy between the observed expression pattern y_i^t and the model output $\hat{y}_i^t$ [16].

$$f = \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\frac{\hat{y}_i^t - y_i^t}{y_i^t} \right)^2 , \tag{3}$$

where T is the number of time points.

Having chosen an appropriate fitness function, the next section outlines the GSA.

3.1 A Brief Tour of the Gravitational Search Algorithm

Gravitational search algorithm (GSA) [18] is a relatively new technique that has been empirically shown to perform well on many function optimization problems [2,8,9,12–14,17,19,20]. GSA inspires from the evolution of complex structures in the universe. In its original version, GSA scatters particles in a feasible region of the search space where they interact with each other under Newtons gravitational force and move in the search area seeking optimal design variable. GSA shares features with several other competing schemes. Just like many of them, GSA has a way of sharing information between solutions. In contrast to EAs where solutions *die* at the end of each generation, in GSA, solutions survive through the course of the optimization process. This provides a substantial source of information for the population when searching the global optimum.

In GSA, just like many other population based optimization techniques, to guide the population in the search space E, some measure of discrimination is needed, referred here to as a fitness of each candidate solution $\mathbf{x}_i$. Each candidate solution is a particle with a mass M_i inversely proportional to its fitness $f(\mathbf{x}_i)$. A good solution is analogous to a particle with high mass and a poor solution represents a particle with a small mass. A particle with high mass resist change more than those with low mass and tend to have higher impact on other particles, thereby sharing their features with low quality solutions. The attractive gravitational force governs the movement of the particles in the search space. The search begins by an attractive force with a strength and direction as a function of the mass of particle itself, mass of other particles and its relative distance to the other particles. The force is applied to static particles of one under which their position in next time step changes and they gain a velocity. The quantity of the resultant force is determined by Newtons gravitational law. A solution with a higher mass exerts a stronger force compared to that of small mass. The kinetic energy stored in particles is a form of memory giving them the possibility to steer their movement under the influence of their memory and external forces. The sum of the force field and the particle's kinetic energy induced from its velocity and mass is the total force acting on them and together with its current position $\mathbf{x}_i(t)$ determines its next position $\mathbf{x}_i(t+1)$ in the search space.

GSA's basic steps in pseudo-code are shown in Algorithm (1). In original GSA [18], the mass of particles considering its quality is assignment as follows:

$$M_i = \frac{m_i}{\sum_{j=1}^{S} m_j}, i = 1, 2, \ldots, S \tag{4}$$

Algorithm 1. Pseudo code of gravitational search algorithm (GSA)

```
Input: Search space $E$, fitness function $f$, $S$, $G_0$, $\alpha$
 1: Initialize particle's location, $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_S)^T$
 2: while $t < MaxIteration$ do
 3:     Fitness calculation
 4:     Update $M_i$, $\forall i = 1, \ldots, S$                 ▷ According to (4) and (5)
 5:     Update G                                        ▷ According to (8)
 6:     Update attractive force $F_i^d$, $\forall i = 1, \ldots, S$
 7:     Update $\mathbf{v}_i$, $\forall i = 1, \ldots, S$                 ▷ According to (9)
 8:     Update $\mathbf{x}_i$, $\forall i = 1, \ldots, S$                ▷ According to (10)
 9:     $t++$                                ▷ $t$ is the number of iterations
10: end while
Output: $\mathbf{x}^*$ and $f(\mathbf{x}^*)$
```

where

$$m_i = \frac{f(\mathbf{x}_i) - \min_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}{\max_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j) - \min_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}, \tag{5}$$

and S is the number of particles. The resultant gravitational force acting on particle i in direction d is determined using Equation (6).

$$F_i^d = \sum_{j\in Kbest} r_j F_{ij}^d, \tag{6}$$

where $Kbest$ is a set of k particles with the highest mass, $r_j \sim \mathcal{U}(0,1)$ and F_{ij}^d is gravitational force exerted by particle j on particle i. To provide a better exploration in the early iterations, $|Kbest|$ is set at S in the beginning; however the exploration must be decreased gradually. Therefore choosing a decremented function for $|Kbest|$ increases the exploitation of the algorithm when the number of iterations rises.

The force exerted by particle j acting on particle i is defined as:

$$F_{ij}^d = G\frac{M_i \times M_j}{R_{ij} + \varepsilon}\left(x_j^d - x_i^d\right) \tag{7}$$

where R_{ij} is Euclidian distance between particles i and j. and G, the gravitational constant initialized at G_0 is determined using Equation (8) as

$$G = G_0 e^{-\alpha\frac{t}{T}} \tag{8}$$

where α is algorithmic tuning parameter.

The equations of motion of every particle is described using (9) and (10) as

$$\mathbf{v}_i(t+1) = \mathbf{R} \times \mathbf{v}_i(t) + \frac{\mathbf{F}_i}{M_i}.\Delta t, \tag{9}$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1).\Delta t, \tag{10}$$

where $\Delta t = 1$, $\mathbf{R} \sim \mathcal{U}(0,1)$ is an array of size D corresponding to each element in vector $\mathbf{v}_i$.

To overcome limitations observed in performance of GSA, the next Section presents a modification of mass assignment procedure of standard GSA.

3.2 mdGSA, a Mass-Dispersed GSA

The CS bias of the standard GSA is a serious barrier to its use as an optimization tool when recognizing the fact that usually no real-world optimization problem has its optimal solution at exact center of the search space. A more intense discrimination of solutions may be a partial solution to this problem. Inspired from the Simulated big bounce algorithm [6], a Mass bounded to the range of $[L_M, U_M]$ is assigned to every particle considering the fitness of each particle. g, the function that maps the fitness to the Mass $g : \mathbb{R} \to \mathbb{R},\ f(\mathbf{x}_i) \mapsto g\left(f\left(\mathbf{x}_i\right)\right),\ \forall \mathbf{x}_i \in \mathbf{x}$ can be any monotonically nondecreasing (and possibly time varying) function in principle with real values defined on a the set of fitness of particle $\mathbf{x}_i$ whose value is non-negative for $f(\mathbf{x}_i)$. We take g as a linear time-invariant strictly increasing function as following [6]:

$$M_i = g\left(f\left(\mathbf{x}_i\right)\right) = L_M + \left(U_M - L_M\right) \frac{f(\mathbf{x}_i) - \max\limits_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}{\min\limits_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j) - \max\limits_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}. \quad (11)$$

4 Experimental Setup and Results

When comparing the effectiveness of different optimization methods, a standard performance measure is the objective value a certain algorithm can reach within a certain predefined number of function evaluations. This is based on the assumption that the dominating factor in measuring computational effort is fitness evaluation, which is usually valid for complex optimization tasks of interest for real-world problems [5,7]. This, in the experiments, is modeled as if the maximum computational resource budget available to devote a task is limited. This is equivalent as if the maximum time budget for which the best solution has to be delivered is limited.

4.1 Parameter Settings

In all the experiments adopted in this study, the population size is set at 50. The total number of fitness evaluation for mathematical optimization problems is set at 100,000 and for GRN model parameter estimation is set at 200,000. The used GSA parameters presented in [18] is as follows: G_0 is set at 100, α is set at 20, $Kbest$ is set at number of particles, S, and is linearly decreased to 1 at the last iteration. For the mdGSA the common setting are the same as GSA settings and the upper and lower bound of mass are set at 1, .01 respectively.

As the studied optimization techniques are stochastic in nature, for a given function of a given dimension 30 independent runs where executed for each experimental setup with average best-of-run and standard deviation of results being reported along with the results of Wilcoxon Rank Sum test for statistical test of significance.

4.2 Standard Optimization Problems

To asses the performance of the GSA, a set of four standard optimization problems, each with distinguishing characteristics posing different difficulties to the optimization algorithm, were selected. The problems selected are benchmark when comparing different optimization algorithms and are taken from prior studies [1]. This set of optimization problems, in their original from, are crafted to have the optima at or near the center of search space.

Some population-based optimization techniques suffer from a notable search bias. They tend to perform best when the optimum is located at or very near to

Table 1. Test problems. Adopted from [1]

Function	Search Space-1 (SS-1)	Search Space-2 (SS-2)
Sphere (F1)	$[-80,100]^D$	$[-20,100]^D$
Rosenbrock (F2)	$[-480,600]^D$	$[-120, 600]^D$
Rastrigin (F3)	$[-40, 50]^D$	$[-10,50]^D$
Griewank (F4)	$[-4.0,5.0]^D$	$[-1.0,5.0]^D$

the center of the search space. When developing a new optimization algorithm this makes the comparison unreliable. So in this study, to move the optimal solutions from the center, the search space is cut out from one side, 10% of the whole search space in the case of search space-1 (SS-1) and 40% in the case of search space-2 (SS-2). The test beds and their associated search spaces are listed in Table 1.

Table 2 compares the GSA and mass-dispersed GSA (mdGSA) algorithms on the set of four adopted mathematical benchmark problems under two different search space and when the dimension D is set at 50. Against all expectations, the performance of the GSA is found to deteriorate as a result of shrinking the search space. That may be explained by search bias. As result of shrinking the search space, the optimal solution moves from the center of the search space. When an algorithm has a center-seeking bias, this move of the location of the optimal solution deteriorates its performance.

In GSA, a change in number of particles changes the mass assigned to them as a result of increase in the denumerator of Equation (4). This increase in denumerator smoothes out the difference between the mass of particles, making them relatively equally the same in exerting attractive force and equally resistant to change in their position as a result of applied gravitational force. The swarm may be considered as one object with uniform mass distribution. Under the Newtonian gravitational force this brings the particles closer to the center resulting in an increase in density of swarm. As a result, they accelerate faster towards the center. This may explain the observed center-seeking behavior of standard GSA on a set of standard optimization problems.

To measure center-seeking bias (CSB) of an algorithm, one needs a benchmark and a criteria. A benchmark is when the search space is shrunk by S_L% of the original search space, and the criteria is deterioration of the fitness value when the search space is shrunk by S_U%. Considering the case of GSA on F4 test function as an example, a measure for CSB is $\text{CSB}^{S_L,S_U}(\text{GSA}, \text{F4}) = (411.27 - 7.42)/(S_U - S_L) = 13.46$, when the S_L and S_U are set at 10 and 40 receptively. A preferred algorithm is one with smallest CSB, among other criterions.

The observed search bias of GSA was indeed a motivation to borrow the mass assignment mechanism of the Simulated big bounce (SBB) algorithm [6] and propose the mdGSA to partially resolve the search bias. In all the four test beds, the $\text{CSB}^{10,40}$ of mdGSA is lower than that of GSA (Table 2).

Table 2. Statistical results of 30 runs obtained by GSA and mdGSA. Mean: Mean of the Best Values, StdDev: Standard Deviation of the Best Values, CSB_{10}^{40}: center-seeking bias when S_L and S_U are set at 10 and 40 respectively.

Function		GSA SS-1	GSA SS-2	$\text{CSB}^{10,40}$	mdGSA SS-1	mdGSA SS-2	$\text{CSB}^{10,40}$
Sphere	Mean	4.41E-17	4.23E-17	~ 0	5.35E-11	5.37E-11	~ 0
	StdDev	(1.21E-17)	(9.59E-18)		(4.70E-12)	(3.85E-12)	
Rastrigin	Mean	27.79	110.83	2.76	36.31	100.62	2.14
	StdDev	(5.43)	(12.57)		(7.68)	(11.23)	
Rosenbrock	Mean	70.53	208.69	4.60	44.45	76.18	1.06
	StdDev	(39.39)	(134.71)		(0.257)	(66.65)	
Griewank	Mean	7.42	411.27	13.46	8.21E-4	1.31E-3	1.63E-5
	StdDev	(2.28)	(17.81)		(3.13E-3)	4.25E-3	

4.3 GRN Model Parameter Identification

To assess the performance of the methodologies studied here, two gene regulatory networks, NET1 and NET2, each consist of a network of two genes generated by the parameters given in Table 3 were adopted.

The gene expression levels of the networks are plotted in Figure 1 each consist of 50 time course of expression level per gene. The search space for α_i and β_i is limited to [20, 0.0] and for g_{ij} and h_{ij} to $[-4.0, 4.0]$.

The fitness transitions for different methodologies for NET1 and NET2 are plotted in a logarithmic scale in Figure 2. The Figures are average of 30 independent runs. In both NET1 and NET2, both the GSA and mdGSA start with a sharp fitness decrease in the beginning. GSA becomes almost stagnate after a short number of fitness evaluation. The mdGSA in both cases had much better progression compared to the GSA.

Preliminary analysis showed that neither the GSA nor the mdGSA produced normally distributed results under all settings. Consequently the GSA and mdGSA was compared using the nonparametric Wilcoxon Rank Sum test to determine which one finds the lowest fitness values. The test, in contrast to *t*-test is solely based on the order in which the observations from the two samples

Table 3. S-System Parameters for Network Model NET1 and NET2 adopted for model validation [22]

GRN	i	α_i	β_i	g_{i1}	g_{i2}	h_{i1}	g_{i2}
NET1	1	3	3	0	2.5	-1	0
	2	3	3	-2.5	0	0	2
NET2	1	3	3	0	-2.5	.1	0
	2	3	3	2.5	0	0	.1

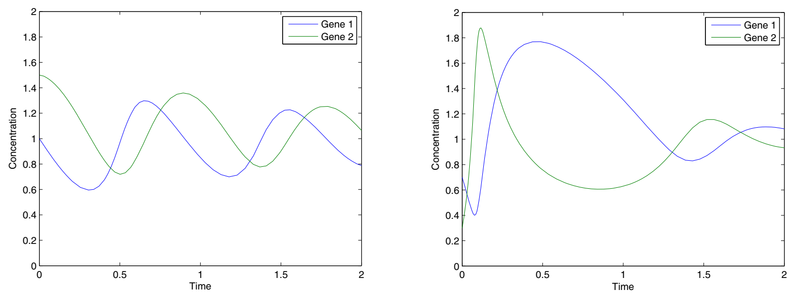


Fig. 1. Target time dynamics of first and second gene networks, NET1 and NET2

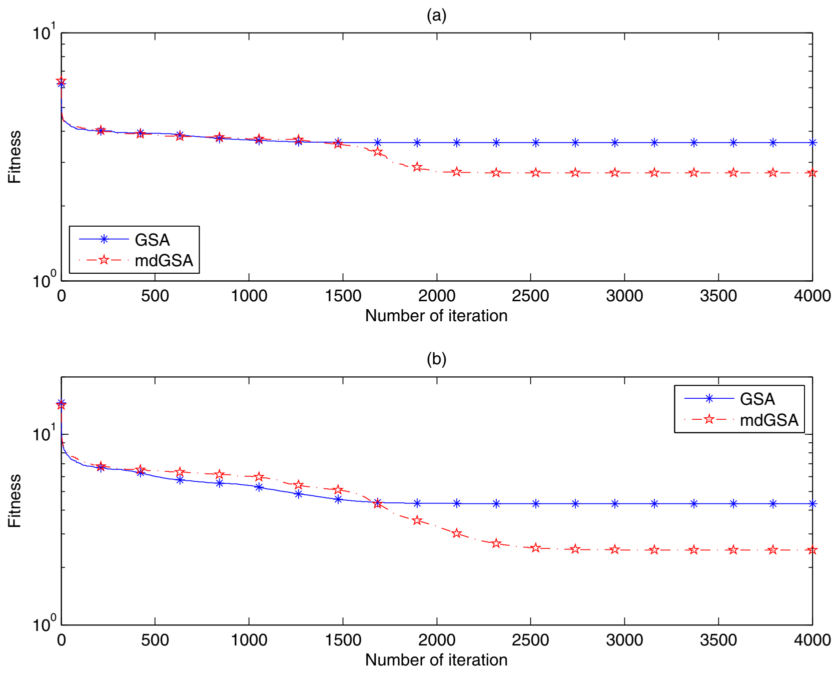


Fig. 2. Performance comparison of the GSA and mdGSA. (a) on NET1, (b) on NET2.

Table 4. A Wilcoxon Rank Sum test of the fitness of last generation for NET1 and NET2 (30 runs) obtained by GSA and mdGSA. Mean: Mean of the Best Values, StdDev: Standard Deviation of the Best Values.

GRN		Simulation results		
		Mean	StdDev	p-Value
NET1	GSA	3.60	0.37	2.14E-5
	mdGSA	2.39	1.30	-
NET2	GSA	4.32	1.61	3.32E-6
	mdGSA	2.43	0.87	-

fall. As presented in Table 4, the results of the proposed mdGSA are better than that of GSA when the standard cut-off for considering a p-value for a statistically significant difference is set at $p < 0.05$.

5 Conclusions and Future Work

In this paper, the GSA and its variant proposed in this paper, mdGSA, are employed for estimating genetic networks using S-system formalism. The performance of the mdGSA method, which enhances the global searching capability of GSA and alleviates its center-seeking bias, was verified using four standard benchmark problems and two networks. The experiments showed that the proposed method is capable to identify model parameters.

The center-seeking bias is a serious barrier to any optimization algorithm when recognizing the fact that no real-world optimization problem has its optimal solution at exact center of the search space. As part of our future work, we are interested in in-depth understanding the observed CS bias of the GSA.

Finally, we wish to apply the GSA and its variants algorithms to actual biological gene network and to conduct a comprehensive comparison against other popular population-based optimization algorithms.

References

1. Angeline, P.J.: Using selection to improve particle swarm optimization. In: International Conference on Evolutionary Computation, pp. 84–89 (1998)
2. Chatterjee, A., Mahanti, G.K., Pathak, N.N.: Comparative performance of gravitational search algorithm and modified particle swarm optimization algorithm for synthesis of thinned scanned concentric ring array antenna. Progress In Electromagnetics Research B 25, 331–348 (2010)
3. Chowdhury, A.R., Chetty, M.: An improved method to infer gene regulatory network using s-system. In: Congress of Evolutionary Computation (CEC 2011), pp. 1012–1019 (2011)
4. Crombach, A., Hogeweg, P.: Evolution of evolvability in gene regulatory networks. PLoS Computational Biology 4(7), e1000112 (2008)
5. Davarynejad, M., Ahn, C.W., Vrancken, J.L.M., van den Berg, J., Coello Coello, C.A.: Evolutionary hidden information detection by granulation-based fitness approximation. Applied Soft Computing 10(3), 719–729 (2010)
6. Davarynejad, M., van den Berg, J.: Simulated big bounce: a continuous space global optimizer. Technical report, Faculty of technology policy and management, Delft University of Technology, The Netherlands (2012)
7. Davarynejad, M., Vrancken, J., van den Berg, J., Coello Coello, C.A.: A Fitness Granulation Approach for Large-Scale Structural Design Optimization. In: Chiong, R., Weise, T., Michalewicz, Z. (eds.) Variants of Evolutionary Algorithms for Real-World Applications, vol. 87, pp. 245–280. Springer, Heidelberg (2012)
8. Duman, S., Güvenç, U., Sönmez, Y., Yörükeren, N.: Optimal power flow using gravitational search algorithm. Energy Conversion and Management 59, 86–95 (2012)
9. Duman, S., Güvenç, U., Yörükeren, N.: Gravitational search algorithm for economic dispatch with valve-point effects. International Review of Electrical Engineering (IREE) 5(6), 2890–2895 (2010)

10. Forghany, Z., Davarynejad, M., Snaar-Jagalska, B.E.: Gene regulatory network model identification using artificial bee colony and swarm intelligence. In: IEEE Conference on Evolutionary Computation (CEC 2012), pp. 949–954 (2012)
11. Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K., Tomita, M.: Dynamic modeling of genetic networks using genetic algorithm and s-system. Bioinformatics 19(5), 643–650 (2003)
12. Li, C., Zhou, J.: Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. Energy Conversion and Management 52(1), 374–381 (2011)
13. Li, C., Zhou, J., Xiao, J., Xiao, H.: Parameters identification of chaotic system by chaotic gravitational search algorithm. Chaos, Solitons & Fractals 45(4), 539–547 (2012)
14. Lopez-Molina, C., Bustince, H., Fernandez, J., Couto, P., De Baets, B.: A gravitational approach to edge detection based on triangular norms. Pattern Recognition 43(11), 3730–3741 (2010)
15. Nakayama, T., Seno, S., Takenaka, Y., Matsuda, H.: Inference of s-system models of gene regulatory networks using immune algorithm. Journal of Bioinformatics and Computational Biology 9, 75–86 (2011)
16. Noman, N., Iba, H.: Inference of gene regulatory networks using s-system and differential evolution. In: Genetic and Evolutionary Computation Conference, Washington, DC, pp. 439–446 (2005)
17. Precup, R.-E., David, R.-C., Petriu, E.M., Preitl, S., Paul, A.S.: Gravitational Search Algorithm-Based Tuning of Fuzzy Control Systems with a Reduced Parametric Sensitivity. In: Gaspar-Cunha, A., Takahashi, R., Schaefer, G., Costa, L. (eds.) Soft Computing in Industrial Applications. AISC, vol. 96, pp. 141–150. Springer, Heidelberg (2011)
18. Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S.: Gsa: a gravitational search algorithm. Information Sciences 179(13), 2232–2248 (2009)
19. Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S.: Filter modeling using gravitational search algorithm. Engineering Applications of Artificial Intelligence 24(1), 117–122 (2011)
20. Shaw, B., Mukherjee, V., Ghoshal, S.P.: A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems. International Journal of Electrical Power & Energy Systems 35(1), 21–33 (2012)
21. Sun, J., Garibaldi, J.M., Hodgman, C.: Parameter estimation using metaheuristics in systems biology: A comprehensive review. IEEE/ACM Transactions on Computational Biology and Bioinformatics 9(1), 185–202 (2012)
22. Tominaga, D., Okamoto, M., Maki, Y., Watanabe, S., Eguchi, Y.: Nonlinear numerical optimization technique based on a genetic algorithm for inverse problems: Towards the inference of genetic networks. In: German Conference on Bioinformatics Computer Science and Biology, pp. 127–140 (1999)
23. Tsai, K.Y., Wang, F.S.: Evolutionary optimization with data collocation for reverse engineering of biological networks. Bioinformatics 21(7), 1180 (2005)
24. Vilela, M., Chou, I.C., Vinga, S., Vasconcelos, A.T., Voit, E.O., Almeida, J.S.: Parameter optimization in s-system models. BMC Systems Biology 16(2), 35 (2008)

A Density Based Approach to the Access Point Layout Smart Distribution Grid Design Optimization Problem

Bin Zhang, Kamran Shafi, and Hussein A. Abbass

School of Engineering and Information Technology,
University of New South Wales,
Canberra, Australia
Bin.Zhang@student.adfa.edu.au, {k.shafi,h.abbass}@adfa.edu.au

Abstract. Advanced metering infrastructure (AMI) is an integral part of the smart grid. It plays a significant role in control and management for utilities. Along with its pervasiveness, effective AMI network design has drawn more attention. To some extent, the reliability and robustness of the whole system is partially pre-determined by the whole smart distribution network design. Location arrangement for Access Points (APs) is an important aspect of the smart distribution grid structure which influences the system performance greatly because an optimized network by itself is effective to reduce cost and deal with emergencies or threats such as a breakdown hazard. This paper is dedicated to employ multi-objective optimization formulations to analyze and solve this network design problem in the smart distribution grid.

Keywords: Smart Grid, Neighborhood Area Network, Layout Optimization, Multi-Objective Optimization, Genetic Algorithm.

1 Introduction

The utility industry is undergoing revolutionary changes towards the concept of “smart grid” or “intelligent grid”. This next-generation electricity grid is required to offer full visibility and pervasive control over its various assets and services [1]. In order to achieve this, the information and communication technology is merged into existing power system, which has already resulted in many innovations in the power generation, transmission, sub-transmission and distribution systems.

For this monitoring and control purpose, current methods for sensing, metering, and measurements at different levels of the power grid need to be upgraded [2]. In the smart distribution grid design, the concept of advanced metering infrastructure (AMI) has been investigated and various approaches have been proposed to support finer-grained demand-side management. For example, ANSI C12.22 and associated standards in North America have enabled a new generation of smart meters which utilize various wireless technologies for communication and networking. Typically, AMI is supposed to provide such capabilities for the smart grid [3]:

- Two-way communication, which means both the utilities' data collection process from metering points and the reverse process for these smart metering points to respond to the instructions from utilities in order to realize automatic service connection.
- Real time monitoring, which requires fine-grained collection of meter data with time stamping to support real time demand estimation and scheduling.
- Self-organization, which means that smart meters can be registered to the distribution network automatically.
- Self-healing, which means the distribution grid can reconfigure itself due to a failure in communications.
- Integration with utility billing system, outage management system and other applications.

In this paper, we will investigate the concept of smart metering in smart power distribution grid and the problem of distribution grid access point (AP) layout. The next sections of this paper are organized as follows: Section 2 explains the smart metering concept and analyzes the distribution network design. An optimization problem is converged regarding to the multi-objective access point layout in section 3. The preliminary experiment result is presented in section 4. Objective functions are revised in section 5. This paper concludes in section 6.

2 Background

2.1 Smart Metering in Power Distribution

Meter Reading System (MRS) enables utilities to read the consumption records, alarms and status from residential meters remotely. Traditionally, this task is taken manually by technicians visiting customers' premises. MRS evolves as various network technologies [4, 5] develops to allow meter reading automation.

However, the next-generation smart distribution grid puts forward higher requirements for meter reading. A smart grid will deploy and utilize advanced digital meters at all customer service locations [6]. These meters provide two-way communication. On the one hand, they record and measure waveforms, voltage, current, active power, reactive power, and allow time-of-use and real-time rate scheme and management. On the other hand, they also allow the utilities to take corrective actions like connecting or disconnecting services remotely based on the information collected by these smart meters.

These smart meters are not used alone. They can be integrated with various sensors, actuators and appliances within the customer's premises to form a home area network (HAN) using the low-power and advanced wireless network technologies such as ZigBee (IEEE 802.15.4) [7]. Based on this, several adjacent HANs constitute a neighborhood area network (NAN) sharing an access point of the distribution grid to communicate with the utility's operational center. An access point receives periodic input from each NAN within its reception range in the neighborhood and then relays or uploads the centralized information to the power utility using long-haul communication technologies such as Ethernet, GPRS/CDMA [8]. The access point is usually a

Micro Controller Unit (MCU) that provides a secure interface and conversion between two dissimilar network such as ZigBee and GPRS as well as data processing and storage capabilities. There are already some commercial products available in the market from manufacturers such as Freescale and Silver Spring Networks, under the name like data concentrator/collector, grid router and so on. A typical route for a smart meter to communicate with utility is shown in Figure 1.

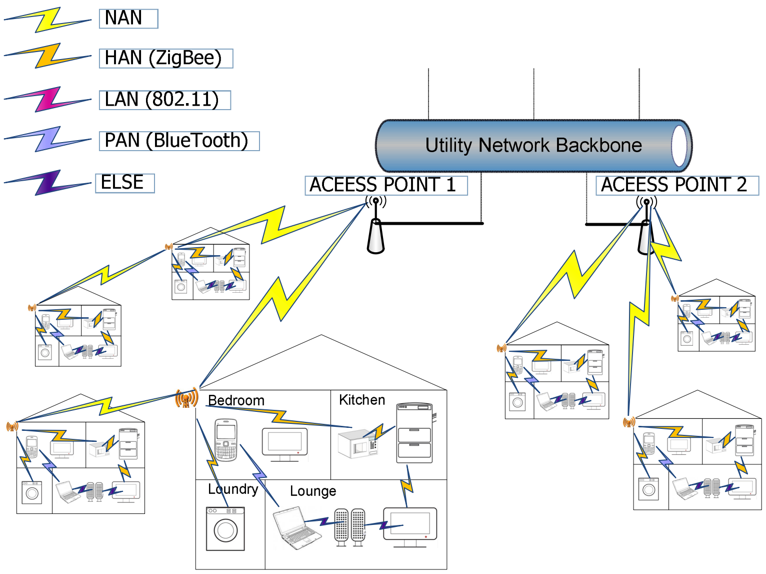


Fig. 1. Smart Distribution Grid Communication

Although the access points are small components compared to the whole smart grid, they are critical equipments and their performance affects the intelligence of the grid significantly. If an access point of the distribution network was out of control, it would result in a loss of reception of the whole neighborhood area network. Hence, the reliability and robustness of access points and their layout are key factors for an accurate information collection in distribution grid, which supports the correct and efficient decision making of utilities. This puts forward several requests when we deploy access points in the grid such as the need to reduce the number of AP devices used while maintain redundancy at a reasonable level. These requests will be elaborated when we formulate the objective functions in the proposed model.

2.2 Multi-Objective Optimization and NSGA-II

Optimization refers to the selection of a best element from some set of available candidate solutions. Optimization problems are common but of significant importance in the Engineering world and in various real situations including the smart distribution grid design that we are attempting to analyze. Although the existence of more than one objective in an optimization problem adds complexity, multi-objective optimization, which requires us to optimize simultaneously and systematically a collection of objective functions under certain constraints [9], is nothing but practicality because they are internal and natural demand of real problems. Further, in some cases, we also have to face nonlinear problems with their special demands and peculiar difficulties because we are surrounded and deeply involved in a world with nonlinear events [10].

Evolutionary computation, especially genetic algorithms is an optimization heuristic [11]. Although it can't promise perfection, evolution can always return more outstanding solutions to particular problems by a process involving iterative generation of solutions. It is quite natural, therefore, to apply evolutionary computation to solve difficult multi-objective optimization problems. Especially when dealing with nonlinear programming, classic methods like gradient descent, deterministic hill climbing, and purely random search (with no heredity) becomes unsatisfactory.

Mathematically, a multi-objective problem gives rise to a set of Pareto-optimal solutions [12]. Over the past few decades, in order to maintain a diverse set of solutions and move towards the true Pareto optimal set in one single simulation run, traditional genetic algorithms are customized to accommodate multi-objective problems [13] and a number of multi-objective evolutionary algorithms have been proposed or improved [14-17]. Among them, NSGA-II has become a popular approach. First, it has comparatively low computational complexity of non-dominated sorting which helps speed up the running time of such evolutionary algorithm. Second, it incorporates elitism to get better performance through keeping good solutions. Third, instead of assigning a sharing parameter in its older version, it adopts crowding distance to preserve diversity.

Based on the above considerations, we employ NSGA-II to analyze this multi-objective optimization problem. NSGA-II is able to find Pareto-optimum access point layout topologies efficiently.

3 Methods

3.1 Problem Formulation and Simulation

Taking both convenience and generalizability into consideration, we assume that the area to be optimized is normalized as a square of dimensionless size 1×1 in which a large number of HANs exist. These HANs are required to communicate with the utility network through certain access points. The HAN distribution is like a dot matrix. Since the distribution grid covers urban, suburban and rural areas, we assume whether a HAN exist or not at a specific position is determined by a predefined global density probability according to a uniform distribution.

There are n AP devices in the environment that are responsible for collecting related information within a specific distance which means that its control area is spherical. Here, we assume that all AP devices are identical and their control radiuses vary from 0.05 to 0.15 because sometimes repeaters will be introduced. We need to determine the location of AP devices to optimize a set of objectives for the whole distribution network. Let the number of AP devices n varies from 0 to 100. The decision variables (DV) of this problem will be

$$[x_1, y_1, r_1, f_1, ..., x_i, y_i, r_i, f_i, ..., x_{100}, y_{100}, r_{100}, f_{100}]$$

in which x_i and y_i means the horizontal and vertical coordinates. r_i stands for the perception radius of AP devices. f_i is a binary flag indicating if this AP device is used or not. So the number of access points in a solution is an integer in the interval [0, 100].

As mentioned earlier, there are several objectives that require us to optimize when we deploy access points in a smart distribution grid.

- From the perspective of the whole distribution grid, since its enormous size, the number of access point devices required to cover the whole service area should be minimized.
- From the perspective of HANs, first, the number of HANs without AP connection should be minimized. Second, we encourage more HANs to be in the reception ranges of at least two access points to create a certain level of redundancy. The concept of redundancy is important for a robust distribution network design. In the case that an access point device is not able to work properly, the area it covered can be transferred or partly transferred to another access point(s).

Naturally, two objective functions extracted are as follows (both for minimization):

$$f_1 = n$$

$$f_2 = \frac{N_{C=0}}{N_T} - \frac{N_{C>1}}{N_T}$$

Here, n means the number of access points;

$N_{C>1}$ means the number of HANs covered by more than 1 AP;

$N_{C=0}$ means the number of HANs not covered by any AP;

N_T means the total number of HANs.

3.2 Experimental Design

As mentioned earlier, we adopt NSGA-II to analyze this layout optimization problem. The necessary parameter setting for NSGA-II is shown in Table 1.

Table 1. Information for NSGA-II

Chromosome	Binary Coded
Population Size	100
Number of Generations	1000
Selection Method	Tournament Selection
Crossover Method	Uniform Crossover
Crossover Probability	0.8
Mutation Probability	0.01
Random Number Generation	Knuth's Random Number Generator [18]

4 Preliminary Experiment Result

NSGA-II came up with two different Pareto Fronts regarding two different HAN density probabilities, 0.2 and 0.8, as shown in Figure 2. We changed the sign of the second objective values in the figures for ease of visualization of the Pareto set.

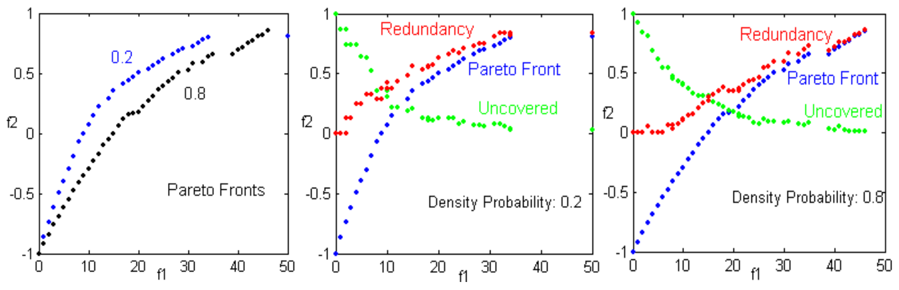


Fig. 2. Pareto Fronts and Components

Possible best layouts suggested by the genetic algorithm are shown in Figure 3. They are the trade-offs between the two objectives. The red stars are the location of HANs determined by the density probability, and are fixed once generated.

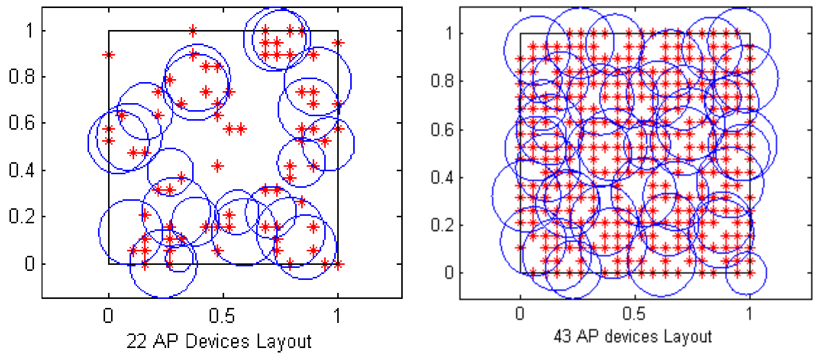


Fig. 3. Layouts Based on Different Density Probabilities

From the Pareto Front and layout examples, we find that these two objectives didn't perform well regarding two aspects that require further improvement. At first, reducing the rate of HANs without any AP connection should come first compared to redundancy. Full coverage is preferred when we compare two layout solutions. This means when we attempt to balance coverage and redundancy, if the number of APs is possible, we should emphasize coverage rather than redundancy until the coverage rate is reasonably high. Second, when evaluating the redundancy, we don't expect the coverage area of an AP is nested in a bigger one. That means the set of HANs covered by one AP is a subset of HANs covered by another AP. These aspects propel us to revise the objective functions to include and reflect such consideration.

5 Evolution of Objectives

5.1 Coverage First and Redundancy Reevaluation

As mentioned earlier, the coverage should have more weight than redundancy for the evaluation of the second objective. And we should remove the exaggerated part from previous redundancy evaluation. Hence, the new objectives are extracted as follows:

$$f_1 = n$$

$$f_3 = 20 \times \frac{N_{C=0}}{N_T} - \left(\frac{N_{C>1}}{N_T} - \frac{N_N}{N_T}\right)$$

N_N means the number of HANs of those APs that are nested in other APs;

20 is the weight factor preceding the rate of uncovered HANs.

One issue here is why we don't treat the full coverage as a constraint. In fact, we left out the coverage component in the objective function deliberately. The algorithm can come up with a dynamic balance between objective components. This helps to create a certain level of flexibility and adaptation in the proposed solutions to the user. On one hand, once we have enough devices, full coverage will always be the preference of evolution because of penalty assigned to under coverage. On the other hand, even if the AP devices are inadequate, the optimization will also deploy them in some areas where the local house density is high and they are in urgent need. It is a point of practicality.

The experiment is carried out under the same evolutionary settings. Pareto Fronts and layout examples of density probabilities 0.2 and 0.8 are shown in Figure 4, 5.

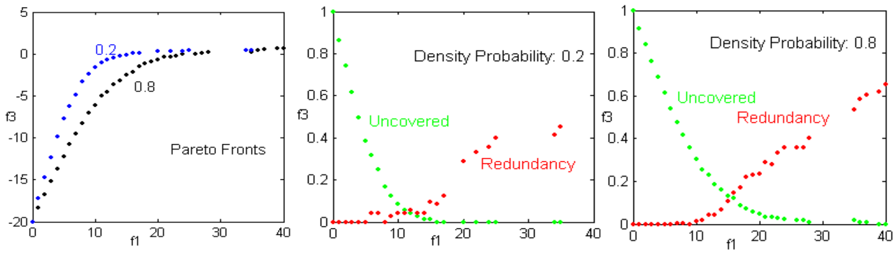


Fig. 4. Pareto Fronts and Components (5000 Generations)

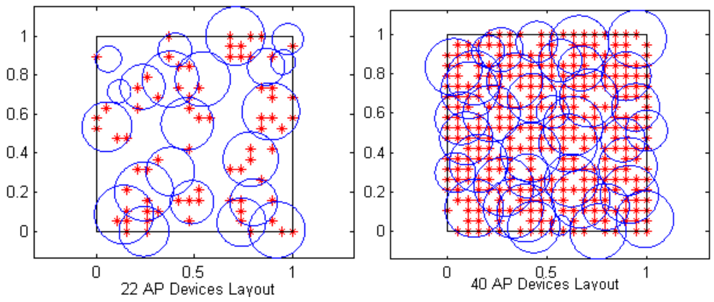


Fig. 5. Layouts Based on Different Density Probabilities

From the experiment's results, we find that the rate of uncovered HANs reduces faster than the first experiment and the redundancy becomes more accurate although it is comparatively smaller. We also find that, however, these two objectives didn't reflect requirements from the perspectives of APs. When we deploy APs, it is not economical to have AP covering low density of HANs. This raises the need to ensure that the AP will cover as many HANs as possible and reduce the reception radius. This also helps reduce the number of "empty" APs.

5.2 Add Density Evaluation from the Perspective of AP

Maximizing the HAN density in the perception range of each AP helps make our solution more efficient. First, this requires full utilization of an AP's covering ability. Second, this helps in distributing the coverage load more evenly among APs. Third, smaller reception radius implies low energy consumption for APs. The updated objective functions we designed are as follows:

$$f_1 = n$$

$$f_4 = 20 \times \frac{N_{C=0}}{N_T} - \left(\frac{N_{C>1}}{N_T} - \frac{N_N}{N_T}\right) - \frac{\frac{1}{n}\sum_{i=1}^{n}\frac{AP_i}{Area_i}}{\max\limits_{1\le i\le n}\left(\frac{AP_i}{Area_i}\right)}$$

AP_i means the number of HANs covered by a particular AP;

$Area_i$ means the area of a particular AP;

Based on the same previous experimental design but with the new objective functions, the Pareto Fronts and layout examples are shown in Figure 6, 7.

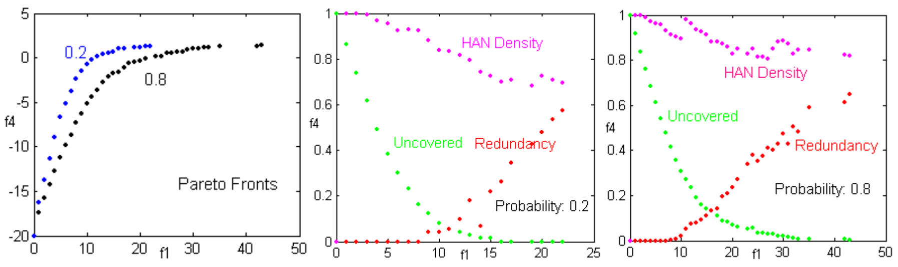


Fig. 6. Pareto Fronts and Components (10000 Generations)

From the result, we can see that the HAN density shows an overall descending trend as the number of AP devices grows. This is reasonable when only one AP existed with the maximum value of 1. The algorithm came up with maximum values of HAN density over the process of evolution, not among solutions in the same population.

Probably, there are still objectives required to be taken into account when we attempt to design a smart distribution grid. However, up to now, the objective functions we devised are effective regarding our network design goals. Each AP has a fair proportion of load based on the environment, the coverage is more promised and the redundancy is satisfying. Of course, not all these objectives can be achieved, multi-objective optimization is nothing but compromising.

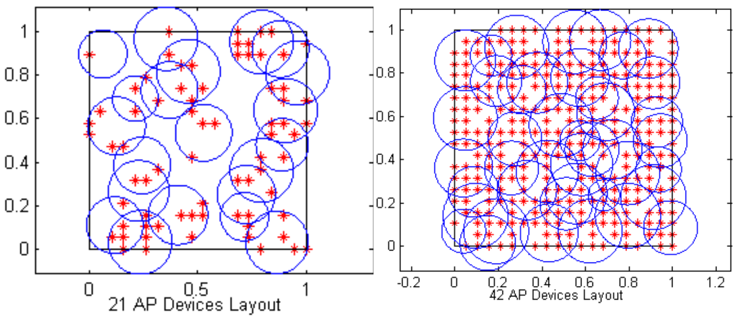


Fig. 7. Layouts Based on Different Density Probabilities

6 Conclusion

Many research works have been done to handle the layout problem in network design. Objective functions are required to be designed case by case. However, reviewing the methodology used can be extraordinarily rewarding. Among the various methods used, Genetic Algorithm is regarded as an effective tool helping to provide analysis and solutions. At the same time, the evolution of objective functions in this paper also reflects a cognitive process of how to deal with a comparatively complex engineering problem. It shows potential application of computational intelligence for educational purpose to educate and support engineers during the design process of the objective functions.

For future work, one side is to keep analyzing the smart distribution grid design to cover more issues in this area, such as cascading failure. On the other side, we will explore the utilization of computational intelligence techniques as a decision making mechanism for the purpose of educational and training system design.

References

1. Farhangi, H.: The path of the smart grid. IEEE Power and Energy Magazine 8(1), 18–28 (2010)
2. Momoh, J.: Smart Grid: Fundamentals of Design and Analysis. John Wiley & Sons (2012)
3. Hart, D.G.: Using AMI to realize the Smart Grid. In: IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century (2008)
4. Oksa, P., et al.: Considerations of Using Power Line Communication in the AMR System. In: 2006 IEEE International Symposium on Power Line Communications and Its Applications (2006)
5. Chih-Hung, W., Shun-Chien, C., Yu-Wei, H.: Design of a wireless ARM-based automatic meter reading and control system. In: IEEE Power Engineering Society General Meeting (2004)

6. Brown, R.E.: Impact of Smart Grid on distribution system design. In: IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century (2008)
7. Bennett, C., Highfill, D.: Networking AMI Smart Meters. In: IEEE Energy 2030 Conference (2008)
8. McDaniel, P., McLaughlin, S.: Security and Privacy Challenges in the Smart Grid. IEEE Security & Privacy 7(3), 75–77 (2009)
9. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26(6), 369–395 (2004)
10. Saaty, T.L., Bram, J.: Nonlinear mathematics. Dover (1964)
11. Mayr, E.: Toward a new philosophy of biology: observations of an evolutionist. Belknap Press of Harvard University Press (1988)
12. Deb, K., et al.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
13. Konak, A., Coit, D.W., Smith, A.E.: Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering &amp; System Safety 91(9), 992–1007 (2002)
14. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms, vol. 518. John Wiley & Sons, Inc. (2001)
15. Fonseca, C.M., Fleming, P.H.: Genetic Algorithms for multiobjective optimization: Formulation, Discussion and Generalization. In: The Fifth International Conference on Genetic Algorithms, San Mateo, CA (1993)
16. Srinivas, N., Deb, K.: Multiobjective function Optimization using nondominated sorting genetic algorithms. IEEE Transactions on Evolutionary Computation 2(3), 221–248 (1995)
17. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
18. Knuth, D.E.: Art of Computer Programming, 3rd edn. Seminumerical Algorithms. Addison-Wesley Professional (1997)

Multi-modal Valley-Adaptive Memetic Algorithm for Efficient Discovery of First-Order Saddle Points

Mostafa Ellabaan, Xianshun Chen, and Nguyen Quang Huy

Center of Computational Intelligence,
School of Computer Engineering,
Nanyang Technological University
{Mostafa.mhashim,seineriver}@gmail.com,
XSCHEN@ntu.edu.sg

Abstract. First-order saddle point represents an important landmark on the problem landscape. This point lies along the minimum energy path connecting two minima, more specifically at the point with maximum energy on the path. Unlike minima or maxima, to identify first-order saddle points require both maximization and minimization tasks. Finding such points is extremely difficult. In this paper, we present a real-coded memetic algorithm for locating first-order saddle points. The proposed algorithm leverage the advantage of valley-adaptive clearing scheme in maintaining multiple solutions and Schlegel algorithm in achieving fast and precise convergence. Empirical results shown that the proposed algorithms achieve more than 90% with converge speed of more than 100 fold when comparing to its evolutionary compeers.

Keywords: Multi-Modal optimization, memetic algorithm, Saddle points.

1 Introduction

First-order saddle points represent important landmarks in the problem landscape, with many applications in science and engineering. In engineering, saddle points have several applications, including the identification of x-junction in the images captured by robotic visual sensors and the construction of roadmaps in visual models to estimate human pose captured from monocular images [1]. In science, saddle points or transition structures play a key role in the chemical reaction rate theory [2, 3], especially for estimating the reaction rate of chemical reactions and processes. Due to their fleeting nature, transition structures are almost impossible to be isolated experimentally. Therefore, search for such structures computationally is unavoidable and remains a great challenge [4].

To identify first-order saddle points, a plethora of optimization methods have been proposed in the last decades. They can be classified as conventional and stochastic methods [4]. Conventional methods, on one hand, make use of domain specification information such gradient and/or hessian in the search process [5-10]. However, like most conventional numerical methods, it requires good initial guesses to locate the first-order saddle points. Due to the use of prior information, conventional methods are well-established to converge efficiently to a first-order saddle points, given an appropriate initial guess.

In stochastic search methods, on the other hand, a population of individuals is usually used to explore the search space [11]. To date, few stochastic methods have been proposed for locating of transition states, making it a fertile area for further research investigations. Examples methods include Chaudhury simulated annealing method [12] and Bungay et al. [11] as well as Chaudhury et al. GA [13]. Most stochastic methods proposed for finding transition states are plagued with many problems, including the lack of sufficient precision, slow convergence speed, as well as, the inability to maintain multiple saddle points in the search.

In this paper, we extend the effort proposed earlier by the authors in [4] to identify multiple saddle points both efficiently and precisely, utilizing the recent advances in computational intelligence, the memetic computing paradigm [14-17]. The proposed algorithm tidily integrates the valley-adaptive clearing scheme with an efficient local searcher, the Schlegel method. The proposed methodology has been compared to our earlier work and the state-of-the-art methods, showing advantage performance in terms of precision, number of uncovered saddle points as well as the convergence speed.

The paper is organized as follows: Section 2 provides a brief definition of the non-linear programming problem of first-order saddle points. The proposed method is presented in section 3 while Section 4 presents benchmark problems. Section 5 reports the results obtained from our computational study. Brief conclusion and future work are then stated in Section 6.

2 Problem Statement

A first-order saddle point is defined as a stationary point with a vanished gradient and only one negative eigenvalue in the hessian matrix [5]. These points can be mathematically expressed as:

$$\mathbf{X}_{\mathrm{T}} = \left\{ \mathbf{x}_k \middle| \left(\frac{6f(\mathbf{x}_k)}{6\mathbf{x}_k} = 0 \right) \& \ \left(\gamma_{k,i} < 0 \right) \& \, f(\mathbf{x}_k) < f_{max} \right\} \tag{1}$$

where $\mathbf{x}_{\mathrm{i}} \in \mathbf{R}^d$, d is the dimensional size, $f(\mathbf{x}_{\mathrm{i}}) \in \mathbf{R}$, $\mathbf{X}_{\mathrm{T}}$ is the set that includes all possible first-order saddle points which lies below a fitness threshold-f_{max}, and $\gamma_{k,i}$ is the only negative eigenvalue of the hessian matrix $\mathbf{H}_k$, respectively. '&' denotes a logical AND operator.

3 Multi-modal Valley-Adaptive Clearing Memetic Algorithm

In this section, a real-coded multi-modal valley-adaptive clearing memetic algorithm for locating first-order saddle points (see Fig. 1) is introduced. In this algorithm, a population of individuals is first randomly generated. Once population individuals are initialized, they undergo the evolutionary process. To comply with the principle of the natural selection in the evolutionary theory, the population individuals are evaluated to define their fitness on solving the problem, using (Eq. 2).

$$f_{\mathrm{ts}}(\mathbf{x}) = \frac{f(\mathbf{x}) - f_{max}}{\left(||\mathrm{g}|| + \varepsilon\right)\left((n-1) + \varepsilon\right)} \tag{2}$$

where f_{max} denotes the energy threshold below which the saddle point of interest lies. $f(\mathbf{x})$ donates the energy at current configuration $\mathbf{x}$, $||\mathbf{g}||$ is the gradient norm at $\mathbf{x}$, n is the number of negative eigenvalues, ε is chosen small to prevent division by zero error. First-order saddle points lie at the optima of the fitness function $f_{ts}(\mathbf{x})$ where the gradient norm vanishes and $n = 1$.

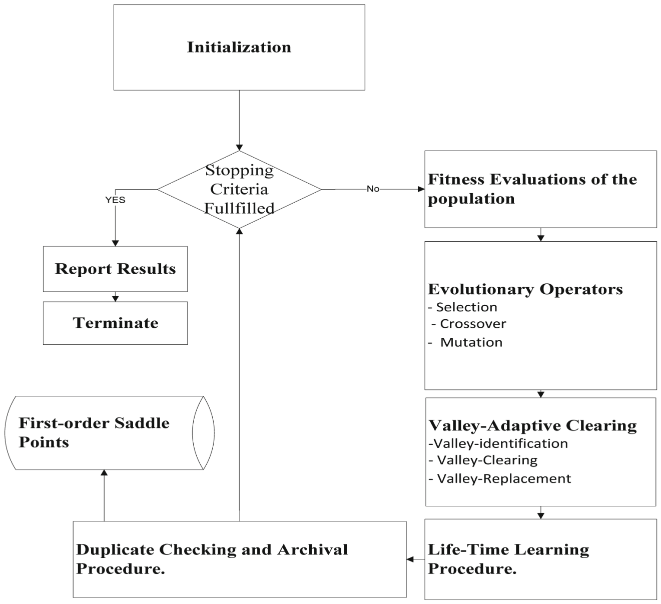


Fig. 1. A memetic algorithm for locating first-order saddle points

Individuals are subsequently selected to undergo the evolutionary process of crossover and mutation. Individual then undergo the valley-adaptive clearing where individuals are segregated into valley groups shall they share a common valley. A valley group then undergoes a valley replacement phase if it has a member satisfying first-order saddle point criteria in Eqn. (1). In such a case, the individuals are replaced into different basins of attraction in the effort toward discovery of new first-order saddle points. Otherwise, the valley group undergoes a valley clearing phase, where the best members (valley elites) are allowed to survive while others are relocated in order to explore the nearby first-order saddle points in the search space. Valley elites are thus allowed to undergo a life-time procedure. In this study, we consider Schlegel method as a life-time procedure for finding first-order saddle points. The same process repeats until a maximum number of generations or other stopping criteria have been satisfied.

4 Benchmark Problems

To visualize the first-order saddle points and test the performance of the proposed algorithm, four commonly used benchmark problems (see Fig. 2) are considered here. These problems are scalable, giving the flexibility to increase or decrease the domain ranges and the number of decision variables of interest. Their landscapes, also, contain a significant number of critical points including maxima, minima and saddle points. These test problems are detailed in what follows:

Problem 1: Sines function
Sines function defined in (Eq. 3) has a multi-modal landscape with approximately 84 first-order saddle points[1] within the range of [-10, 10].

$$f(x,y) = 1 + \sin^2(x) + \sin^2(y) - 0.1e^{-x^2-y^2} \tag{3}$$

where $x, y \in [-10, 10]$

Problem 2: Multi-function
Multi-function defined in (Eq. 4) represents a challenging 2D test function for saddle point algorithms. Its fitness landscape involves 144 saddle points within [-2, 2].

$$f(x,y) = -1 - x\sin(4\pi x) + y\sin(4\pi y + \pi) \tag{4}$$

where $x, y \in [-2, 2]$

Problem 3: Rastrigin function
Rastrigin function defined in (Eq. 5) represents another problem with a highly multi-modal landscape. Its landscape contains one global optima lying at (0, 0) and 223 first-order saddle points within [-5, 5].

$$f(\mathbf{x}) = 10n + \sum_{1}^{n} x_i^2 - 10\cos(2\pi x_i) \tag{5}$$

where $x_i \in [-5, 5]$

Problem 4: Schwefel Function
Schwefel function defined in (Eq. 6) denotes one of the other widely used test function in global optimization. It is composed of a large number of valleys and peaks. The distances between the global and local optima are also significant large, causing many search algorithms to be trapped at local optima. The 2D landscape in the range specified in (Eq. 6) contains about 66 first-order saddle point.

$$f(\mathbf{x}) = (\sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|}) \tag{6}$$

where $x_i \in [-350, 350]$

[1] The number of 1st order saddle points is approximated by means of visualization.

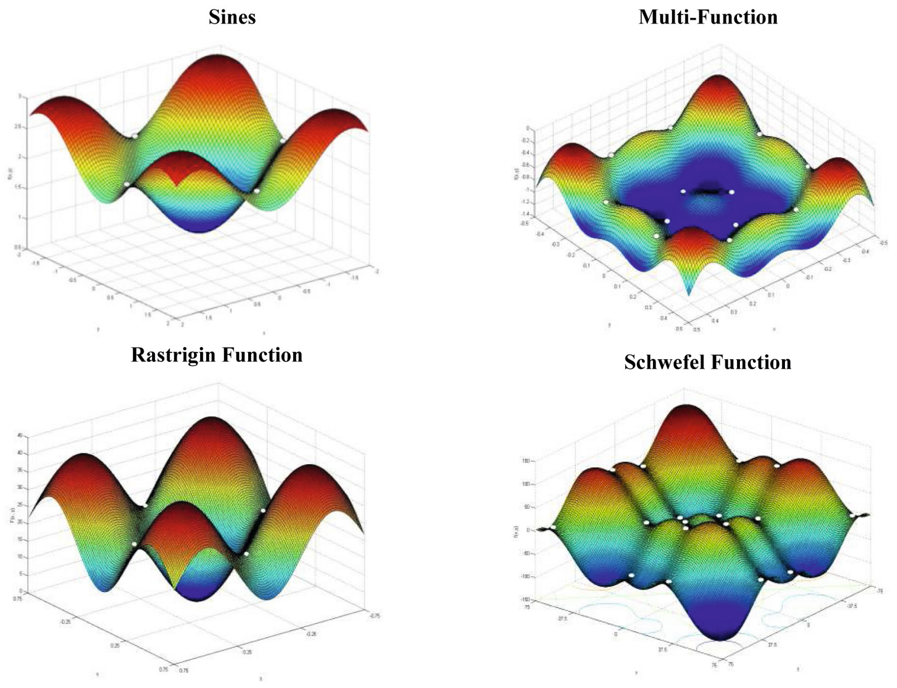


Fig. 2. 2D landscapes with some first-order saddle points highlighted by white dots

5 Experimental Study

In this section, we study the efficacy of the proposed memetic algorithm (MMA) for locating first-order saddle points and pit it against several existing state-of-the-art first-order saddle point optimization methods, including the stochastic multi-start local search(SMLS) [18] as representative of single-ended methods, while Chaudhury GA (ChGA) [13] and Bungay GA (BGA) [11] as representatives of the genetic algorithm. The sequential niching memetic algorithm (SNMA) [19] , AVAC [4, 20] and Mahalanobis with self-adaptive-niche radius CMA-ES (M-S-CMA-ES)[21], as representatives of recent advances in niching and memetic algorithms , are also considered for comparison in the present study. Brief descriptions of the algorithms considered are given in Table 1.

All methods work as baselines for comparison on the first-order saddle points optimization problem on a set benchmark problems in section 4, using a comprehensive set of performance measures that are described in Table 2. The experimental settings and numerical results obtained are reported. In particular, the algorithmic parameter settings used in the present study are listed in Table 3. Algorithms with local searcher or life time learning procedures are allowed to run with population size of 100 for a maximum of 1000 generations while all algorithms are allowed to run for a maximum of 10^6 gradient evaluations.

Table 1. Descriptions of the state-of-the-art algorithms considered

Category	Algorithm	Description
Single Ended Methods	Stochastic Multi-start Local Search (SMLS)	A typical stochastic multi-start solver with the Schlegel algorithm considered for locating first-order saddle points.
Genetic Algorithm	Bungay Genetic Algorithm (BGA [11])	A binary-coded GA proposed by Bungay to find the transition states of some chemical reactions such as HNC isomerization.
	Chaudhurry Genetic Algorithm (ChGA [13])	A real-coded genetic algorithm proposed to find the transition states on Lennard Jones clusters.
Recent Advances in Niching and/or Memetic Algorithms	Sequential Niching Memetic Algorithm (SNMA) [18]	An advanced niching technique that extends the effort of Beasley et al. [19] on sequential niching technique to locate multiple optimal using a local search to improve solution precision. SNMA incorporates a gradient-based local search process, a derating function and the basic clearing technique.
	M-S-CMA-ES, Mahalanobis Self-adaptive niche radii CMA-ES [21]	An advanced niching method which extend CMA-ES with adaptive niche radius strategy coupled with Mahalanobis distance for solving complex, non-isotropic high-dimensionality multi-modal optimization problems.

Table 2. Descriptions of the performance measures considered

Performance Measure	Description
Percentage of uncovered first-order saddle points	The ratio of the uncovered first-order saddle points and number of first-order saddle points that the problem possesses.
Precision of first-order saddle points	Precision of first-order saddle points in computing in term of tenth logarithmic of gradient norm ($\log_{10}(\|g\|)$)
Convergence Speed	The number of gradient evaluation incurred to attain the first encountered solution or first-order saddle point

Table 3. Algorithmic parameters settings

Parameter	Value
$P_{mutation}$	0.5
$P_{crossover}$	0.5
$r_{nicheRadius}$	0.1

This section is organized in the following manner. Section 5.1 represent results on the percentages and precision accuracy of uncovered first-order saddle points while Section 5.2 detail the convergence speed of different algorithms on different landscapes.

5.1 Percentages and Precision Accuracy of Uncovered First-Order Saddle Points

The percentage of uncovered first-order saddle points is considered as a crucial performance measure for studying the efficacy of any optimization algorithms. Here, several algorithms are investigated in the experimental study on discovery of multiple first-order saddle points where a maximum computational budget of million gradient calls is considered. During the search, all discovered first-order saddle points are archived. The experimental results are summarized in Fig. 3. As observed from the figure, the traditional stochastic multi-start local search locates higher percentages of first-order saddle points than traditional genetic algorithms of BGA and CGA- The percentages roughly maintained for accuracy up to -10. Multi-modal evolutionary algorithms attain several folds more than the stochastic multi-local search. However, one with increasing accuracies, the number of saddle points significantly decreases to approach zero which sometimes exponentially, as the accuracies go below -5. On contrary, the multi-modal memetic algorithms attain the highest percentage of the uncovered first-order saddle points which is maintained across higher accuracies as witnessed from the standard deviation of the first-order saddle points uncovered under different accuracies and landscapes on Table 4. MMA was observed to maintain the largest percentage of uncovered saddle points with more than 90% across different landscapes.

Table 4. Standard deviation of uncovered first-order saddle points using different algorithms with varying gradient precision accuracy, on all benchmark test problems

	SMLS	BGA	CGA	M-S-CMA-ES	AVAC	SNMA	MMA
Sine function	0	3.12	1.43	13.6	44.611	2.59	1.96
Multi-function	1.198	3.25	1.436	22.9	44.611	2.45	0.506
Rastrigin Function	1.187	3.25	1.436	9.66	42.6	0.5	1.96
Schwefel function	0.72	3.25	1.43	17.03	44.61	0.51	1.96
Average	0.77625	3.2175	1.433	15.7975	44.108	1.5125	1.5965

5.2 Convergence Speed

Convergence speed is another important performance measure that shows how fast a given method can converge to a solution. The number of gradient calls is considered as the convergence speed indicator. The fewer the number of the gradient calls, the more efficient is the algorithm.

Here, the number of gradient call is reported as the computational budget incurred by the method upon uncovering a single first-order saddle point with the gradient precision accuracy threshold of 10^{-1}. Experimental results tabulated in Table 5 show that canonical GA always outperform the recent multi-modal niching EA in the convergence speed. Imbalance between search space exploration and exploitation decreases the efficiency of the recent multi-modal niching EA of converging as fast as canonical EA. Such imbalance has been resolved in the multi-modal memetic evolutionary algorithm such SNMA and MMA. Among multi-modal memetic algorithm, the proposed MMA attains more than 10 folds convergence speed when compared to its multi-modal memetic compeer, the SNMA. The number of folds increases even more when compared to others methodologies. For example, the improvement rate, when compared to AVAC arrives, on the multi-function landscape, to 100 times. Clearly, the multi-modal memetic algorithm proposed incurred significantly small number of gradient calls as compared to conventional stochastic search method-the SMLS, the canonical GA and the recent advances in multimodal niching and memetic algorithms for the gradient precision criterion considered.

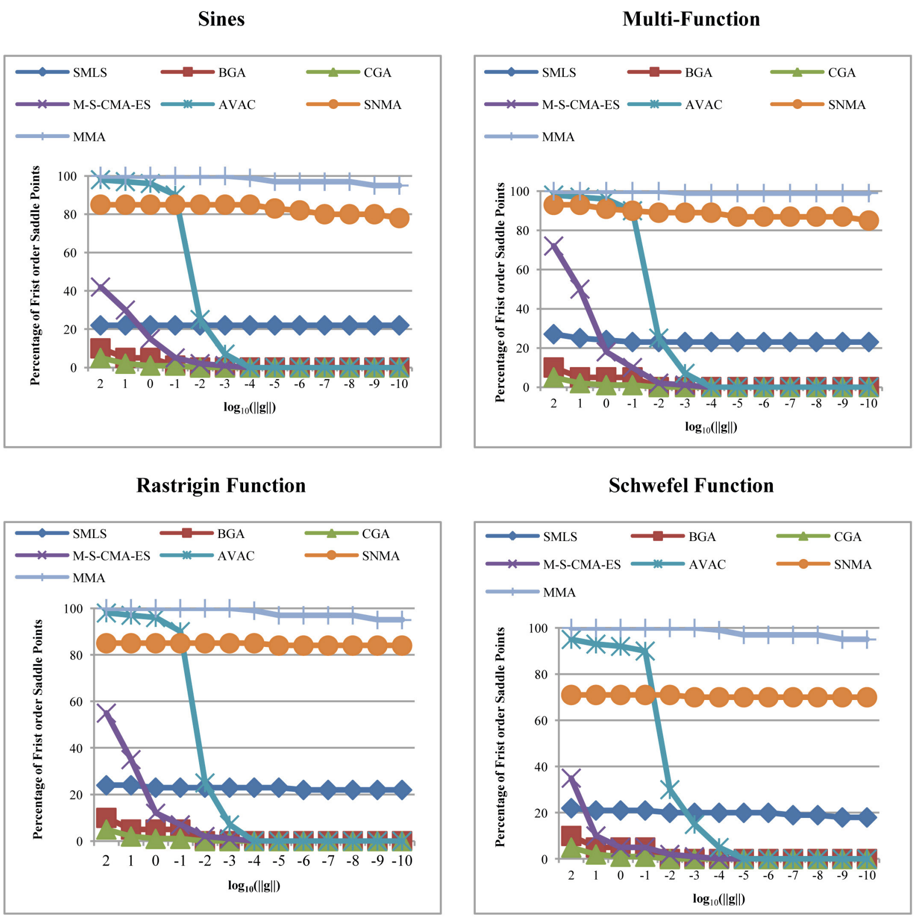


Fig. 3. Percentages of uncovered first-order saddle points using different algorithms with varying gradient precisions, on all benchmark test problems

Table 5. Number of gradient calls incurred by the algorithms considered on different landscapes

		Sines Function	Multi-Function	Rastrigin Function	Schwefel Function
SMLS	SMLS -Schlegel	1,187	1,313	1,207	1,086
Canonical EA	BGA	3,758	2,750	3,294	4,532
	CGA	4,246	3,147	3,972	5, 241
Recent Multi-Modal Niching EA	M-S-CMA-ES	7,598	5,479	6,654	15,470
	AVAC	4,598	4,010	4,427	6,470
Multi-Modal Memetic EA	SNMA	950	780	1,265	1,725
	MMA	70	90	60	250

6 Conclusion

In this paper, a novel multi-modal memetic algorithm for finding first-order saddle points has been proposed. Some important aspects for implementing the algorithm were discussed. Unlike canonical EAs that merely consider stochastic search operators, memetic algorithms consider, in addition to the stochastic search operators, an individual learning stage that refines the prospective solutions. In the proposed algorithm, we considered, in addition to the Schlegel algorithm – the local search algorithm, valley-adaptive clearing scheme to maintain multiple first-order saddle points. Experimental studies on benchmark problems show the efficacy of the proposed method in terms of solution quality and convergence speed as compared to other counterparts. The multi-modal valley-adaptive clearing memetic algorithm maintains more than 90% of saddle points in several synthetic landscapes with gradient precision lower than 10^{-10}.

The proposed framework has been applied for several biochemical systems [22] such as water clusters [23] and small molecules [24]. Next, we are planning to extend the proposed algorithm to locate first-order saddle points in complex biochemical systems, considering domain knowledge, incorporation of machine learning, symbiosis of local searcher [25], multi-scale search strategies [26] and advanced high performance computing paradigms.

References

[1] Lim, K.K., Ong, Y.S., Lim, M.H., Chen, X., Agarwal, A.: Hybrid ant colony algorithms for path planning in sparse graphs. Soft Computing-A Fusion of Foundations, Methodologies and Applications 12(10), 981–994 (2008)
[2] Toda, M.: Transition State Theory Revisited. Wiley-Interscience, City (2002)
[3] Jones, D., Sleeman, B.: Differential equations and mathematical biology. CRC Press (2003)
[4] Ellabaan, M.M.H., Ong, Y.S., Lim, M.H., Kuo, J.-L.: Finding Multiple First Order Saddle Points Using a Valley Adaptive Clearing Genetic Algorithm. In: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 2009), Daejeon, Korea (2009)
[5] Trygubenko, S., Wales, D.: A Doubly Nudged Elastic Band Method for Finding Transition States. Chem. Phy. 120(5), 2082–2094 (2004)
[6] Reddy, C.K., Chiang, H.D.: Stability boundary based method for finding saddle points on potential energy surfaces. J. Comput. Biol. 13(3), 745–766 (2006)
[7] Goodrow, A., Bell, A.T., Head-Gordon, M.: Transition state-finding strategies for use with the growing string method. Journal of Chemical Physics 130, 24 (2009)
[8] del Campo, J.M., Koster, A.M.: A hierarchical transition state search algorithm. Journal of Chemical Physics 129(2), 12 (2008)
[9] Henkelman, G., Jonsson, H.: A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives. Chem. Phy. 15(111), 7010–7022 (1999)

[10] Olsen, R.A., Kroes, G.J., Henkelman, G., Arnaldsson, A., Jonsson, H.: Comparison of methods for finding saddle points without knowledge of the final states. Journal of Chemical Physics 121(20), 9776–9792 (2004)
[11] Bungay, S.D., Poirier, R.A., Charron, R.J.: Optimization of transition state structures using genetic algorithms. J. Math. Chem. 28(4), 389–401 (2000)
[12] Chaudhury, P., Bhattacharyya, S.P.: A simulated annealing based technique for locating first-order saddle points on multidimensional surfaces and constructing reaction paths: several model studies. Theochem-J. Mol. Struct. 429, 175–186 (1998)
[13] Chaudhury, P., Bhattacharyya, S.P., Quapp, W.: A genetic algorithm based technique for locating first-order saddle point using a gradient dominated recipe. Chem. Phys. 253(2-3), 295–303 (2000)
[14] Nguyen, Q.H., Ong, Y.S., Lim, M.H.: A Probabilistic Memetic Framework. IEEE Transactions on Evolutionary Computation 13(3), 604–623 (2009)
[15] Chen, X., Ong, Y.S., Lim, M.-H., Tan, K.C.: A Multi-Facet Survey on Memetic Computation. IEEE Transactions on Evolutionary Computation 15(5), 591–607 (2011)
[16] Ong, Y.S., Lim, M., Chen, X.: Memetic Computation:Past, Present & Future Research Frontier. IEEE Computational Intelligence Magazine 5(2), 24–31 (2010)
[17] Lim, M.-H., Gustafson, S., Krasnogor, N., Ong, Y.-S.: Editorial to the first issue. Memetic Computing 1(1), 1–2 (2009)
[18] Kok, S., Sandrock, C.: Locating and Characterizing the Stationary Points of the Extended Rosenbrock Function. Evolutionary Computation 17(3), 437–453 (2009)
[19] Vitela, J.E., Castaños, O.: A real-coded niching memetic algorithm for continuous multimodal function optimization. In: IEEE Congress on Evolutionary Computation (2008)
[20] Ellabaan, M.M.H., Ong, Y.S.: Valley-Adaptive Clearing Scheme for Multimodal Optimization Evolutionary Search. In: The 9th International Conference on Intelligent Systems Design and Applications (ISDA 2009), Pisa, Italy (2009)
[21] Shir, O.M., Emmerich, M., Bäck, T.: Adaptive niche radii and niche shapes approaches for niching with the CMA-ES. Evolutionary Computation 18(1), 97–126 (2010)
[22] Ellabaan, M.M.H., Ong, Y.S.: Experiences on memetic computation for locating transition states in biochemical applications. In: Proceedings of the GECCO, Philadelphia, Pennsylvania, USA (2012)
[23] Ellabaan, M., Ong, Y.S., Nguyen, Q.C., Kuo, J.-L.: Evolutionary Discovery of Transition States in Water Clusters. Journal of Theoretical and Computational Chemistry 11(5) (2012)
[24] Ellabaan, M.M., Handoko, S.D., Ong, Y.S., Kwoh, C.K., Bahnassy, S.A., Elassawy, F.M., Man, H.Y.: A tree-structured covalent-bond-driven molecular memetic algorithm for optimization of ring-deficient molecules. Computers & Amp; Mathematics with Applications
[25] Le, M.N., Ong, Y.S., Jin, Y., Sendhoff, B.: A Unified Framework for Symbiosis of Evolutionary Mechanisms with Application to Water Clusters Potential Model Design. IEEE Computational Intelligence Magazine 7(1), 20–35 (2012)
[26] Nguyen, Q.C., Ong, Y.S., Soh, H., Kuo, J.-L.: Multiscale Approach to Explore the Potential Energy Surface of Water Clusters $(H_2O)_n$ $n \leq 8$. The Journal of Physical Chemistry A 112(28), 6257–6261 (2008)

Ensemble Fuzzy Rule-Based Classifier Design by Parallel Distributed Fuzzy GBML Algorithms

Hisao Ishibuchi, Masakazu Yamane, and Yusuke Nojima

Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
{hisaoi,nojima}@cs.osakafu-u.ac.jp,
masakazu.yamane@ci.cs.osakafu-u.ac.jp

Abstract. We have already proposed an island model for parallel distributed implementation of fuzzy genetics-based machine learning (GBML) algorithms. As in many other island models, a population of individuals is divided into multiple subpopulations. Each subpopulation is assigned to a different island. The main characteristic feature of our model is that training patterns are also divided into multiple training data subsets. Each subset is assigned to a different island. The assigned subset is used to train the subpopulation in each island. The assignment of the training data subsets is periodically rotated over the islands (e.g., every 100 generations). A migration operation is also periodically used. Our original intention in the use of such an island model was to decrease the computation time of fuzzy GBML algorithms. In this paper, we propose an idea of using our island model for ensemble classifier design. An ensemble classifier is constructed by choosing the best classifier in each island. Since the subpopulation at each island is evolved using a different training data subset, a different classifier may be obtained from each island to construct an ensemble classifier. This suggests a potential ability of our island model as an ensemble classifier design tool. However, the diversity of the obtained classifiers from multiple islands seems to be decreased by frequent training data subset rotation and frequent migration. In this paper, we examine the effects of training data subset rotation and migration on the performance of designed ensemble classifiers through computational experiments.

Keywords: Genetics-based machine learning (GBML), genetic fuzzy systems, fuzzy rule-based classifiers, parallel evolutionary algorithms, island model.

1 Introduction

Since the early 1990s [1]-[4], evolutionary algorithms have been frequently used for fuzzy system design. Such a hybrid approach is referred to as a genetic fuzzy system (GFS [5]-[8]). This is because genetic algorithms have been mainly used for fuzzy system design. Multi-objective evolutionary algorithms have also been used for fuzzy system design [9]-[12]. Such a multi-objective hybrid approach is often referred to as a multi-objective genetic fuzzy system (MoGFS).

Applications of evolutionary algorithms to machine learning are called genetics-based machine learning (GBML). Many GBML algorithms have been proposed for rule-based classifier design [13]-[16]. Fuzzy GBML is GBML for fuzzy rule-based classifier design [17]-[20], which can be viewed as a special class of GFS.

GBML algorithms are often categorized into three classes: Pittsburgh, Michigan and iterative rule learning (IRL) approaches [16]. In Pittsburgh approach, an entire rule-based classifier is coded as an individual. As a result, a population of individuals is a set of rule-based classifiers. Since the fitness of each individual is directly related to the performance of the corresponding rule-based classifier, Pittsburgh approach can directly maximize the performance of rule-based classifiers through the search for individuals with high fitness values. In Michigan approach, a single rule is coded as an individual. A population of individuals is handled as a rule-based classifier. Thus Michigan approach indirectly maximizes the performance of rule-based classifiers through the search for good rules with high fitness values. In IRL approach, a single rule is coded as an individual as in Michigan approach. A single rule is obtained from a single run of a GBML algorithm in IRL approach. Thus multiple runs are needed to design a rule-based classifier. Since IRL approach searches for a single best rule in each run, classifier optimization is indirectly and sequentially performed.

In order to drastically decrease the computation time of fuzzy GBML algorithms in Pittsburgh approach, we proposed an idea of parallel distributed implementation [21]-[24] using an island model. As in other island models, a population of individuals is divided into multiple subpopulations. Our island model also divides training patterns into multiple training data subsets. In each island of our island model, a subpopulation is evolved using a training data subset. Let N_{CPU} is the number of available processors for parallel computation. Since both the population size and the training data size are decreased to $1/N_{\text{CPU}}$ of their original size in each island, the speedup by our island model is the order of $1/(N_{\text{CPU}})^2$. This is the main advantage of our island model over other parallel implementation methods with the speedup of the order of $1/N_{\text{CPU}}$.

In this paper, we propose an idea of using our island model for fuzzy rule-based ensemble classifier design. In our former studies [21]-[24], only a single best fuzzy rule-based classifier was selected from all individuals at the final generation. With no increase in computation load, we can choose a single best fuzzy rule-based classifier from each island to construct an ensemble classifier. Since a different training data subset is used in each island, a different classifier is likely to be obtained from each island. We examine the usefulness of our idea through computational experiments.

This paper is organized as follows. First we briefly explain fuzzy rule-based classifiers and fuzzy GBML algorithms in Section 2. In Section 3, we explain our island model for parallel distributed implementation of fuzzy GBML algorithms. Then we report experimental results in Section 4 where a single best classifier and an ensemble classifier are compared with each other. This comparison is performed under various specifications of two important parameters in our island model. One is a training data subset rotation interval. This parameter controls the frequency of the rotation of training data subsets over islands. The other is a migration interval. This parameter controls the frequency of the migration of individuals (i.e., the migration of fuzzy rule-based classifiers) over subpopulations. Frequent rotation and/or frequent migration may decrease the diversity of classifiers in different subpopulations, which may lead to the deterioration in the performance of designed ensemble classifiers. The effects of rotation and migration are examined through computational experiments in Section 4. Finally we conclude this paper in Section 5.

2 Fuzzy Rule-Based Classifiers and Fuzzy GBML Algorithms

In this paper, we used the same parallel distributed fuzzy GBML algorithm as in our former study [24]. Our fuzzy GBML algorithm has a framework of Pittsburgh approach (i.e., an individual is a set of fuzzy rules). Michigan approach is used as a kind of local search for each individual. Our fuzzy GBML algorithm can be viewed as a hybrid algorithm for fuzzy rule-based classifier design (see [17] for details).

Let us assume that we have an n-dimensional pattern classification problem with m training patterns $\mathbf{x}_p = (x_{p1}, ..., x_{pn})$, $p = 1, 2, ..., m$. We use fuzzy rules of the following type for our classification problem:

$$\text{Rule } R_q : \text{If } x_1 \text{ is } A_{q1} \text{ and ... and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \tag{1}$$

where R_q shows the qth fuzzy rule, A_{qi} is an antecedent fuzzy set ($i = 1, 2, ..., n$), C_q is a class label, and CF_q is a rule weight.

Let S be a fuzzy rule-based classifier for our classification problem. The fuzzy rule-based classifier S is a set of fuzzy rules of the type in (1). When a pattern $\mathbf{x}_p$ is to be classified by S, a single winner rule is chosen for $\mathbf{x}_p$ from S. The selection of the winner rule for $\mathbf{x}_p$ is based on the compatibility grade of each fuzzy rule R_q with $\mathbf{x}_p$ and the rule weight CF_q. More specifically, first the product of the compatibility grade and the rule weight is calculated for each fuzzy rule in S as a winner rule selection criterion. Then the fuzzy rule with the maximum product is chosen as the winner rule for $\mathbf{x}_p$, which is assigned to the consequent class of the winner rule.

Since our study on fuzzy rule-based classifiers in the early 1990s [25], the single winner-based fuzzy reasoning method has been frequently used in fuzzy rule-based classifiers together with fuzzy rules of the type in (1). For other types of fuzzy reasoning methods and fuzzy rules for classification problems, see [26], [27].

Our fuzzy GBML algorithm is used to find the best fuzzy rule-based classifier S with respect to the following fitness function (This fitness function is minimized):

$$fitness(S) = w_1 f_1(S) + w_2 f_2(S) + w_3 f_3(S), \tag{2}$$

where w_1, w_2 and w_3 are non-negative weights, and $f_1(S)$, $f_2(S)$ and $f_3(S)$ are the following measures to evaluate the fuzzy rule-based classifier S:

$f_1(S)$: The error rate of S on training patterns in percentage,
$f_2(S)$: The number of fuzzy rules in S,
$f_3(S)$: The total rule length over fuzzy rules in S.

The total rule length is the total number of antecedent conditions of fuzzy rules in S. In the fitness function in (2), $f_1(S)$ is an accuracy measure of S while $f_2(S)$ and $f_3(S)$ are complexity measures of S. These three measures are combined into a single fitness function in (2). It is possible to use these measures as separate objectives in multi-objective fuzzy GBML algorithms [10]. In this paper, we use the weighted sum fitness function as in our former study [24].

In our fuzzy GBML algorithm, a fuzzy rule-based classifier (i.e., a set of fuzzy rules) is coded as an integer string. More specifically, antecedent fuzzy sets of fuzzy rules in a fuzzy rule-based classifier are represented by an integer string. The string length is not fixed (i.e., variable string length) because the number of fuzzy rules in fuzzy rule-based classifiers is not pre-specified. The consequent class and the rule weight of each fuzzy rule are not coded. This is because they can be easily determined in a heuristic manner from compatible training patterns with each fuzzy rule (e.g., the consequent class is the majority class among the compatible training patterns [28]).

3 Island Model for Parallel Distributed Implementation

Island models [29]-[34] have been frequently used for the speedup of evolutionary algorithms through parallel implementation. We also use an island model for parallel distributed implementation of our fuzzy GBML algorithms. As in other island models, a population of individuals (i.e., rule sets) is divided into multiple subpopulations. Each subpopulation is viewed as an island. The number of subpopulations is the same as the number of processors for parallel computation. A single processor is assigned to each subpopulation (i.e., each island). We use a simple migration operation to periodically send a copy of the best rule set in each island to the next one. We assume a ring structure of islands when the migration operation is executed (i.e., when a copy of the best rule set is sent to the next island). The worst rule set in each island is removed just before migration in order to maintain the subpopulation size unchanged. The number of generations between consecutive executions of migration is referred to as a "migration interval". This is an important parameter in our island model.

The main characteristic of our island model is that training patterns are divided into training data subsets. The number of training data subsets is the same as the number of subpopulations (i.e., the number of islands). A different training data subset is assigned to each island. The assigned training data subsets are periodically rotated over the islands with a ring structure. That is, the training data subset assigned to each island is periodically moved to the next island. The number of generations between consecutive executions of rotation is referred to as a "rotation interval". This is also an important parameter in our island model. Our island model is illustrated in Fig. 1 where training data and a population of rule sets are divided into seven subsets.

An interesting setting in our island model is that the training data subset rotation is performed in the opposite direction to the migration (see Fig. 1). That is, the training data subset at the ith island is moved to the $(i+1)$th island while a copy of the best rule set in the ith island is sent to the $(i-1)$th island. This is because the migration severely counteracts positive effects of the training data subset rotation when they are performed in the same direction at the same generation. In this case, a copy of the best rule set is sent to the next island together with the training data subset. For details of negative effects of such a synchronized migration and rotation, see [24].

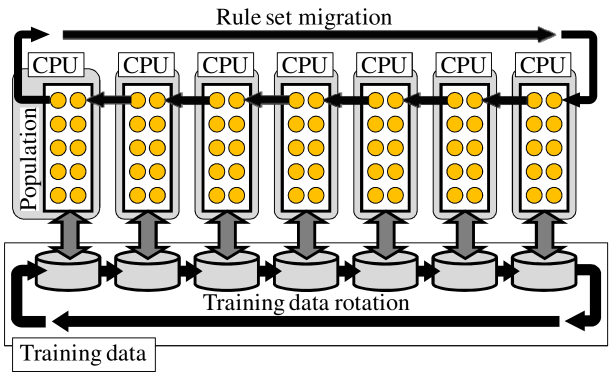


Fig. 1. Our island model for parallel distributed implementation [23]

4 Experimental Results

In our computational experiments, we applied our fuzzy GBML algorithm to the satimage data set in the KEEL project database [35]. The satimage data set has 6,435 patterns from six classes. Each pattern has 36 attributes. Since the satimage data set has many attributes (i.e., 36 attributes), we applied our fuzzy GBML algorithm to this data set using a large computation load: 50,000 generations of a population with 210 rule sets. Our computational experiments were performed on a workstation with two Xeon 2.93 GHz quad processors (i.e., eight CPU cores in total). Among the eight CPUs of the workstation, seven CPUs were used for parallel computation. This means that a population of 210 rule sets was divided into seven subpopulations of size 30 in our island model for parallel distributed implementation.

In our computational experiments, the following three variants of our fuzzy GBML algorithm were compared:

1. Standard non-parallel non-distributed model for single classifier design [17]
2. Our island model for single classifier design [23]
3. Our island model for ensemble classifier design

The standard non-parallel non-distributed algorithm was executed on a single CPU of the workstation while seven CPUs were used in the two variants of parallel distributed implementation. We used the ten-fold cross-validation procedure (i.e., 10CV) in our computational experiments. The 10CV was iterated three times to calculate the average test data accuracy as well as the average training data accuracy of designed classifiers by each of the above-mentioned three variants (i.e., 3×10CV was used for performance evaluation in this paper).

In the first two variants for single classifier design, the best rule set with respect to the fitness function in (2) for all training patterns was chosen from the final population. In the last variant for ensemble classifier design, the best rule set with respect to the fitness function in (2) for the assigned training data subset was chosen from each subpopulation. The selected seven rule sets were used as seven fuzzy rule-based classifiers in an ensemble classifier. Pattern classification was performed in the designed ensemble classifier using a simple majority voting scheme as follows: First a pattern was classified by each of the seven fuzzy rule-based classifiers. According to the classification result, each classifier voted for a single class. The final classification result by the ensemble classifier was the class with the maximum vote. When multiple classes had the same maximum vote, one of those classes was randomly chosen.

From the non-parallel non-distributed algorithm, we obtained the following results:

Average classification rate on training data: 86.31%,
Average classification rate on test data: 84.46%,
Average computation time: 658.89 minutes (10.98 hours) for a single run.

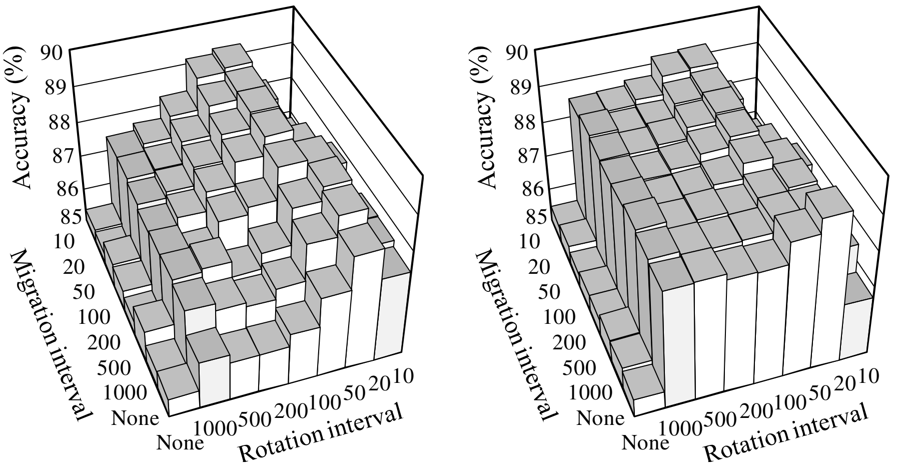
In Fig. 2, we summarize experimental results by our island model on training data. The vertical axis of each plot (i.e., the height of each bar) is the average classification rate on training data. The two axes of the base of each plot show the rotation interval and the migration interval. As shown in Fig. 2, we examined 8×8 combinations of the following specifications in our computational experiments:

Rotation interval: 10, 20, 50, 100, 200, 500, 1000, None,
Migration interval: 10, 20, 50, 100, 200, 500, 1000, None.

In these specifications, “None” means that we did not use the rotation and/or the migration. For example, when the rotation interval was 10 and the migration interval was “None” (i.e., the bottom-right bar in each plot of Fig. 2), only the training data subset rotation was performed every 10 generations.

From the comparison between the two plots in Fig. 2, we can see that better results with higher training data accuracy were obtained by ensemble classifiers in a wide range of parameter specifications in Fig. 2 (b). For example, average classification rates higher than 88% were always obtained in Fig. 2 (b) from the five specifications of the rotation interval: 50, 100, 200, 500 and 1000 (i.e., 5×8 combinations). However, such a good result was not obtained in Fig. 2 (a) when the rotation interval was specified as 500 or 1000. When the rotation interval was 1000, the fuzzy GBML algorithm was executed in each island for 1000 generations using the assigned training data subset. In this case, it is likely that each subpopulation was over-fitted to the assigned training data subset during 1000 generations. As a result, any classifiers are not likely to have high accuracy for the entire training data as shown in Fig. 2 (a).

Whereas good results were not obtained from over-fitted classifiers in the case of the rotation interval of 1000 in Fig. 2 (a), they can be good components (i.e., good base classifiers) of an ensemble classifier. This is because classifiers in different islands are likely to be over-fitted to different training data subsets. That is, a set of classifiers from different islands is likely to have high diversity. High diversity of base classifiers is essential in the design of high-performance ensemble classifiers. Actually, ensemble classifiers with high accuracy were obtained in Fig. 2 (b) by choosing the locally best classifier with respect to the assigned training data subset from each island when the rotation interval was large (e.g., 500 and 1000).



(a) Accuracy of single best classifiers. (b) Accuracy of ensemble classifiers.

Fig. 2. Experimental results of our island model on training data

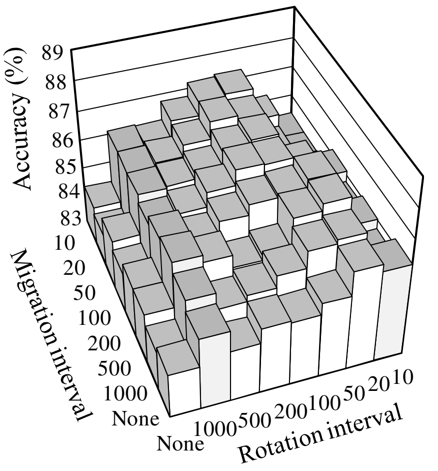
It should be noted that almost all average classification rates in Fig. 2 (a) are higher than the result 86.31% by the non-parallel non-distributed algorithm (except for the eight combinations with no rotation in Fig. 2 (a)). This observation suggests that the rotation of training data subsets has a positive effect on the search ability of the fuzzy GBML algorithm to find good classifiers. At each island, the training data subset rotation can be viewed as a periodical change of the environment. Such a periodical change seems to help the fuzzy GBML algorithm to escape from local optima. When we did not use the rotation, good results were not obtained in Fig. 2 (a).

In Fig. 2, good results were not obtained from the rotation interval of 10, either. It seems that too frequent changes of the environment had a negative effect on the search ability of the fuzzy GBML algorithm. In this case, the use of ensemble classifiers did not work well. This may be because similar classifiers were obtained from different islands due to the frequent rotation of the assigned training data subsets.

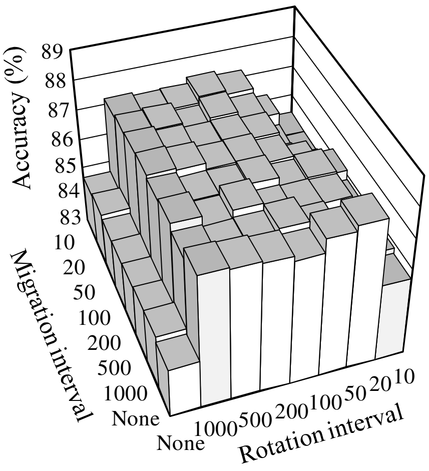
In Fig. 3, we summarize experimental results by our island model on test data. We can obtain almost the same observations from Fig. 3 as those from Fig. 2. That is, better results were obtained from ensemble classifiers in Fig. 3 (b) in a wide range of parameter specifications than single best classifiers in Fig. 3 (a). Especially when the rotation interval was large (e.g., 500 or 1000), good results were obtained from ensemble classifiers whereas the test data accuracy of single classifiers was not high.

Except for the case with no rotation, almost all classification rates in Fig. 3 are higher than 84.46% by the non-parallel non-distributed algorithm. This observation supports a positive effect of the training data subset rotation on the test data accuracy of obtained fuzzy rule-based classifiers and their ensemble classifiers.

When we used the non-parallel non-distributed algorithm, the average computation time for a single run of our fuzzy GBML algorithm was 658.89 minutes (10.98 hours). The average computation time was drastically decreased by the use of our



(a) Accuracy of single best classifiers.



(b) Accuracy of ensemble classifiers.

Fig. 3. Experimental results of our island model on test data

island model with seven CPUs for parallel computation. We summarize the average computation time of our island model in Fig. 4 where the height of each bar shows the average computation time. The average computation time was decreased from about 11 hours of the non-parallel non-distributed algorithm to about 20 minutes (about 1/33 of 11 hours) by the use of our island model. In our island model, the size of the assigned subpopulation to each CPU is 1/7 of the population size. Moreover, the size of the assigned training data subset to each CPU is 1/7 of the training data size. As a result, the computation time of our island model can be potentially decreased up to the order of 1/49 from the case of the non-parallel non-distributed algorithm.

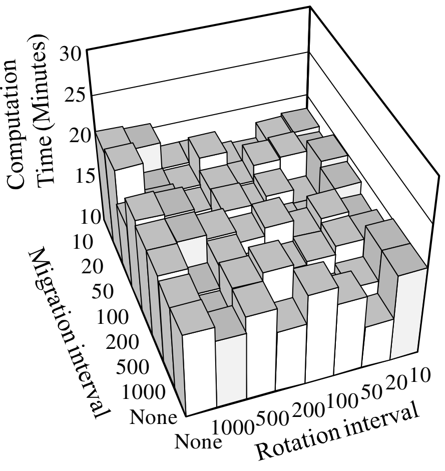


Fig. 4. Average computation time of our island model

It should be noted that there is no difference in the computation time of our island model between single best classifier design and ensemble classifier design. Both a single best classifier and an ensemble classifier are obtained from a single run of our island model. The difference between these two approaches is only the selection of classifiers from the final population after the execution of our island model.

5 Conclusions

In this paper, we proposed an idea of using an island model for ensemble classifier design. In our island model, a population of classifiers was divided into multiple subpopulations as in other island models. Training patterns were also divided into multiple training data subsets. A different training data subset was assigned to each island (i.e., each subpopulation). The assigned training data subsets were periodically rotated over the islands. After the execution of our island model, the locally best classifier was selected from each island. Through computational experiments, we demonstrated that good ensemble classifiers with high accuracy were obtained from a wide range of parameter specifications of migration and rotation intervals.

If the generalization ability maximization is the main goal in classifier design, the use of our island model for ensemble classifier design seems to be a good choice. However, if the interpretability of classifiers is also important, ensemble approaches are not recommended. This is because a set of classifiers is usually less interpretable than a single classifier. In this case, we may need a multi-objective approach to find a good accuracy-interpretability tradeoff. Parallel distributed implementation of multi-objective fuzzy GBML algorithms is an interesting future research issue.

Our island model is a general framework for parallel distributed implementation of Pittsburgh-style GBML algorithms. Implementation and performance evaluation of our island model on other GBML algorithms is an interesting future research issue.

References

1. Thrift, P.: Fuzzy Logic Synthesis with Genetic Algorithms. In: Proc. of 4th International Conference on Genetic Algorithms, pp. 509–513 (1991)
2. Karr, C.L.: Design of an Adaptive Fuzzy Logic Controller using a Genetic Algorithm. In: Proc. of 4th International Conference on Genetic Algorithms, pp. 450–457 (1991)
3. Karr, C.L., Gentry, E.J.: Fuzzy Control of pH using Genetic Algorithms. IEEE Trans. on Fuzzy Systems 1, 46–53 (1993)
4. Ishibuchi, H., Nozaki, K., Yamamoto, N., Tanaka, H.: Selecting Fuzzy If-Then Rules for Classification Problems using Genetic Algorithms. IEEE Trans. on Fuzzy Systems 3, 260–270 (1995)
5. Cordón, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten Years of Genetic Fuzzy Systems: Current Framework and New Trends. Fuzzy Sets and Systems 141, 5–31 (2004)
6. Herrera, F.: Genetic Fuzzy Systems: Status, Critical Considerations and Future Directions. International Journal of Computational Intelligence Research 1, 59–67 (2005)

7. Herrera, F.: Genetic Fuzzy Systems: Taxonomy, Current Research Trends and Prospects. Evolutionary Intelligence 1, 27–46 (2008)
8. Cordón, O.: A Historical Review of Evolutionary Learning Methods for Mamdani-Type Fuzzy Rule-Based Systems: Designing Interpretable Genetic Fuzzy Systems. International J. of Approximate Reasoning 52, 894–913 (2011)
9. Ishibuchi, H., Murata, T., Turksen, I.B.: Single-Objective and Two-Objective Genetic Algorithms for Selecting Linguistic Rules for Pattern Classification Problems. Fuzzy Sets and Systems 89, 135–150 (1997)
10. Ishibuchi, H., Nakashima, T., Murata, T.: Three-Objective Genetics-Based Machine Learning for Linguistic Rule Extraction. Information Sciences 136, 109–133 (2001)
11. Ishibuchi, H.: Multiobjective Genetic Fuzzy Systems: Review and Future Research Directions. In: Proc. of 2007 IEEE International Conference on Fuzzy Systems, pp. 913–918 (2007)
12. Fazzolari, M., Alcalá, R., Nojima, Y., Ishibuchi, H., Herrera, F.: A Review of the Application of Multi-Objective Genetic Fuzzy Systems: Current Status and Further Directions. IEEE Trans. on Fuzzy Systems (to appear)
13. Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer (2002)
14. Bull, L., Bernado-Mansilla, E., Holmes, J.: Learning Classifier Systems in Data Mining. Springer (2008)
15. García, S., Fernández, A., Luengo, J., Herrera, F.: A Study of Statistical Techniques and Performance Measures for Genetics-Based Machine Learning: Accuracy and Interpretability. Soft Computing 13, 959–977 (2009)
16. Fernández, A., García, S., Luengo, J., Bernadó-Mansilla, E., Herrera, F.: Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy, and Comparative Study. IEEE Trans. on Evolutionary Computation 14, 913–941 (2010)
17. Ishibuchi, H., Yamamoto, T., Nakashima, T.: Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems. IEEE Trans. on Systems, Man, and Cybernetics - Part B 35, 359–365 (2005)
18. Ishibuchi, H., Nojima, Y.: Analysis of Interpretability-Accuracy Tradeoff by Multiobjective Fuzzy Genetics-Based Machine Learning. International J. of Approximate Reasoning 44, 4–31 (2007)
19. Abadeh, M.S., Habibi, J., Lucas, C.: Intrusion Detection using a Fuzzy Genetics-Based Learning Algorithm. Journal of Network and Computer Applications 30, 414–428 (2007)
20. Orriols-Puig, A., Casillas, J., Bernadó-Mansilla, E.: Genetic-Based Machine Learning Systems are Competitive for Pattern Recognition. Evolutionary Intelligence 1, 209–232 (2008)
21. Nojima, Y., Ishibuchi, H., Kuwajima, I.: Parallel Distributed Genetic Fuzzy Rule Selection. Soft Computing 13, 511–519 (2009)
22. Nojima, Y., Mihara, S., Ishibuchi, H.: Parallel Distributed Implementation of Genetics-Based Machine Learning for Fuzzy Classifier Design. In: Deb, K., Bhattacharya, A., Chakraborti, N., Chakroborty, P., Das, S., Dutta, J., Gupta, S.K., Jain, A., Aggarwal, V., Branke, J., Louis, S.J., Tan, K.C. (eds.) SEAL 2010. LNCS, vol. 6457, pp. 309–318. Springer, Heidelberg (2010)
23. Ishibuchi, H., Mihara, S., Nojima, Y.: Training Data Subdivision and Periodical Rotation in Hybrid Fuzzy Genetics-Based Machine Learning. In: Proc. of 10th International Conference on Machine Learning and Applications, pp. 229–234 (2011)

24. Ishibuchi, H., Mihara, S., Nojima, Y.: Parallel Distributed Hybrid Fuzzy GBML Models with Rule Set Migration and Training Data Rotation. IEEE Trans. on Fuzzy Systems (to appear)
25. Ishibuchi, H., Nozaki, K., Tanaka, H.: Distributed Representation of Fuzzy Rules and Its Application to Pattern Classification. Fuzzy Sets and Systems 52, 21–32 (1992)
26. Cordón, O., del Jesus, M.J., Herrera, F.: A Proposal on Reasoning Methods in Fuzzy Rule-Based Classification Systems. International J. of Approximate Reasoning 20, 21–45 (1999)
27. Ishibuchi, H., Nakashima, T., Morisawa, T.: Voting in Fuzzy Rule-Based Systems for Pattern Classification Problems. Fuzzy Sets and Systems 103, 223–238 (1999)
28. Ishibuchi, H., Yamamoto, T.: Rule Weight Specification in Fuzzy Rule-Based Classification Systems. IEEE Trans. on Fuzzy Systems 13, 428–435 (2005)
29. Alba, E., Tomassini, M.: Parallelism and Evolutionary Algorithms. IEEE Trans. on Evolutionary Computation 6, 443–462 (2002)
30. Nedjah, N., Alba, E., de Macedo Mourelle, L.: Parallel Evolutionary Computations. Springer, Berlin (2006)
31. Ruciński, M., Izzo, D., Biscani, F.: On the Impact of the Migration Topology on the Island Model. Parallel Computing 36, 555–571 (2010)
32. Araujo, L., Merelo, J.: Diversity through Multiculturality: Assessing Migrant Choice Policies in an Island Model. IEEE Trans. on Evolutionary Computation 15, 456–469 (2011)
33. Luque, G., Alba, E.: Parallel Genetic Algorithms: Theory and Real World Applications. Springer, Berlin (2011)
34. Candan, C., Goëffon, A., Lardeux, F., Saubion, F.: A Dynamic Island Model for Adaptive Operator Selection. In: Proc. of 2012 Genetic and Evolutionary Computation Conference, Philadelphia, pp. 1253–1260 (2012)
35. KEEL dataset repository, `http://keel.es/`

HEMH2: An Improved Hybrid Evolutionary Metaheuristics for 0/1 Multiobjective Knapsack Problems

Ahmed Kafafy, Ahmed Bounekkar, and Stéphane Bonnevay

Laboratoire ERIC - Université Claude Bernard Lyon 1,
Ecole Polytechnique Universitaire, 15 Boulevard Latarjet,
69622 Villeurbanne cedex, France
ahmedkafafy80@gmail.com, {bounekkar,stephane.bonnevay}@univ-lyon1.fr

Abstract. Hybrid evolutionary metaheuristics tend to enhance search capabilities, by improving intensification and diversification, through incorporating different cooperative metaheuristics. In this paper, an improved version of the Hybrid Evolutionary Metaheuristics (HEMH) [7] is presented. Unlike HEMH, HEMH2 uses simple inverse greedy algorithm to construct its initial population. Then, the search efforts are directed to improve these solutions by exploring the search space using binary differential evolution. After a certain number of evaluations, path relinking is applied on high quality solutions to investigate the non-visited regions in the search space. During evaluations, the dynamic-sized neighborhood structure is adopted to shrink/extend the mating/updating range. Furthermore, the Pareto adaptive epsilon concept is used to control the archiving process with preserving the extreme solutions. HEMH2 is verified against its predecessor HEMH and the MOEA/D [13], using a set of MOKSP instances from the literature. The experimental results indicate that the HEMH2 is highly competitive and can achieve better results.

Keywords: Hybrid Metaheuristics, Adaptive Binary Differential Evolution, Path Relinking, 0/1 Multiobjective Knapsack Problems.

1 Introduction

Multiobjective combinatorial optimization problems (MOCOP) are often characterized by their large size and the presence of multiple and conflicting objectives. The basic task in multiobjective optimization is to identify the set of Pareto optimal solutions or even a good approximation to Pareto front (PF). Despite the progress in solving MOCOP exactly, the large size often means that metaheuristics are required for their solution in a reasonable time.

Solving multiobjective optimization problems (MOOP) using evolutionary algorithms (MOEAs) has been investigated by many authors [3,13,15,16]. Pareto dominance based MOEAs such as SPEA [15], NSGAII [3] and SPEA2 [16] have been dominantly used in the recent studies. Based on many traditional mathematical programming methods for approximating PF [10], the approximation of PF can be decomposed into a set of single objective subproblems. This idea is adopted in MOGLS [5] and MOEA/D [13]. Many of the search algorithms attempt to obtain the best from a set of different metaheuristics that perform together, complement each other and augment their exploration capabilities. They are commonly called *hybrid* metaheuristics. Search algorithms must balance between sometimes-conflicting two goals, intensification and diversification [1]. The design of hybrid metaheuristics can give the ability to control this balance [9].

This paper tends to improve the hybrid evolutionary metaheuristics (HEMH) proposed in [7] through developing a new version called HEMH2 with two variants HEMH$_{de}$ and HEMH$_{pr}$. The motivations of this work are to overcome the limitations from which the performance of HEMH suffers. In HEMH2, an adaptive binary differential evolution is used as a reproduction operator rather than classical crossover. Also, path relinking is applied on high quality solutions generated after a certain number of evaluations. Moreover, all improvement proposals and their effects on the search process will be discussed in details. The rest of the paper is organized as follows: section 2 presents some of the basic concepts and definitions. HEMH framework is reviewed in section 3. Section 4 explains the adaptive binary differential evolution. Path relinking strategy is discussed in section 5. The proposed HEMH2 and its variants are presented in section 6. Additionally, the experimental design and results are involved in sections 7 and 8 respectively. Finally, the conclusions and future works are involved in section 9.

2 Basic Concepts and Definitions

Without loss of generality, the MOOP can be formulated as:

$$\begin{array}{c} Maximize\, F(x) = (f_1(x), f_2(x), \cdots, f_m(x)) \\ Subject\, to:\ x \in \Omega \end{array} \tag{1}$$

where $F(x)$ is the m-dimensional objective vector, $f_i(x)$ is the i^{th} objective to be maximized, $x = (x_1, \cdots, x_n)^T$ is the n-dimensional decision vector, Ω is the feasible decision space. In the case $x \in \mathbb{Z}$, the MOOP is called MOCOP.

Definition 1. A solution x dominates y (noted as: $x \succcurlyeq y$) if: $f_i(x) \geq f_i(y) \forall i \in \{1, \cdots, m\}$ and $f_i(x) > f_i(y)$ for at least one i.

Definition 2. A solution x is said to ϵ-dominate a solution y for some $\epsilon > 0$ (noted as: $x \succcurlyeq_\epsilon y$) if and only if: $f_i(x) \geq (1+\epsilon) f_i(y)$, $\forall i \in \{1, \cdots, m\}$.

Definition 3. A solution x is called efficient (*Pareto-optimal*) if: $\nexists y \in \Omega : y \succcurlyeq x$

Definition 4. The Pareto optimal set (P^*) is the set of all efficient solutions:

$$P^* = \{x \in \Omega \,|\, \nexists y \in \Omega,\, y \succcurlyeq x\}$$

Definition 5. The Pareto front (PF) is the image of P^* in the objective space:

$$PF = \{F(x) = (f_1(x), \cdots, f_m(x)) : x \in P^*\}$$

Definition 6. Given a reference point r^* and a weight vector $\Lambda = [\lambda_1, \cdots, \lambda_m]$ such that $\lambda_i \geq 0, \forall i \in \{1, \cdots, m\}$, the *weighted sum* (F^{ws}) and the *weighted Tchebycheff* (F^{Tc}) scalarizing functions corresponding to (1) can be defined as:

$$F^{ws}(x, \Lambda) = \textstyle\sum_{i=1}^{m} \lambda_i f_i(x) \tag{2}$$

$$F^{Tc}(x, r^*, \Lambda) = Max_{1 \leq i \leq m} \{\lambda_i (r_i^* - f_i(x))\} \tag{3}$$

Definition 7. Given a set of m knapsacks and a set of n items, the 0/1 Multi-objective Knapsack Problem (MOKSP) can be written as:

$$\begin{array}{rl} Max: & f_i(x) = \sum_{j=1}^{n} c_{ij} x_j, \ \forall i \in \{1, \cdots, m\} \\ s.t.: & \sum_{j=1}^{n} w_{ij} x_j \leq W_i, \ \forall i \in \{1, \cdots, m\} \\ & x = (x_1, \cdots, x_n)^T \in \{0, 1\}^n \end{array} \tag{4}$$

where, $c_{ij} \geq 0$ is the profit of the j^{th} item in the i^{th} knapsack, $w_{ij} \geq 0$ is the weight of the j^{th} item in the i^{th} knapsack, and W_i is the capacity of the i^{th} knapsack. When x_j=1, it means that the j^{th} item is put in all knapsacks.
The MOKSP is NP-hard and can model a variety of applications.

3 HEMH, an Overview

In HEMH, a combination of different cooperative metaheuristics is provided to handle 0/1 MOKSP. The MOEA/D framework [13] is adopted to carry out the combination. The *weighted sum* defined in (2) is considered to decompose the MOKSP in (4) into a set of N single objective subproblems, based on a set of N evenly distributed weight vectors $\{\Lambda^1, \cdots, \Lambda^N\}$. HEMH attempts to simultaneously optimize these subproblems. The HEMH framework consists of the following:

- A population P of N individuals, $P = \{x^1, \cdots, x^N\}$, where x^i represents the current solution of the i^{th} subproblem.
- A set of N evenly distributed weight vectors $\{\Lambda^1, \cdots, \Lambda^N\}$, correspond to the N subproblems. Each $\Lambda = [\lambda_1, \cdots, \lambda_m]$ has m components correspond to m-objectives, such that: $\sum_{i=1}^{m} \lambda_i = 1$, $\forall \lambda_i \in \{0/H, \cdots, H/H\}$, and $H \in \mathbb{Z}^+$.
- A neighborhood B_i for each subproblem $i \in \{1, \cdots, N\}$, which includes all subproblems with the T closest weight vectors $\{\Lambda^{i1}, \cdots, \Lambda^{iT}\}$ to Λ^i.
- An *archive* to collect all efficient solutions explored over the search process.

The HEMH consists of two basic phases, *initialization* and *main loop*. In the initialization phase, an initial population of high quality solutions is constructed by applying DMGRASP on each subproblem. Then, the search efforts are concentrated on the promising regions to explore new efficient solutions. In the main loop phase, for each subproblem i, the mating/updating range M_i is chosen to be either the neighborhood B_i or the whole population P. Then, two individuals are randomly selected from M_i for reproduction. Single point crossover and mutation or greedy randomize path relinking is applied on the selected individuals to generate a new offspring, which is used to update M_i and the archive. This process is repeated until a certain number of evaluations. We refer to [7] for more details. The limitations that affect the HEMH performance are briefed as:

- Despite using DMGRASP achieves high quality initial solutions, it consumes more time and evaluations especially with large populations. Thus, the second phase will not have enough chance to improve the search process.
- Collecting all efficient solutions causes waste in time and storage space especially in many objective cases. So, archiving process should be controlled.
- For each subproblem i, the mating/updating range M_i is either the fixed (static) size neighborhood B_i or the whole population P. This may cause less execution of path relinking, or consume more time.
- Reproduction is made only by single point crossover and mutation or greedy randomized path relinking.
- Path relinking adopts single bit flipping per move and also uses local search to improve the generated solution. This causes more time consumption.

From the above, this paper presents an improved version of HEMH called HEMH2 with two variants HEMH$_{de}$ and HEMH$_{pr}$, which have the ability to overcome those limitations and can achieve an enhanced performance.

4 Adaptive Binary Differential Evolution

Differential evolution (DE) is a simple and efficient evolutionary algorithm to solve optimization problems mainly in continuous search domains [2,11]. DE's success relies on the differential mutation, that employs difference vectors built with pairs of candidate solutions in the search domain. Each difference vector is scaled and added to another candidate solution, producing the so-called mutant vector. Then, DE recombines the mutant vector with the parent solution to generate a new offspring. The offspring replaces the parent only if it has an equal or better fitness. DE has some control parameters as the mutation factor F, that used to scale the difference vectors, and the crossover rate CR. In this paper, an adaptive binary DE strategy is introduced to improve the exploration capabilities of HEMH instead of classical crossover and mutation. This strategy is described in Alg. 1. Given a population P of N individuals, where each individual represented by a n-component 0/1 vector. The main idea is to select at random three distinct individuals x^a, x^b and x^c from P for each target individual $x^i \in P$, $\forall i \in \{1, \cdots, N\}$. The mutant individual v^i is produced by applying binary differential mutation on the selected individuals according to (5). First, the *difference vector* is calculated by applying logical XOR on the two parent differential individuals x^b and x^c. Then, v^i is determined by applying logical OR on the parent based individual x^a and the difference vector previously obtained. Finally, the new generated offspring u^i is produced by applying crossover according to (6).

$$v^i = x^a + (x^b \oplus x^c) \tag{5}$$

$$u^i_j = \begin{cases} v^i_j & \text{if } rnd(j) \leq CR, \text{ or } j \in e, \forall j = 1, \cdots, n. \\ x^i_j & \text{otherwise, } \forall j = 1, \cdots, n. \end{cases} \tag{6}$$

where $rnd(j) \in [0, 1]$ is a random number generated for the j^{th} component, n is the individual length, e is a random sequence selected from the range $\{1, \cdots, n\}$

to insure that at least one component of u^i is contributed by v^i and $CR \in [0,1]$ denotes crossover rate. Here, CR is adapted periodically to avoid the premature convergence based on (7) proposed in [14].

$$CR = CR_0 \cdot e^{(-a.(G/G_{max}))} \tag{7}$$

where G and G_{max} are the current and the maximum evolutionary generations, CR_0 is the initial crossover rate. a is a constant.

Algorithm 1. ABDEvol$(x, x^a, x^b, x^c, CR_0, a)$

```
Inputs:
  $x$:                    Current solution
  $x^a, x^b, x^c$:        Parents individuals
  $CR_0 \in [0,1]$:       Crossover rate
  $a$:                    Plus constant
1: Begin:
2: $CR \leftarrow CR_0 \cdot e^{(-a.(G/G_{max}))}$;        ▷ Adapt CR
3: for all $j \in \{1, \cdots, n\}$ do:        ▷ For all items
4:     $v_j = x^a_j + (x^b_j \oplus x^c_j)$;        ▷ Binary Diff. Mutation
5:     $u_j \leftarrow \begin{cases} v_j & \text{if } rnd(j) \le CR \vee j \in e, \\ x_j & \text{otherwise.} \end{cases}$
6: end for
7: return $u$;
8: End
```

Algorithm 2. InverseGreedy(x, Λ)

```
Inputs:
  $x$:                                  Initial Solution
  $\Lambda = [\lambda_1, \cdots, \lambda_m]$:   Search direction
1: Begin: $CL \leftarrow \varnothing$;
2: for all $j \in \{1, \cdots, n\}$ do: $x_j \leftarrow 1$;        ▷ Put all
3: while $\exists j | j \notin CL \wedge Min_{j=1}^{n} \left( \frac{\sum_{i=1}^{m} \lambda_i c_{ij}}{\sum_{i=1}^{m} w_{ij}} \right)$ do:
4:     $CL \leftarrow$ Append$(j)$;
5: end while
6: while $x$ violates any constraint in (4) do:
7:     $j \leftarrow$ExtractTheFirst$(CL)$;
8:     $x \leftarrow$RemoveItem$(x, j)$;
9: end while
10: return $x$;
11: End
```

5 Path Relinking

Path relinking generates new solutions by exploring trajectories that connect high quality solutions. Starting from the starting solution x^s, path relinking builds a path in the neighborhood space that leads toward the guiding solution x^t. The relinking procedure has a better chance to investigate in more details the neighborhood of the most promising solutions if relinking starts from the best of x^s and x^t [12]. In this paper, a path relinking with two bits per move [8] is used as an intensification strategy, integrated with the adaptive binary DE. It will be invoked on the higher generations to guarantee applying the relinking process on high quality solutions. Alg. 3 describes the proposed procedure. Firstly, the best of x^s and x^t is chosen to start with. Then, the best fitness z^* and the best solution x^* are initialized. The candidate lists CL and CL_{cmp} are constructed. The procedure builds the path that connects x^s with x^t gradually by creating intermediate points through flipping two bits/move. Initially, the intermediate x is set to x^s. Then, the Hamming distance $\Delta(x, x^t)$ is calculated. The next move is carried out by flipping two of unmatched ℓ^1, ℓ^2 to be matched. If both CL and CL_{cmp} are not empty, then the first elements of CL and CL_{cmp} are extracted to be ℓ^1 and ℓ^2 respectively. Else, if one of them is empty, then, the first and the second elements of the non-empty one will be extracted to be ℓ^1 and ℓ^2 respectively. The new intermediate x is obtained by flipping the two items x_{ℓ^1} and x_{ℓ^2}. If x is infeasible, then x is repaired to get y that updates both x^* and z^*. This process is repeated until there is only two unmatched items between the current x and the guiding x^t.

Algorithm 3. PathRelinking(x^s, x^t, Λ)

```
Inputs:
  $x^s, x^t$:                              Starting and Guiding solutions
  $\Lambda = [\lambda_1, \cdots, \lambda_m]$: weight vector of the current subproblem
1: Begin:
2: $x^* \leftarrow$GetTheBestOf($x^s, x^t, \Lambda$);
3: if $x^* \neq x^s$ then: Swap($x^s, x^t$);
4: $z^* \leftarrow F^{ws}(x^*, \Lambda)$; $CL\&CL_{cmp} \leftarrow \varnothing$;
5: while $\exists j | j \notin CL \wedge x^s_j \neq x^t_j \wedge x^s_j = 0 \wedge Max^n_{j=1}\left(\sum^m_{i=1} \lambda_i c_{ij} / \sum^m_{i=1} w_{ij}\right)$ do:
6:     $CL \leftarrow$ Append($j$);
7: end while
8: while $\exists j | j \notin CL_{cmp} \wedge x^s_j \neq x^t_j \wedge x^s_j = 1 \wedge Min^n_{j=1}\left(\sum^m_{i=1} \lambda_i c_{ij} / \sum^m_{i=1} w_{ij}\right)$ do:
9:     $CL_{cmp} \leftarrow$ Append($j$);
10: end while
11: $x \leftarrow x^s$;
12: $\Delta(x, x^t) \leftarrow |\{j \in \{1, \cdots, n\} : x_j \neq x^t_j\}|$;
13: while $\Delta(x, x^t) \geq 2$ do:                    ▷ Relinking loop
14:     if $|CL| \neq 0 \wedge |CL_{cmp}| \neq 0$ then:
15:         $\ell^1 \leftarrow$ ExtractTheFirst($CL$)
16:         $\ell^2 \leftarrow$ ExtractTheFirst($CL_{cmp}$)
17:     else if $|CL| > 1$ then:
18:         $\ell^1 \leftarrow$ ExtractTheFirst($CL$)
19:         $\ell^2 \leftarrow$ ExtractTheFirst($CL$)
20:     else:
21:         $\ell^1 \leftarrow$ ExtractTheFirst($CL_{cmp}$)
22:         $\ell^2 \leftarrow$ ExtractTheFirst($CL_{cmp}$)
23:     end if
24:     $x \leftarrow$ FilppBits($x, \ell^1, \ell^2$);
25:     $y \leftarrow$Repair($x, \Lambda$)
26:     if ($F^{ws}(y, \Lambda) > z^*$) then:
27:         $x^* \leftarrow y$; $z^* \leftarrow F^{ws}(y, \Lambda)$;
28:     end if
29:     $\Delta(x, x^t) \leftarrow |\{j \in \{1, \cdots, n\} : x_j \neq x^t_j\}|$;
30: end while
31: return $x^*$;
32: End
```

Algorithm 4. HEMH2($N, T, t, \epsilon, \gamma, CR_0, a$)

```
Inputs:
  $N$:                   Population size or no. of subproblems
  $T$,:                  Min. neighborhood size
  $t$:                   Max. replaced solutions
  $\epsilon$:            Min. hamming distance
  $\gamma$:              Controls Path-relinking execution
  $CR_0 \in [0,1], a$:   Crossover rate, Plus constant
1: Begin:
2: $W_v \leftarrow \{\Lambda^1, \cdots, \Lambda^N\}$;          ▷ Set of N weight vectors
3: for $i \leftarrow 1$ to $N$ do:                        ▷ Construct Neighborhoods
4:     $B_i \leftarrow [i1, \cdots, iN]$;  ▷ where $\Lambda^{i1}, \cdots, \Lambda^{iN}$ are
5: end for                         ▷ increasingly sorted by ED to $\Lambda^i$
6: $Arch \leftarrow \varnothing$;                                     ▷ Empty archive
7: $Evl \leftarrow 0$;
8: for $i \leftarrow 1$ to $N$ do:                       ▷ Initialization phase
9:     $x^i \leftarrow$InverseGreedy($x^i, \Lambda^i$);
10:    $P \leftarrow$AddSubProblem($x^i, \Lambda^i$);
11:    $Extremes \leftarrow$Update($x^i$);Update($Evl$);
12: end for
13: while ($Evl < Mevls$) do:                                ▷ Main Loop
14:     for $i \leftarrow 1$ to $N$ do:              ▷ for each subproblem i
15:         $x^a, x^b, x^c \leftarrow$ Selection($B_i, i$);
16:                                 ▷ Where: $x^i \neq x^a \neq x^b \neq x^c$
17:         $x^j, x^k \leftarrow$ RandSelection($x^a, x^b, x^c$);
18:         $D \leftarrow \Delta(x^j, x^k)$;                ▷ Hamming distance
19:         $E \leftarrow \gamma \times Mevls$;              ▷ min. eval for PR
20:         if ($D \geq \epsilon \wedge Eval \geq E$) then:
21:             $y \leftarrow$PathRelinking($x^j$,$x^k$,$\Lambda^i$);
22:         else:
23:             $u \leftarrow$ABDEvol($x^i, x^a, x^b, x^c, CR_0, a$);
24:             $y \leftarrow$Repair($u, \Lambda^i$)
25:         end if
26:         $P \leftarrow$UpdateSolutions($y, t, B_i$);
27:         $Arch \leftarrow$ UpdateArchive$^{pa\epsilon}$($y$);
28:         $Extremes \leftarrow$Update($y$);Update($Evl$);
29:     end for
30: end while
31: $Arch \leftarrow$ AddExtremes($Extremes$);
32: return $Arch$;
33: End
```

6 The Proposed HEMH2

Motivated by the results achieved in [8], some proposals are adopted to improve HEMH performance and to overcome the limitations discussed. The main differences between HEMH2 and its predecessor are briefly presented as follows:

- Initial population is created using the simple inverse greedy algorithm (Alg.2) for each search direction rather than DMGRASP. The quality of the obtained initial solutions will be affected, but this will give a better chance to the second phase to improve and enhance the search process.
- Instead of collecting all efficient solutions, the Pareto-adaptive epsilon dominance ($pa\epsilon$-dominance) [4] is adopted to control the quality and the quantity of the efficient solutions collected in the archive.
- Dynamic neighborhood size that permit to shrink/extend the neighborhood for each subproblem is considered. Consequently, the parent solutions of a subproblem are always selected from its neighborhood. This can overcome the limitations of the binary differential mutation.

- The adaptive binary DE is used as a reproduction operator instead of crossover and mutation beside the path relinking.
- Path relinking is applied only after a certain number of evaluations as a post optimization strategy. This action guarantees the existence of high quality solutions. Moreover, path relinking flips two bits at each relinking step.
- In HEMH2, local search is avoided either after path relinking or after inverse greedy construction as proposed in HEMH.

Now, the reasons behind the above proposals are explained. Firstly, there is no doubt that generating the initial population using DMGRASP can achieve better quality solutions, but it forces us to use small populations. In some cases, local search highly consumes more time and evaluations to investigate a small specified region in the search space. Consequently, the *main loop* phase has a small chance to improve the search process. To overcome this limitation, the inverse greedy construction is proposed. From the empirical results, the inverse greedy obtains solutions as close as possible to the boundary regions than simple greedy construction. Secondly, using $pa\epsilon$-dominance will control the size of the archive, especially in many objective cases. Consequently, saving more resources of time and storage space with persevering the quality of the collected solutions. Thirdly, the poor performance of the binary differential mutation occurs when treating differential individuals with large Hamming distance. Selecting parents from the whole population can encourage this scenario. In HEMH2, parents of each subproblem are always selected from its neighborhood which has a dynamic size, this guarantees obtaining individuals with suitable Hamming distances. Fourthly, the adaptive binary DE empirically has the ability to explore the search space better than classical crossover and mutation. Thus, the performance of HEMH will be improved by adopting adaptive binary DE for reproduction rather than crossover. Finally, the proposed path relinking applies two bits flipping/move, that minimizes the whole relinking time. Also, avoiding local search saves both time and evaluations.

In Alg. 4, the HEMH2 procedure is introduced. Firstly, a set of N evenly distributed weight vectors is created. Then, the neighborhood structure is constructed for each subproblem i by assigning all subproblems sorted increasingly by the Euclidean distance between their weight vectors and the current weight vector Λ_i. After that, an initial population P is created by applying the inverse greedy (Alg.2) for each search direction. Now, the main loop is executed until achieving the maximum evaluations $Mevls$ (line 13). For each subproblem i, SELECTION routine is invoked to determine the *current size* of the neighborhood B_i such that: $|B_i| = T + r$, where T and r represent the number of different and repeated solutions in B_i respectively. This means, the SELECTION routine extends B_i size to guarantee the existence of at least T different solutions and randomly selects three of them x^a, x^b and x^c for reproduction. Two of the three selected parents x^j, x^k are chosen randomly. Then, path relinking is used only if the Hamming distance $\Delta(x^j, x^k)$ is greater than a certain value ϵ and the number of evaluations $Eval$ exceeds a certain ratio γ of the maximum evaluations $Mevls$ allowed to guarantee applying path relinking on high quality solutions.

Else, the adaptive binary DE is applied to generate a new offspring y. The new generated offspring y is evaluated and used to update the neighborhood (B_i) according to the parameter t, which controls the number of replaced solutions. The Archive is also updated by y according to $pa\epsilon$-dominance [4]. Finally, the Extreme solutions are added to the archive which is returned as an output.

In order to study the effects of both adaptive binary DE and path relinking operators distinctly, two additional algorithms variants called HEMH$_{de}$ and HEMH$_{pr}$ are considered. Both of them have the same procedure as HEMH2 explained in Alg. 4 except that HEMH$_{de}$ only adopts adaptive binary DE for reproduction. Whereas, HEMH$_{pr}$ replaces the adaptive binary DE in HEMH2 procedure by crossover and mutation.

7 Experimental Design

In this paper, both MOEA/D and HEMH are involved to verify our proposals. The comparative study for different algorithms is carried out on a set of test instances from the literature [15] listed in Table 1. All experiments are performed on a PC with Intel Core i5-2400 CPU, 3.10 GHz and 4.0 GB of RAM.

7.1 Parameter Settings

Here, the different parameters used for each algorithm are discussed. For MOEA/D and our proposals HEMH$_{de}$, HEMH$_{pr}$ and HEMH2, the parameter H which controls the population size N by the relation ($N = C_{m-1}^{H+m-1}$) is determined for each instance in Table 1 according to the complexity. The initial population used in MOEA/D is randomly generated such that each member $x = (x_1, \cdots, x_n)^T \in \{0,1\}^T$, where $x_i = 1$ with probability equal to 0.5. For HEMH, the parameter $\acute{H}$ that controls the population size $\acute{N}$ is also considered. HEMH uses the parameters α=0.1 and β=0.5. The maximum number of evaluations ($Mevls$) is used as a stopping criterion for each algorithm. For fair comparison, the same archiving strategy based on $pa\epsilon$-dominance [4] are applied to each algorithm to get the final approximation set. The $pa\epsilon$-dominance uses the archive size (A_s) listed in Table 1. HEMH applies the path relinking proposed here. Single-point crossover and standard mutation are considered. Mutation was performed for each item independently with probability $(1/n)$. The other control parameters are listed in Table 2. Finally, the statistical analysis is applied on 30 independent runs for each instance.

Table 1. Set of knapsack instances

Inst.	m	n	$N(H)$	$\acute{N}(\acute{H})$	A_s	$Mevls$
KS252	2	250	150(149)	75(74)	150	75000
KS502	2	500	200(199)	100(99)	200	100000
KS752	2	750	250(249)	125(124)	250	125000
KS253	3	250	300(23)	153(16)	200	100000
KS503	3	500	300(23)	153(16)	250	125000
KS753	3	750	300(23)	153(16)	300	150000
KS254	4	250	364(11)	165(8)	250	125000
KS504	4	500	364(11)	165(8)	300	150000
KS754	4	750	364(11)	165(8)	350	175000

Table 2. Set of common parameters

Parameters	HEMH			
	-	**-***de*	**-***pr*	**-2**
Neighborhood size: T	10	10	10	10
Max. replaced sols: t	2	2	2	2
Parents selection: δ	0.9	-	-	-
Min. support: σ	$\{1\}$	-	-	-
Ratio controls PR: γ	-	-	0.8	0.8
Min. Ham. Distance: ϵ	10	-	10	10
Crossover rate: CR_0	-	0.4	-	0.4
Plus constant: a	-	2	-	2

7.2 Assessment Metrics

Let $A, B \subset \Re^m$ be two approximations to PF, $P^*, r^* \subset \Re^m$ be a reference set and a reference point respectively. The following metrics can be expressed as:

1. **The Set Coverage.** (I_C) [15] is used to compare two approximation sets. The function I_C maps the ordered pair (A, B) to the interval $[0, 1]$ as:

$$I_C(A, B) = |\{u|u \in B, \exists v|v \in A : v \succcurlyeq u\}| / |B| \tag{8}$$

   where $I_C(A, B)$ is the percentage of the solutions in B that are dominated by at least one solution from A. $I_C(B, A)$ is not necessarily equal to 1-$I_C(A, B)$. If $I_C(A, B)$ is large and $I_C(B, A)$ is small, then A is better than B in a sense.
2. **The Hypervolume.** (I_H) [15] for a set A is defined as:

$$I_H(A) = \mathcal{L}(\cup_{u \in A} \{y|u \succcurlyeq y \succcurlyeq r^*\}) \tag{9}$$

   where $\mathcal{L}$ is the Lebesgue measure of a set. $I_H(A)$ describes the size of the objective space that is dominated by A and dominates r^*. We use the referenced indicator such that: $I_{RH}(A) = I_H(P^*) - I_H(A)$ and r^* is the origin.
3. **The Generational.** (I_{GD}) and Inverted Generational Distance (I_{IGD}) of a set A are defined as:

$$\begin{aligned} I_{GD}(A, P^*) &= \tfrac{1}{|A|} \textstyle\sum_{u \in A} \{min_{v \in P^*} d(u, v)\} \\ I_{IGD}(A, P^*) &= \tfrac{1}{|P^*|} \textstyle\sum_{u \in P^*} \{min_{v \in A} d(u, v)\} \end{aligned} \tag{10}$$

   where $d(u, v)$ is the Euclidean distance between u,v in $\Re^m$. The $I_{GD}(A, P^*)$ measures the average distance from A to the nearest solution in P^* that reflects the closeness of A to P^*. In contrast, the $I_{IGD}(A, P^*)$ measures the average distance from P^* to the nearest solution in A that reflects the spread of A to a certain degree.
4. **R-indicator.** (I_{R_3}) [6] uses a set of utility functions u, which can be any scalar function. both of weighted sum and weighted Tchebycheff functions with a sufficiently large set of evenly distributed normalized weight vectors (Λ) are used. I_{R_3} can be evaluated as follows:

$$I_{R_3}(A, P^*) = \frac{\sum_{\lambda \in \Lambda} [u^*(\lambda, P^*) - u^*(\lambda, A)] / u^*(\lambda, P^*)}{|\Lambda|} \tag{11}$$

   where $u^*(\lambda, A)$=$max_{z \in A} u(\lambda, z)$, $u(\lambda, z) = -(max_{1 \le j \le m} \lambda_j |z_j^* - z_j| + \rho \sum_{j=1}^m |z_j^* - z_j|)$, ρ is a small positive integer, and for each weight vector $\lambda \in \Lambda$, $\lambda = [\lambda_1, \cdots, \lambda_m]$ such that $\lambda_i \in [0, 1]$ and $\sum_{i=1}^m \lambda_i = 1$.

Here, the reference set P^* for each instance is formed by gathering all efficient solutions found by all algorithms in all runs. Also, all approximation sets are normalized in the range [1,2].

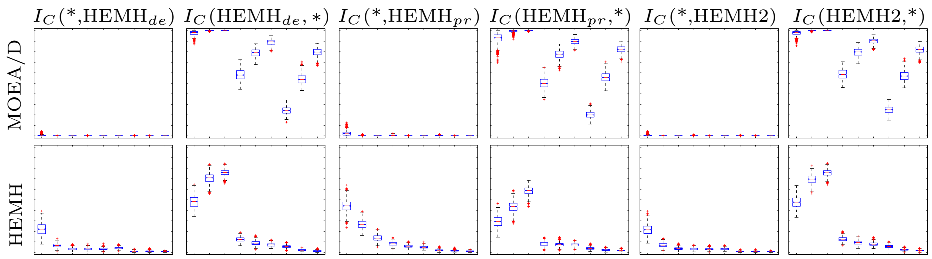


Fig. 1. Results of I_C indicator

8 Experimental Results

Here, the different simulation results are shown in details. Firstly, Fig.1 depicts the results of I_C metric. It contains a chart (with scale 0 at the bottom and 1 at the top) for the ordered pairs depicted. Each chart consists of nine box plots representing the distribution of I_C values. Each box plot (from left to right) represents an instance in Table 1 (from top to down) respectively. It is clear from the results in Fig.1 that all proposals HEMH2, HEMH$_{de}$ and HEMH$_{pr}$ outperform the original MOEA/D in all test instances. Whereas, Both HEMH2 and HEMH$_{de}$ outperform HEMH for all test instances. Also, HEMH$_{pr}$ slightly performs better than HEMH in most test instances.

Table 3. Average Referenced Hypervolume (I_{RH})

Inst.	**Algorithms**				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KP252	4.66E-02	1.02E-02	**6.07E-03**	9.33E-03	6.08E-03
KP502	5.67E-02	1.55E-02	5.57E-03	1.16E-02	**5.56E-03**
KP752	4.73E-02	1.64E-02	**3.84E-03**	7.34E-03	3.94E-03
KP253	2.24E-01	1.21E-01	8.85E-02	1.09E-01	**8.77E-02**
KP503	2.76E-01	1.16E-01	**7.39E-02**	9.01E-02	**7.39E-02**
KP753	2.89E-01	8.85E-02	6.48E-02	7.63E-02	**6.39E-02**
KP254	8.56E-01	5.55E-01	**4.26E-01**	4.90E-01	4.27E-01
KP504	1.07E+00	4.53E-01	3.96E-01	4.10E-01	**3.84E-01**
KP754	1.23E+00	3.93E-01	3.54E-01	3.64E-01	**3.41E-01**

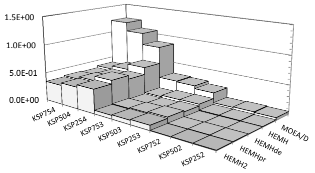


Fig. 2. Average Results

Table 4. Average Generational Distance (I_{GD})

Inst.	**Algorithms**				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KP252	1.50E-03	5.17E-04	3.40E-04	5.74E-04	**3.37E-04**
KP502	1.53E-03	4.41E-04	**1.44E-04**	3.12E-04	1.54E-04
KP752	1.33E-03	4.77E-04	**7.52E-05**	1.91E-04	7.81E-05
KP253	1.98E-03	6.83E-04	3.92E-04	6.82E-04	**3.88E-04**
KP503	2.25E-03	4.95E-04	2.76E-04	4.25E-04	**2.70E-04**
KP753	2.16E-03	3.63E-04	2.17E-04	2.77E-04	**2.07E-04**
KP254	2.52E-03	1.14E-03	**6.07E-04**	9.13E-04	6.14E-04
KP504	3.45E-03	6.63E-04	4.08E-04	5.14E-04	**3.62E-04**
KP754	3.79E-03	4.91E-04	2.88E-04	3.56E-04	**2.53E-04**

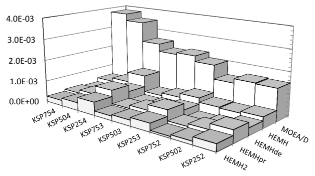


Fig. 3. Average Results

Table 5. Average Inv. Generational Dist. (I_{IGD})

Inst.	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KP252	9.32E-04	4.83E-04	**3.49E-04**	4.51E-04	3.50E-04
KP502	7.31E-04	2.72E-04	**1.31E-04**	2.10E-04	1.32E-04
KP752	5.41E-04	2.43E-04	**7.89E-05**	1.14E-04	7.95E-05
KP253	6.31E-04	4.48E-04	**3.83E-04**	4.41E-04	3.84E-04
KP503	5.28E-04	3.33E-04	**2.58E-04**	2.99E-04	2.60E-04
KP753	4.55E-04	2.46E-04	**1.93E-04**	2.24E-04	1.95E-04
KP254	6.75E-04	5.69E-04	**4.88E-04**	5.33E-04	4.91E-04
KP504	5.95E-04	4.04E-04	**3.46E-04**	3.67E-04	3.46E-04
KP754	5.46E-04	3.28E-04	2.71E-04	2.85E-04	**2.70E-04**

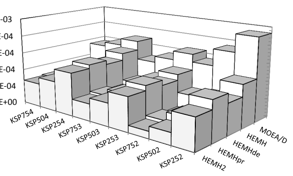


Fig. 4. Average Results

Table 6. Average R_3 indicator (I_{R_3})

Inst.	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KP252	6.05E-03	1.88E-03	**1.29E-03**	2.32E-03	1.30E-03
KP502	7.30E-03	2.06E-03	**6.89E-04**	1.62E-03	7.12E-04
KP752	6.88E-03	2.47E-03	**5.46E-04**	1.21E-03	5.54E-04
KP253	1.11E-02	5.62E-03	4.49E-03	5.24E-03	**4.48E-03**
KP503	1.34E-02	5.16E-03	3.83E-03	4.60E-03	**3.78E-03**
KP753	1.42E-02	4.25E-03	3.38E-03	3.92E-03	**3.32E-03**
KP254	1.61E-02	9.70E-03	7.86E-03	8.84E-03	**7.84E-03**
KP504	2.02E-02	7.91E-03	7.18E-03	7.40E-03	**7.06E-03**
KP754	2.39E-02	7.02E-03	6.02E-03	6.26E-03	**5.82E-03**

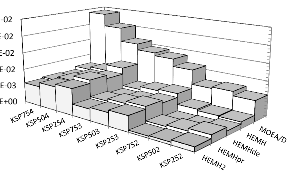


Fig. 5. Average Results

In Tables 3, 4, 5 and 6, the average values of the indicators I_{RH}, I_{GD}, I_{IGD} and I_{R_3} are listed respectively. Additionally, Figures 2, 3, 4 and 5 visualize theses values in the same order. Each of those tables contains the average values achieved over 30 independent runs for each test instance for each algorithm. Based on those results, it is clear that the proposals HEMH2, HEMH$_{de}$ and HEMH$_{pr}$ outperform the original MOEA/D and HEMH. Sience they have the minimum average values. Moreover, the proposed HEMH2 and HEMH$_{de}$ has the superiority for all test instances, followed by HEMH$_{pr}$, which confirm the results of I_C metric.

In some cases, it is observed that HEMH$_{de}$ achieves better results than HEMH2 and HEMH$_{pr}$, depending on adaptive binary DE, which reflects that the adaptive binary DE capabilities in exploring the search space is more effective than path relinking and classical crossover and mutation.

9 Conclusion

In this paper, an improved hybrid evolutionary metaheuristics HEMH2 and two other variants called HEMH$_{de}$ and HEMH$_{pr}$ were proposed to enhance HEMH performance. The HEMH2 adopts the inverse greedy procedure in its initialization phase. Both adaptive binary DE and path relinking operators are used. The HEMH$_{de}$ only uses adaptive binary DE. Whereas, HEMH$_{pr}$ uses crossover and mutation beside path relinking. The proposals were compared with the original MOEA/D and HEMH using a set of MOKSP instances from the literature. A set of quality indicators was also used to assess the performance. The experimental results indicate the superiority of all proposals over the original MOEA/D and HEMH based on the assessment indicators used in this study. According to the results, we can deduce that the adaptive binary DE included in both HEMH2 and HEMH$_{de}$ has better exploration capabilities which overcome the local search capabilities contained in the original HEMH. Therefore both of HEMH2 and HEMH$_{de}$ outperform HEMH. In some cases, HEMH$_{de}$ can achieve highly competitive results compared with HEMH2 based on the adaptive binary DE which can achieve better performance than path relinking. In the future work, the tuning parameters of HEMH2 and its variants will be investigated. Moreover, other metaheuristics will be studied to improve the performance of HEMH2 and to handle other types of combinatorial optimization problems.

References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR) 35(3), 268–308 (2003)
2. Chakraborty, U.K. (ed.): Advances in Differential Evolution. SCI, vol. 143. Springer, Berlin (2008)
3. Deb, K., Pratab, A., Agrawal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGAII. IEEE Trans. on Evolutionary Computation 6(2), 182–197 (2002)
4. Hernández-Díaz, A.G., Santana-Quintero, L.V., Coello, C.A., Luque, J.M.: Pareto-adaptive epsilon-dominance. Evolutionary Computation 15(4), 493–517 (2007)
5. Jaszkiewicz, A.: On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - A comparative experiment. IEEE Transactions on Evolutionary Computation 6(4), 402–412 (2002)
6. Knowles, J., Corne, D.: On metrics for comparing nondominated sets. In: IEEE International Conference in E-Commerce Technology, vol. 1, pp. 711–716 (2002)
7. Kafafy, A., Bounekkar, A., Bonnevay, S.: A hybrid evolutionary metaheuristics (HEMH) applied on 0/1 multiobjective knapsack problems. In: GECCO 2011, pp. 497–504 (2011)
8. Kafafy, A., Bounekkar, A., Bonnevay, S.: Hybrid metaheuristics based on MOEA/D for 0/1 multiobjective knapsack problems: A comparative study. In: IEEE World Congress on Computational Intelligence (WCCI 2012), Brisbane, June 10-15, pp. 3616–3623 (2012)
9. Lozanoa, M., García-Martínez, C.: Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. Computers and Operations Research 37(3), 481–497 (2010)
10. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
11. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization, 1st edn. Natural Computing Series. Springer (2005)
12. Ribeiro, C., Uchoa, E., Werneck, R.F.: A hybrid GRASP with perturbations for the Steiner problem in graphs. Informs Journal on Computing 14(3), 228–246 (2002)
13. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Trans. on Evolutionary Computation 11(6), 712–731 (2007)

14. Zhang, M., Zhao, S., Wang, X.: Multi-objective evolutionary algorithm based on adaptive discrete Differential Evolution. In: IEEE Congress on Evolutionary Computation (CEC 2009), pp. 614–621 (2009)
15. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto evolutionary algorithm. IEEE Transaction on Evolutionary Computation 3, 257–271 (1999)
16. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Proceedings of Evolutionary Methods for Design, Optimization and Control with Application to Industrial Problems (EUROGEN 2001), Athena, Greece, pp. 95–100 (2001)

Guided Reproduction in Differential Evolution

Prashant Singh Rana*, Harish Sharma, Mahua Bhattacharya, and Anupam Shukla

ABV-Indian Institute of Information Technology and Management, Gwalior, MP, India
{psrana,harish.sharma0107}@gmail.com
http://www.iiitm.ac.in

Abstract. Differential Evolution (DE) is a vector population based and stochastic search optimization algorithm. DE converges faster, finds the global minimum independent to initial parameters, and uses few control parameters. DE is being trapped in local optima due to its greedy updating approach and inherent differential property. In order to maintain the proper balance between exploration and exploitation in the population a novel strategy named Guided Reproduction in Differential Evolution(GRDE) algorithm is proposed. In GRDE, two new phases are introduced into classical DE; first phase enhance the diversity while second phase exploits the search space without increasing the function evaluation. With the help of experiments over 20 well known benchmark problems 3 real world optimization problems; it has been shown that GRDE outperform as compared with classical DE.

Keywords: Differential Evolution, Reproduction, Evolutionary Algorithm, Meta-heuristics.

1 Introduction

Differential Evolution(DE) was proposed by Rainer Storn and Ken Price in 1995. DE is a very popular evolutionary algorithm (EA) and exhibits good results in a wide variety of problems from diverse fields [12]. Like other EAs, DE uses mutation, crossover, and selection operators at each generation to move its population towards the global optimum. The DE performance mainly depends on two components, one is its trial vector generation strategy (i.e. mutation and crossover operators), and the other is its control parameters (i.e. population size NP, scaling factor F, and crossover rate CR). When we use DE to solve optimization problems, we first determine the trial vector and then tune the control parameters by a trial-and-error procedure.

Researchers are continuously working to improve the performance of DE. Some of the recently developed versions of DE with appropriate applications can be found in [1]. Experiments over several numerical benchmarks [15] show that DE performs better than the genetic algorithm (GA) [6] and particle swarm optimization (PSO) [7]. DE has successfully been applied to various areas of science and

* Corresponding author.

technology, such as chemical engineering [9], signal processing [2], mechanical engineering design [14], machine intelligence, and pattern recognition [11].

DE also outperforms in multi-model and constrained problems. These results indicate that DE has the advantage of fast convergence and low computational consumption of function evaluations. Though DE performs well in many fields, its fast convergence property usually leads the optimization process to be trapped by a local optimum [8, 10, 13]. This disadvantage comes from the two aspects of DE: greedy updating method and intrinsic differential property. The greedy updating strategy and differential operation result a premature convergence in DE [8, 10]. To overcome the aforementioned drawback and to find a trade-off between exploration and exploitation capability of DE, a new strategy is proposed named Guided Reproduction in Differential Evolution (GRDE).

Rest of the paper is organized as follows: Section 2 describes brief overview of classical Differential Evolution algorithm. Section 3 describe the detail working of GRDE. In section 4 the proposed strategy is tested over benchmark and real world optimization problems and statistical tests are carried out. Finally, in section 5 paper is concluded.

2 Classical Differential Evolution Algorithm

In DE, initially the population is generated randomly with uniform distribution followed by mutation, crossover and selection operation to generate a new population. The generation of trial vector is a crucial step in DE process. The two operators mutation and crossover are used to generate the trial vectors. The selection operator is used to select the best trial vector for the next generation. DE operators are explained briefly in following subsections.

Mutation: A trial vector is generated by the DE mutation operator for each individual of the current population. For generating the trial vector, a target vector is mutated with a weighted differential. An offspring is produced in the crossover operation using the newly generated trial vector. If G is the index for generation counter, the mutation operator for generating a trial vector $u_i(G)$ from the parent vector $x_i(G)$ is defined as follows:

- Select a target vector, $x_{i1}(G)$, from the population, such that $i \neq i1$.
- Again, randomly select two individuals, x_{i2} and x_{i3}, from the population such that $i \neq i1 \neq i2 \neq i3$.
- Then the target vector is mutated for calculating the trial vector as follows:

$$u_i(G) = x_{i1}(G) + \underbrace{F \times \overbrace{(x_{i2}(G) - x_{i3}(G))}^{\text{Variation Component}}}_{\text{Step size}} \tag{1}$$

where $F \in (0, 1)$ is the mutation scale factor which is used in controlling the amplification of the differential variation [5].

Crossover: Offspring $x'_i(G)$ is generated using the crossover of parent vector, $x_i(G)$ and the trial vector, $u_i(G)$ as follows:

$$x'_{ij}(G) = \begin{cases} u_{ij}(G), & \text{if } j \in J \\ x_{ij}(G), & \text{otherwise.} \end{cases} \tag{2}$$

J is the set of crossover points or the points that will go under perturbation, $x_{ij}(G)$ is the j^{th} element of the vector $x_i(G)$.

Selection: There are two functions of the selection operator, first it select the individual for the mutation operation to generate the trial vector and second, it selects the best between the parent and the offspring based on their fitness value for the next generation. If the fitness of the offspring is better than the parent than it replaces the parent else parent remain in the population.

$$x_i(G+1) = \begin{cases} x'_i(G), & \text{if } f(x'_i(G)) > f(x_i(G)). \\ x_i(G), & \text{otherwise.} \end{cases} \tag{3}$$

This ensures that the population's average fitness does not deteriorate. The pseudo code for classical DE strategy is described in Algorithm 1 [5], where F is the scale factor, CR is the crossover rate and P is the population vector.

Algorithm 1. Classical Differential Evolution

```
Initialize the control parameters;
Initialize the population;
while stopping condition(s) not true do
    for each individual $x_i(G) \in P(G)$ do
        (i) Evaluate the fitness, $f(x_i(G))$;
        (ii) Create the trial vector, $u_i(G)$ by applying the mutation operator;
        (iii) Create an offspring, $x'_i(G)$, by applying the crossover operator;
        if $f(x'_i(G))$ is better than $f(x_i(G))$ then
            Add $x'_i(G)$ to $P(G+1)$;
        else
            Add $x_i(G)$ to $P(G+1)$;
        end if
    end for
end while
Return the individual with the best fitness as the solution;
```

3 GRDE: Guided Reproduction in Differential Evaluation

To overcome the drawbacks of classical DE, the two new phases are introduced that maintain the proper balance between exploration and exploitation and maintain the diversity in the population. The Figure 1 describes the methodology of GRDE. During experiments, it is observed that classical DE solves the

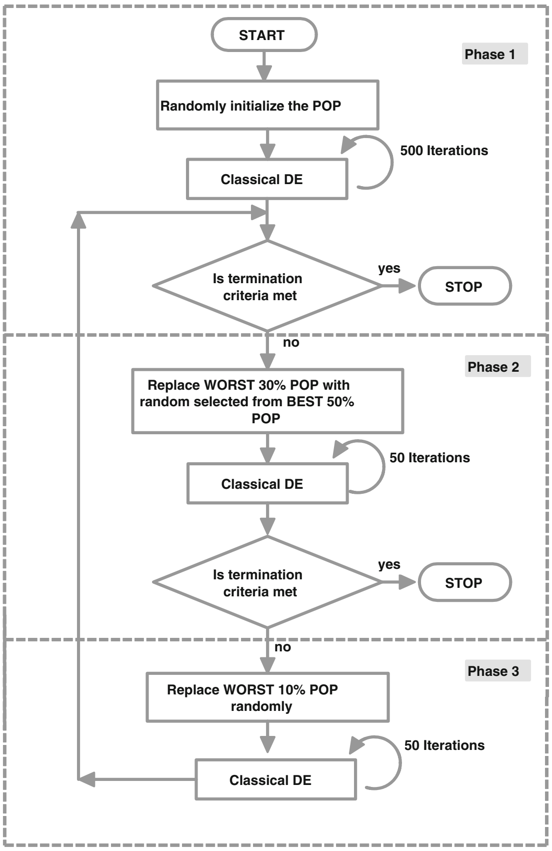

START
Phase 1
Randomly initialize the POP
500 Iterations
Classical DE
Is termination criteria met
yes
STOP
no
Phase 2
Replace WORST 30% POP with random selected from BEST 50% POP
50 Iterations
Classical DE
Is termination criteria met
yes
STOP
no
Phase 3
Replace WORST 10% POP randomly
50 Iterations
Classical DE

Fig. 1. Guided Reproduction in DE

most of the benchmark problems very quickly or unable to solve. GRDE trying to utilize the fast solving capability of DE. In the proposed strategy two new phases are introduced namely phase 2 and phase 3.

In the phase 1, the classical DE is executed for finding the required result of the given problem. Therefore phase 1 is executed for 500 iterations that explore and exploit the search space. If the required results are not obtained in the phase 1 then it enters into the phase 2. Both Phase 2 and Phase 3 are run for 50 iterations and consider as learning period. In Phase 2, worst 30% population is replaced by using the best 50% population. Every dimension of new individual is selected from randomly selected individuals corresponding dimensions from the best 50%population. This strategy is considered as guided reproduction of the population. The new individuals are reproduced using the best population, therefore the new population exploits the search space around the best populations. Further, as shown in Figure 1, the DE algorithm is executed for next 50 iterations and checked that the objective is achieved or not.

Phase 3 maintains the exploration in which the worst 10% population is replaced by randomly initialized population in the search space. This phase is used to maintain the diversity in the population and to get-ride of stagnation problem. After it, the DE algorithm is again executed for further 50 iteration and checked the objective value. The phase 1 and 2 is repeatedly executed till the termination criteria is not satisfied, therefore it is expected that these two phases maintain the balance between exploration and exploitation, without increasing the function evaluation. The percentage of population selected for reproduction and randomly initialization are adopted based on the empirical study. There may be other strategies also for selection of the percentage of population for regeneration and randomly initialization.

The pseudo-code of the proposed strategy is described in algorithm 2. The phase 2 and phase 3 are explained in Algorithms 3 and 4 respectively.

Algorithm 2. GRDE: Guided Reproduction in Differential Evaluation

```
Phase 1: Randomly initialize the Population.
    Step 1: Run Classical DE for 500 iterations, STOP if termination criteria met.
Phase 2: For exploitation.
    Step 2: Replace WORST 30% populations (dimension wise) by randomly
            selected individual from BEST 50% Population.
    Step 3: Run Classical DE for 50 iterations, STOP if termination criteria met.
Phase 3: For adding diversity in Population for exploration.
    Step 4: Replace WORST 10% Population randomly.
    Step 5: Run Classical DE for 50 iterations, STOP if termination criteria met.
    Step 6: Repeat STEP 2 to STEP 6 until termination criteria met.
END
```

Algorithm 3. GRDE: Phase 2

```
STEP1: Select BEST 50% individuals from current population;
STEP2: Generate 30% new Population as follows:
           N=(Population Size / 100) * 30
           for every i in [1,2,.....N]:
              for every j in [1,2,.....DIM]:
                 select an individual r from BEST 50% population;
                 $x_{ij} = x_{rj}$
STEP3: Replace newly generated Population with WORST 30% population.
```

Algorithm 4. GRDE: Phase 3

```
STEP1: Generate 10% new random population as follows:
           M=(Population Size / 100) * 10
           for every i in [1,2,.....M]:
              for every j in [1,2,.....DIM]:
                 $x_{ij} = rand(LB[i][j], UB[i][j])$
STEP2: Replace newly generated Population with WORST 10% population.
```

4 Experiment Results and Discussion

In this section, DE and GRDE are compared in terms of performance, reliability and accuracy. To test DE and GRDE over benchmark and real world problems following experimental setting is adopted:

- The crossover probability CR=0.9,
- The scale factor which controls the implication of the differential variation F=0.5; [12],
- Population size NP=100,
- The stopping criteria is either maximum number of function evaluation which is set to be 3×10^5 is reached or the required error is achieved. Respective acceptable error for each test problem is mentioned in Table 1,
- The number of simulations =100,

The algorithms (DE and GRDE) tested over 20 ($f_1 - f_{20}$) benchmark problems mentioned in Table 1.

Table 1. Test problems

Objective Function	**Search Space**	**Optimum Value**	**D**	**Acceptable Error**
$f_1=\sum_{i=1}^{D} i.(x_i)^4$	[-5.12, 5.12]	0	30	1.0E-05
$f_2=1+\frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})$	[-600, 600]	0	30	1.0E-05
$f_3=\sum_{i=1}^{D}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	[-30, 30]	0	30	1.0E-02
$f_4=10D+\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)]$	[-5.12, 5.12]	0	30	1.0E-05
$f_5=-20+e+exp(-\frac{0.2}{D}\sqrt{\sum_{i=1}^{D} {x_i}^3})$ $-exp(\frac{1}{D}\sum_{i=1}^{D}\cos\ (2\pi.x_i)x_i)$	[-1, 1]	0	30	1.0E-05
$f_6=-\sum_{i=1}^{D}\sin\ x_i(\sin(\frac{i.{x_i}^2}{\pi})^{20})$	[0, π]	-9.66015	10	1.0E-05
$f_7=1-cos(2\pi\sqrt{\sum_{i=1}^{D} x_i^2})+0.1(\sqrt{\sum_{i=1}^{D} x_i^2})$	[-100, 100]	0	30	1.0E-01
$f_8=-\sum_{i=1}^{D-1}\left(\exp\left(\frac{-(x_i^2+x_{i+1}^2+0.5x_ix_{i+1})}{8}\right)\times \text{I}\right)$ $where, I=cos\left(4\sqrt{x_i^2+x_{i+1}^2+0.5x_ix_{i+1}}\right)$	[-5, 5]	-D+1	10	1.0E-05
$f_9=a(x_2-bx_1^2+cx_1-d)^2+e(1-f)\cos x_1+e$ $where, a=1, b=\frac{5.1}{4\Pi^2}, c=\frac{5}{\Pi}, d=6, e=10$	x$_1$=[-5, 10], x$_2$=[0, 15]	$\frac{1}{8\Pi}$	2	1.0E-05
$f_{10}=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	[-5, 5]	0.000307486	4	1.0E-05
$f_{11}=\sum_{i=1}^{D} z_i^2+f_{bias}$, $where, z=(x-o), x=[x_1,...x_D], o=[o_1,...o_D]$	[-100, 100]	450	10	1.0E-05
$f_{12}=\sum_{i=1}^{D}(z_i^2-10\cos(2\pi z_i)+10)$ $where, z=(x-o), x=[x_1,...x_D], o=[o_1,...o_D]$	[-5, 5]	-330	10	1.0E-02
$f_{13}=\sum_{i=1}^{D}(\sum_{j=1}^{i} z_j)^2$ $where, z=(x-o), x=[x_1,...x_D], o=[o_1,...o_D]$	[-100, 100]	-450	10	1.0E-05
$f_{14}=\sum_{i=1}^{D}\frac{z_i^2}{4000}-\prod_{i=1}^{D}\cos(\frac{z_i}{\sqrt{i}})+1$ $where, z=(x-o), x=[x_1,...x_D], o=[o_1,...o_D]$	[-600, 600]	-180	10	1.0E-05
$f_{15}=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i))+20+e$ $where, z=(x-o), x=(x_1,...x_D), o=(o_1,...o_D)$	[-32, 32]	-140	10	1.0E-05
$f_{16}=\sum_{i=1}^{D-1}(100(z_i^2-z_{i+1})^2+(z_i-1)^2)$ $where, z=(x-o+1), x=[x_1,....x_D], o=[o_1,...o_D]$	[-100, 100]	390	10	1.0E-01
$f_{17}=(1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2))\cdot$ $(30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2))$	[-2, 2]	3	2	1.0E-14
$f_{18}=(1-8x_1+7x_1^2-7/3x_1^3+1/4x_1^4)x_2^2\exp(-x_2)$	x$_1$=[0, 5] x$_2$=[0, 6]	-2.3458	2	1.0E-06

Table 1. (*continued*)

Objective Function	**Search Space**	**Optimum Value**	**D**	**Acceptable Error**
$f_{19}{=}{-}[A\prod_{i=1}^{n}\sin(x_i-z)+\prod_{i=1}^{n}sin(B(x_i-z))]$	[0, 180]	-3.5	10	1.0E-02
$where, A=2.5, B=5, z=30$				
$f_{20}{=}\sum_{i=1}^{D}(5ix_i^2)$	[-5.12, 5.12]	0	30	1.0E-15
Pressure vessel confinement method [16]	x_1=[1.1, 12.5]			
$f_{21}{=}0.6224x_1x_3x_4+1.7781x_2x_3^2+3.1661x_1^2x_4+19.84x_1^2x_3$	x_2=[0.6, 12.5]	0	6	1.0E-05
	x_3=[0, 240]			
	x_4=[0, 240]			
Lennard jones [3]				
$f_{22}{:}p_i=(x_i,y_i,z_i)\}; i=1,...D,$	[-2, 2]	-9.103852	3	1.0E-03
$V_D(p)=\sum_{i=1}^{D-1}\sum_{j=i+1}^{D}(r_{ij}^{-12}-2.r_{ij}^{-6})$				
Parameter Estimation for Frequency-Modulated [3](FM)				
$f_{23}{=}\sum_{t=0}^{100}(y(t)-y_\circ(t))^2$	[-6.4,6.35]	0	6	1.0E-05

To test the robustness of the proposed strategy, it is tested over 3 real world optimization problems ($f_{21}-f_{23}$) namely Pressure vessel confinement method [16], Lennard Jones [3] and Parameter Estimation [3] for Frequency-Modulated (FM) mentioned in Table 1. Numerical results are given in Table 2. In Table 2, standard deviation (SD), mean error (ME), average function evaluations (AFE) and success rate (SR) are reported. Table 2 shows that most of the time inclusion of guided reproduction in DE improves the reliability, efficiency and accuracy. Performance index statistical analysis has been carried out for results of classical DE and $GRDE$.

In order to compare the consolidated performance of DE with $GRDE$, the value of a performance index PI [4] is computed. This index gives a weighted importance to the success rate, the mean error as well as the average number of function evaluations. Assume that k_1, k_2 and k_3 ($k_1+k_2+k_3=1$ and $k_1,k_2,k_3\leq 1$) are the weights assigned to success rate, average number of function evaluations and mean error respectively. The resultant cases are as follows:

1. $k_1=W, k_2=k_3=\frac{1-W}{2}, 0\leq W\leq 1;$
2. $k_2=W, k_1=k_3=\frac{1-W}{2}, 0\leq W\leq 1;$
3. $k_3=W, k_1=k_2=\frac{1-W}{2}, 0\leq W\leq 1;$

Table 2. Comparison of the results of test problems

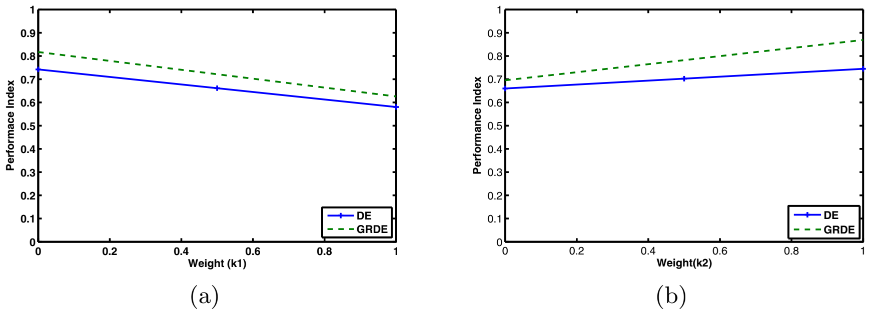
Test Problems	**DE**				**GRDE**			
	SD	**ME**	**AFE**	**SR**	**SD**	**ME**	**AFE**	**SR**
f_1	$7.76E-07$	$3.88E-07$	53456	100	$1.02E-06$	$5.10E-07$	53054.8	100
f_2	0.00158259	$7.91E-04$	85200	96	0.000979905	$4.90E-04$	77827.7	99
f_3	0.132631827	$6.63E-02$	295093	33	0.625728863	$3.13E-01$	294896.9	33

Table 2. (*continued*)

Test Problems	DE				GRDE			
	SD	**ME**	**AFE**	**SR**	**SD**	**ME**	**AFE**	**SR**
f_4	22.65206375	$1.13E+01$	300000	0	5.206093525	$2.60E+00$	300030	0
f_5	$4.59E-07$	$2.30E-07$	102831	100	$5.99E-07$	$2.99E-07$	101005.2	100
f_6	0.020436685	$1.02E-02$	235895	56	0.019530448	$9.77E-03$	180142.9	66
f_7	0.034188442	$1.71E-02$	299219	4	0.037919187	$1.90E-02$	296240.7	6
f_8	0.414863257	$2.07E-01$	234605	68	0.320114802	$1.60E-02$	220626.4	72
f_9	$7.20E-06$	$4.03E-06$	62779	80	$6.04E-06$	$3.55E-06$	36185.3	89
f_{10}	0.000333271	$1.67E-04$	55535	84	0.000272671	$1.36E-04$	38104	90
f_{11}	$1.36E-06$	$6.80E-07$	18838	100	$1.59E-06$	$7.95E-07$	18743	100
f_{12}	15.35514103	$7.68E+00$	300000	0	11.77034447	$5.89E+00$	300030	0
f_{13}	5869.778866	$2.93E+03$	300000	0	3987.106069	$1.99E+03$	300030	0
f_{14}	0.009648219	$4.82E-03$	225114	68	0.003092198	$1.55E-03$	166697.6	90
f_{15}	$9.40E-07$	$4.71E-07$	28596	100	$7.98E-07$	$3.99E-07$	28467	100
f_{16}	0.006709735	$3.35E-03$	44458	100	0.006066271	$3.03E-03$	44998.9	100
f_{17}	$4.78E-14$	$2.28E-14$	158867	48	$4.81E-14$	$2.28E-14$	153012	50
f_{18}	$2.51E-06$	$1.21E-06$	285081	5	$3.83E-06$	$1.88E-06$	261214.1	13
f_{19}	0.158572269	$7.93E-02$	300000	0	0.214693134	$1.07E-01$	292254.8	23
f_{20}	$8.25E-17$	$4.13E-17$	141010	100	$1.03E-16$	$5.15E-17$	138050.9	100
f_{21}	$3.69E-05$	$1.68E-05$	150990	53	$3.50E-05$	$1.62E-05$	114412.2	66
f_{22}	0.576294693	$2.88E-01$	275671	61	0.625329126	$3.13E-01$	274588.7	61
f_{23}	4.991971307	$2.50E-00$	104837	79	4.646551167	$2.32E-00$	98862.7	81

The graphs corresponding to each of the cases (1), (2) and (3) are shown in Figure 2(a), 2(b), and 2(c) respectively. In these figures the horizontal axis represents the weight W and the vertical axis represents the performance index PI. In case (1), average number of function evaluations and the mean error are given equal weights. PIs of DE and $GRDE$ are superimposed in the Figure 2(a) for comparison of the performance. It is observed that for $GRDE$, the value of PI is more than the classical DE. In case (2), equal weights are assigned to the success rate and average function evaluations and in case (3), equal weights are assigned to the success rate and the mean error. It is clear from Figure 2(b)and Figure 2(c) that the algorithms perform same as in case (1). Therefore, it is concluded that the $GRDE$ is better than classical DE.

DE and $GRDE$ are also compared through SR, ME and AFE. First SR is compared for both the algorithms and if it is not possible to distinguish the algorithms based on SR then comparison is made on the basis of ME. AFE is used for comparison if it is not possible on the basis of SR and ME both. On the basis of this comparison, results of Table 2 is analyzed and Table 3 is constructed. In Table 3, '+' indicates that the goodness of GRDE and '-' indicate that there is no significant improvement of GRDE over classical DE. It is observed from Table 3 that the GRDE outperforms for 15 problems compared to the classical DE.



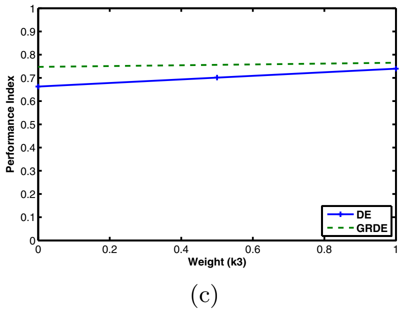


Fig. 2. Performance index; (a) for case (1), (b) for case (2) and (c) for case (3)

Table 3. Results of the comparison

Test Problem	GRDE	Test Problem	GRDE
f_1	-	f_{13}	+
f_2	+	f_{14}	+
f_3	-	f_{15}	-
f_4	+	f_{16}	-
f_5	-	f_{17}	+
f_6	+	f_{18}	+
f_7	+	f_{19}	+
f_8	+	f_{20}	-
f_9	+	f_{21}	+
f_{10}	+	f_{22}	-
f_{11}	-	f_{23}	+
f_{12}	+		

5 Conclusion

In this paper, premature phenomenon of conventional differential evolution algorithm is analyzed and illustrated with numerical results. A novel Guided Reproduction in Differential Evolution (GRDE) strategy is proposed to improve the global search ability of DE. With the aid of guided reproduction method, GRDE is able to achieve the balance between exploration and exploitation. Numerical results of 20 well known bench mark problems and 3 real world problems prove that the GRDE outperforms the DE in most of the cases.

Future directions of the research: The proposed strategy is an example only and there may be various strategies for guided reproduction of the population. Therefore, one required to find a best one strategy which is the best enough to explores and exploits the search space.

References

[1] Chakraborty, U.K.: Advances in differential evolution. Springer (2008)
[2] Das, S., Konar, A.: Two-dimensional iir filter design with modern search heuristics: A comparative study. International Journal of Computational Intelligence and Applications 6(3), 329–355 (2006)
[3] Das, S., Suganthan, P.N.: Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Department of Electronics and Telecom-munication Engineering. Technical Report (2011)
[4] Thakur, M., Deep, K.: A new crossover operator for real coded genetic algorithms. Applied Mathematics and Computation 188(1), 895–911 (2007)
[5] Engelbrecht, A.P.: Computational intelligence: an introduction. Wiley (2007)
[6] Holland, J.H.: Adaptation in natural and artificial systems, vol. (53). University of Michigan Press (1975)
[7] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
[8] Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: Proceedings of MENDEL, pp. 76–83. Citeseer (2000)
[9] Liu, P.K., Wang, F.S.: Inverse problems of biological systems using multi-objective optimization. Journal of the Chinese Institute of Chemical Engineers 39(5), 399–406 (2008)
[10] Mezura-Montes, E., Velázquez-Reyes, J., Coello Coello, C.A.: A comparative study of differential evolution variants for global optimization. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 485–492. ACM (2006)
[11] Omran, M.G.H., Engelbrecht, A.P., Salman, A.: Differential evolution methods for unsupervised image classification. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 966–973. IEEE (2005)
[12] Price, K.V.: Differential evolution: a fast and simple numerical optimizer. In: 1996 Biennial Conference of the North American Fuzzy Information Processing Society, NAFIPS, pp. 524–527. IEEE (1996)

[13] Price, K.V., Storn, R.M., Lampinen, J.A.: Differential evolution: a practical approach to global optimization. Springer (2005)
[14] Rogalsky, T., Kocabiyik, S., Derksen, R.W.: Differential evolution in aerodynamic optimization. Canadian Aeronautics and Space Journal 46(4), 183–190 (2000)
[15] Vesterstrom, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Congress on Evolutionary Computation, CEC 2004, vol. 2, pp. 1980–1987. IEEE (2004)
[16] Wang, X., Gao, X.Z., Ovaska, S.J.: A simulated annealing-based immune optimization method. In: Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning, Porvoo, Finland, pp. 41–47 (2008)

A Study of Breakout Local Search for the Minimum Sum Coloring Problem

Una Benlic and Jin-Kao Hao

LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers Cedex 01, France
{benlic,hao}@info.univ-angers.fr

Abstract. Given an undirected graph $G = (V, E)$, the minimum sum coloring problem (MSCP) is to find a legal assignment of colors (represented by natural numbers) to each vertex of G such that the total sum of the colors assigned to the vertices is minimized. In this paper, we present Breakout Local Search (BLS) for MSCP which combines some essential features of several well-established metaheuristics. BLS explores the search space by a joint use of local search and adaptive perturbation strategies. Tested on 27 commonly used benchmark instances, our algorithm shows competitive performance with respect to recently proposed heuristics and is able to find new record-breaking results for 4 instances.

Keywords: minimum sum coloring, adaptive perturbation strategy, heuristic, combinatorial optimization.

1 Introduction

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E, a legal k-coloring of G is a mapping $C : V \to \{1, ..., k\}$ such that $\forall\{v, u\} \in E, C(v) \neq C(u)$, i.e., no two adjacent vertices are assigned the same color label. A legal k-coloring C can also be represented as a partition of vertex set V into k mutually disjoint independent sets (also called color classes) $\{S_1, ..., S_k\}$ such that $\bigcup_i S_i = V$ and $v \in S_i$ if $C(v) = i$ (v receives color label i). The well-known NP-hard *graph coloring problem* (GCP) is to determine a legal k-coloring with the minimum value k. A related problem to the GCP is *the minimum sum coloring problem* (MSCP) which consists in finding a legal coloring $C = \{S_1, ..., S_k\}$ such that the following total sum of color labels is minimized:

$$Sum(C) = \sum_{i=1}^{k} i \cdot |S_i| \tag{1}$$

where $|S_1| \geq ... \geq |S_k|$ and $|S_i|$ is the size of S_i (i.e., number of vertices in S_i).

The MSCP is known to be NP-hard with several practical applications including VLSI design, scheduling, and distributed resource allocation (see [12] for a list of references). Over the past two decades, it has been studied mainly from a theoretical point of view. Only recently, several heuristics have been proposed for the practical solving of the general MSCP [3,4,6,9,10,13].

In this work, we introduce the Breakout Local Search (BLS) for the MSCP. BLS follows the basic scheme of iterated local search [11] and combines features from other well-known methods including tabu search [5] and simulated annealing [8]. The basic idea of BLS is to use descent-based local search to discover local optima and employ adaptive perturbations to continually move from one attractor to another in the search space. Based on the information on the state of search, the perturbation strategy of BLS introduces a varying degree of diversification by dynamically determining the number of moves for perturbation and by adaptively selecting between several types of dedicated moves.

We evaluate the performance of BLS on 27 benchmark graphs which are commonly used in the literature to test MSCP algorithms. Despite its simplicity, BLS shows competitive results with respect to the most recent MSCP heuristics.

2 Breakout Local Search (BLS) for the MSCP

2.1 General BLS Procedure

Recall that a local optimum with respect to a given neighborhood N is a solution s^* such that $\forall s \in N(s^*)$, $f(s^*) \leq f(s)$, where f is the objective function to be minimized. A basin of attraction of a local optimum l is the set B_l of solutions that lead the local search to the given local optimum l, i.e., $B_l = \{s \in S | LocalSearch(s) = l\}$. Since a local optimum l acts as an attractor with respect to the solutions B_l, the terms attractor and local optimum will be used interchangeably throughout this work.

Basically, our Breakout Local Search (BLS) approach moves from one basin of attraction formed by a local optimum to another basin by applying a smaller or larger number of perturbation moves whose type (e.g., random or directed moves) is determined adaptively. BLS starts from an initial solution C_0 and applies to it local search (descent) to reach a local optimum or an attractor C. Each iteration of the local search algorithm scans the whole neighborhood and selects the best improving neighboring solution to replace the current solution. If no improving neighbor exists, local optimality is reached. At this point, BLS tries to escape from the basin of attraction of the current local optimum and move into a neighboring basin of attraction. For this purpose, BLS applies a number of dedicated moves to the current optimum C (we say that C is perturbed). Each time an attractor is perturbed, the perturbed solution is used as the new starting point for the next round of the local search procedure.

If the search returns to the last attractor C, BLS perturbs C more strongly by increasing the number of moves to be applied for perturbation. After visiting a certain number of local optima without improving the best solution found so far, BLS applies a significantly stronger perturbation in order to drive definitively the search toward a new and more distant region in the search space.

The success of the described method depends crucially on two factors. First, it is important to determine the number L of perturbation moves (also called "jump magnitude") to be applied to change or perturb the solution. Second, it is equally important to consider the type of perturbation moves to be applied. While conventional perturbations are often based on random moves, more focused perturbations using dedicated information could be more effective. The degree of diversification, introduced by a perturbation mechanism, depends both on the jump magnitude and the type of moves used for perturbation. A weak diversification induces a high probability for the local search procedure to cycle between two or more locally optimal solutions, leading to search stagnation. On the other hand, a too strong diversification will have the same effect as a random restart, which usually results in a low probability of finding better solutions in the following local search phases.

Algorithm 1. Breakout Local Search

```
Require: Initial jump magnitude $L_0$, jump magnitude $L_s$ for strongest perturbation, max. number
    $T$ of non-improving attractors visited before strong perturb.
Ensure: A solution $C_{best}$.
1: $GenerateInitialSolution(C)$
2: $C_{best} \leftarrow C$    /* $C_{best}$ records the best solution found so far */
3: $C_p \leftarrow C$    /* $C_p$ records the last local optimum */
4: $\omega \leftarrow 0$    /* Set counter for consecutive non-improving local optima */
5: while stopping condition not reached do
6:     if Solution $C$ is not legal then
7:         $C \leftarrow LocalSearch2(C)$
8:     end if
9:     $C \leftarrow LocalSearch1(C)$
10:    if ($Sum(C) < Sum(C_{best})$) and ($C$ is a legal solution) then
11:        $C_{best} \leftarrow C$;    /* Update the best solution found so far */
12:        $\omega \leftarrow 0$  /* Reset the counter of consecutive non-improving local optima */
13:    else
14:        $\omega \leftarrow \omega + 1$
15:    end if
16:    /* Determine the perturbation strength $L$ to be applied to $C$ */
17:    if $\omega > T$ then
18:        /* Search is stagnating, strongest perturbation required */
19:        $L \leftarrow L_s$
20:    else if $C = C_p$ then
21:        /* Search returned to previous local optimum, increment perturb. strength */
22:        $L \leftarrow L + 1$
23:    else
24:        /* Search escaped from previous local optimum, reinitialize perturb. strength */
25:        $L \leftarrow L_0$
26:    end if
27:    /* Perturb the current local optimum $C$ with $L$ perturbation moves */
28:    $C_p \leftarrow C$
29:    $C \leftarrow Perturbation(C, L)$
30: end while
```

Algorithm 1 presents the general BLS algorithm for the MSCP, whose ingredients are detailed in the following sections. BLS starts from an initial random solution C which may be a conflicting coloring (line 1). To reach a legal coloring, BLS employs two different local search procedures (lines 6-9, see next section), based on different move operators and evaluation functions. Once a local optimum is reached, the jump magnitude L is determined depending on whether the search escaped or returned to the previous local optimum, and whether the search is stagnating in a non-promising region (lines 17-26). BLS then applies L

perturbation moves to C in order to get a new starting point for the local search procedures (line 29).

2.2 Neighborhood Relations and Evaluation Functions

As previously explained, a solution to sum coloring with k colors can be represented as a k-partition $C = \{S_1, S_2, ..., S_k\}$ of vertices into k disjoint subsets $S_1, ..., S_k$, where each subset $S_i, i \in \{1, ..., k\}$ is associated with a unique integer i (i.e., color). The objective of MSCP is to find a coloring C that minimizes the total sum of colors assigned to vertices, while its implicit constraint requires that any two adjacent vertices $\{u, v\} \in E$ belong to different subsets. If this constraint is violated, we say the coloring is illegal with conflicting vertices.

A common move for minimum sum coloring and graph coloring problems is the exchange move (v, j), which consists in moving a vertex (element) $v \in V$ from its current subset S_i to another subset S_j (i.e., changing the color of v from i to j). The proposed BLS algorithm distinguishes two types of move exchange operators. The first type of move operators only considers exchange moves that do not violate the implicit problem constraint. For the second type of move operators, the implicit constraint is not strictly imposed in order to permit more freedom during the search. These two move operators are used in BLS under specific conditions as explained below.

If the solution C is a legal coloring, BLS applies to C a local search procedure (call it *LocalSearch*1) which consists in identifying the exchange move that decreases the most the objective value of Eq. (1) such that the new coloring remains legal. This process is repeated until reaching a local optimum.

If the solution C is an illegal coloring (i.e., with conflicting vertices), before applying *LocalSearch*1, BLS first applies another local search procedure (call it *LocalSearch*2) which evaluates all the moves (i.e., $\forall v \in V$, and $\forall j \in \{1, ..., k\}$ and $C(v) \neq j$) by considering both the variation $\Delta_{conf}(v, j)$ in the number of conflicts and the variation $\Delta_{sc}(v, j)$ in the sum of colors when exchanging the color of vertex v to j. Both $\Delta_{conf}(v, j)$ and $\Delta_{sc}(v, j)$ are negative for an improving move. Each move is then evaluated with the following relation:

$$\Delta f(v, j) = \Delta_{conf}(v, j) * \gamma(v, j), \text{ where} \tag{2}$$

$$\gamma(v, j) = \begin{cases} abs(\Delta_{sc}(v, j)) + k + 1 & \text{if } \Delta_{sc}(v, j) < 0 \\ k - \Delta_{sc}(v, j) + 1 & \text{otherwise} \end{cases} \tag{3}$$

*LocalSearch*2 performs at each step an exchange move with the smallest Δf, and stops after reaching a solution with no conflicting vertices (i.e., when $\Delta f = 0$). The evaluation function from Eq. 2 ensures that a conflicting vertex is eventually assigned a color $k + 1$ in case the conflict cannot be resolved by changing the color of v to a color less than or equal to k. However, a local optimum attained with *LocalSearch*2 is not necessarily a local optimum with respect to *LocalSearch*1. After reaching a local optimum with *LocalSearch*2, BLS thus applies $LocalSearch1$ to this local optimum which may improve the solution in terms of the objective value of Eq. (1).

Upon reaching a feasible locally optimal solution, BLS triggers its perturbation mechanism as described in the following section.

2.3 Adaptive Perturbation Mechanism

General Idea. The perturbation mechanism plays a crucial role within BLS since the descent-based local search alone cannot escape from a local optimum. BLS thus tries to move to the another basin of attraction by applying perturbations of different intensities depending on the search state. The idea of BLS is to first explore neighboring attractors. Therefore, after the local search phase, BLS performs most of the time a weak perturbation (by applying a small number L of moves) that is hopefully just strong enough to escape the current basin of attraction and to fall under the influence of a neighboring local optimum. If the jump was not sufficient to escape the current attractor, the perturbation strength L is incremented and the perturbation is applied again to the current attractor (lines 20–23 of Alg. 1). Otherwise, the number of perturbation moves L is reset to its default value, i.e., $L = L_0$. After visiting consecutively T legal local optima without any improvement of the best solution found, BLS sets the number of perturbation moves L to a significantly larger value $L = L_0$. (see lines 17–19 of Alg. 1). In addition to the number of perturbation moves, we also determine which type of moves are to be applied. Instead of making random jumps all the time, BLS alternates between several types of dedicated perturbation moves, depending on the desired amount of diversification (i.e., the state of search).

The Perturbation Strategies. The proposed BLS algorithm employs two directed perturbation strategies (Dp_1 and Dp_2), a recency-based perturbation (RB_p) and a random perturbation (R_p).

Directed perturbations are based on the idea of tabu list from tabu search [5]. These perturbations use a selection rule that favors the moves that minimize solution degradation, under the constraint that the moves are not prohibited by the tabu list. Move prohibition is determined in the following way. Each time vertex v is moved from subset S_i to S_j, it is forbidden to move v back to S_i for the next tt iterations (tt takes a random value from a given range). The information for move prohibition is maintained in a matrix H where the element $H_{v,j}$ is the iteration number when vertex v was last moved to subset S_j. The tabu status of a move is neglected only if the move leads to a new solution better than the best solution found so far. The *directed perturbations* rely thus both on history information and the quality of the moves to be applied for perturbation.

The first directed perturbation Dp_1 consists in making a legal non-tabu move which reduces the most the objective value, under constraint that the move does not violate an additional problem constraint. The second directed perturbation Dp_2 consists in performing a non-tabu exchange move (v, j) of the smallest $\Delta f(v, j)$, $\forall v \in V$, and $\forall j \in \{1, ..., k\}$ and $C(v) \neq j$ (see Eq. 2), such that the number of vertices in S_j is greater than or equal to the number of vertices in the current subset of v.

The recency-based perturbation RB_p consists in making the least recent legal exchange move (v, j) (i.e., a move that would not violate an additional constraint) provided that the subset S_j is not empty.

The move with random perturbation R_p consists first in selecting randomly two non-empty subsets S_i and S_j, such that $|S_i| \leq |S_j|$. A vertex v is then randomly chosen from S_i and moved to its new subset S_j.

Since moves with perturbations Dp_2 and R_p always lead to an illegal solution, we consider them to be stronger than perturbations Dp_1 and RB_p.

Our BLS approach takes turns probabilistically between a weaker perturbation (Dp_1 or RB_p) and a stronger perturbation (Dp_2 or R_p) depending on the search state, i.e., the current number of consecutive non-improving legal attractors visited ω. The idea is to apply weaker perturbation with a higher probability as the search progresses toward improved new solutions (when ω is small). With the increase of ω, the probability of using a stronger perturbation increases for the purpose of an important diversification.

Additionally, it has been observed from an experimental analysis that it is useful to guarantee a minimum of applications of a weaker perturbation. Therefore, we constraint the probability P of applying perturbations Dp_1 or RB_p to take values no smaller than a threshold P_0:

$$P = \begin{cases} e^{-\omega/T} & \text{if } e^{-\omega/T} > P_0 \\ P_0 & \text{otherwise} \end{cases} \tag{4}$$

where T is the maximum number of legal non-improving local optima visited before carrying out a stronger perturbation.

Given the probability P of applying perturbation Dp_1 or RB_p over Dp_2 or R_p, the probability of applying perturbation Dp_1 over RB_p is $P \cdot Q_1$ and $P \cdot (1 - Q_1)$ respectively, while the probability of applying Dp_2 over R_p is defined by $(1-P) \cdot Q_2$ and $(1-P) \cdot (1-Q_2)$ respectively. Q_1 and Q_2 are two coefficients that take a value from $[0, 1]$.

2.4 Experimental Results

Experimental Protocol. Our BLS algorithm is programmed in C++, and compiled with GNU gcc on a Xeon E5440 with 2.83 GHz and 2 GB. Two sets of benchmark graphs from the literature which are commonly used to test sum coloring algorithms are considered in the experiments. The first set is composed of 11 DIMACS[1] (DSJC125.1 to DSJC1000.5) instances. The second benchmark set is composed of 16 graphs from the COLOR02 competition website[2]. For all the instances, we run our BLS algorithm 20 times, with the maximum time limit set to 2 hours. However, BLS often attains its best result long before this limit. The parameter settings used to obtain the reported results are the following: initial jump magnitude $L_0 = 0.1 * |V|$, jump magnitude during strong perturbation $L_s = 0.25 * |V|$, maximum number of non-improving legal attractors

[1] ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/

[2] http://mat.gsia.cmu.edu/COLOR02/

visited before strong perturbation $T = 100$, smallest probability for applying weaker perturbation P_0 and probability coefficients Q_1 and Q_2 are set to 0.5.

We compare the performance of BLS with six recent algorithms from the literature. The comparison is solely based on the quality criterion according to Eq. (1) since information like the computing time are not always available for the reference approaches.

Table 1. Computational results of BLS on DIMACS and COLOR02 instances

Name	Sum*	Sum	Avg(Std)	t(min)	*Name*	Sum*	Sum	Avg(Std)	t(min)
DSJC125.1	326	326	326.9 (0.6)	18.3	jean	217	217	217.0 (0.0)	0.1
DSJC125.5	1015	**1012**	1012.9 (0.3)	51.0	queen5.5	75	75	75.0 (0.0)	0.0
DSJC125.9	2511	**2503**	2503.0 (0.0)	1.1	queen6.6	138	138	138.0 (0.0)	0.0
DSJC250.1	977	**973**	982.5 (3.4)	111.7	queen7.7	196	196	196.0 (0.0)	0.0
DSJC250.5	3246	**3219**	3248.5 (14.5)	104.1	queen8.8	291	291	291.0 (0.0)	0.8
DSJC250.9	8286	8290	8316 (13.2)	41.6	games120	443	443	443.0 (0.0)	0.5
DSJC500.1	2850	2882	2942.9 (19.6)	112.4	miles250	325	327	328.8 (0.9)	31.1
DSJC500.5	10910	11187	11326.3 (75.3)	118.8	miles500	709	710	713.3 (1.2)	86.3
DSJC500.9	29912	30097	30259.2 (63.3)	37.1	myciel3	21	21	21 (0.0)	0.0
DSJC1000.1	9003	9520	9630.1 (65.7)	112.9	myciel4	45	45	45.0 (0.0)	0.0
DSJC1000.5	37598	40661	41002.6 (169.3)	113	myciel5	93	93	93.0 (0.0)	1.3
anna	276	276	276.0 (0.0)	14.8	myciel6	189	189	196.6 (4.3)	20.0
david	237	237	237.0 (0.0)	2.5	myciel7	381	381	393.8 (8.4)	38.3
huck	243	243	243.0 (0.0)	0.3					

Computational Results and Comparisons. Table 1 presents computational results of our BLS algorithm on the sets of 27 DIMACS and COLOR02 instances. Column *Sum** shows the best-known results for these instances taken from references [6,13]. Column *Sum* indicates the best result obtained with BLS after 20 independent runs. Columns *Avg(Std)* and *t(min)* provide respectively the average and standard deviation values over 20 executions and the average CPU time in minutes required by BLS to reach its best result. From Table 1, we observe that for the DIMACS instance, BLS improved the best-known result for four instances, but failed to reach the best reported result for six other (larger) instances. The results for instances of COLOR02 benchmark indicate that BLS attained the best-known result for 14 instances, and was unable to reach the current best result for two instances. The average computing times required by BLS to attain its best reported results from column *Sum* range from less then a minute up to 120 minutes for the largest instances.

To further evaluate the performance of BLS, we show a comparison with the following approaches from the literature: a very recent heuristic EXSCOL [13]; a hybrid local search (HLS) [4]; a greedy algorithm (MRLF) based on the popular RLF graph coloring heuristic [10]; a parallel genetic algorithm (PGA) [9]; a tabu search algorithm (TS) [3]; and a recent local search combining ideas of variable neighborhood search and iterated local search (MDSLS) [6]. EXSCOL [13] is the current best-performing approach in the literature and is particularly effective on large graphs. It is based on an iterative extraction of large independent sets, where each independent set defines a color class. Moreover, EXSCOL is highly effective in terms of computing time. On the same computing platform as that we used to test BLS, it requires from 1 minute up to 30 minutes for large graphs (e.g., DIMACS1000.X) to reach the reported results. The time limit used by MDSLS

[6] is 1 hour on a computer similar to ours. For other reference approaches [3,4,9,10], details on testing conditions are not available.

Tables 2 and 3 report respectively the comparative results with these reference algorithms on the tested DIMACS and COLOR02 instances. From the results of Table 2, we observe that our BLS algorithm outperforms, for each of the tested DIMACS instance, the reference algorithms MRLF [10], TS [3], and MDSLS [6] (except on DSJC125.1 where BLS and MDSLS report the same result). However, compared to the highly effective EXSCOL algorithm [13], BLS shows a worse performance on large DIMACS instances, but is able to report a better result than EXSCOL for 4 DIMACS instances up to 250 vertices. Notice that two reference algorithms HLS and PGA do not report results for these instances.

For the COLOR02 instances, we observe in Table 3 that BLS shows a comparable performance to the recent MDSLS algorithm [6] and a slightly better performance than EXSCOL [13]. Moreover, the best result obtained with BLS is in each case either better or as good as that reported by HLS [4], MRLF [10] and PGA [9]. For TS, no results are reported on these instances in [3].

From the given comparison on both sets of benchmarks, we can conclude that our BLS algorithm for the MSCP is very competitive with the state of art approaches from the literature.

Table 2. Comparative results between our BLS algorithm and four reference approaches on the set of DIMACS instances

Name	Sum*	BLS	EXSCOL [13]	MRLF [10]	TS [3]	MDSLS [6]
DSJC125.1	326	**326**	**326**	352	344	**326**
DSJC125.5	1015	**1012**	1017	1141	1103	1015
DSJC125.9	2511	**2503**	2512	2653	2631	2511
DSJC250.1	977	**973**	985	1068	1046	977
DSJC250.5	3246	**3219**	3246	3658	3779	3281
DSJC250.9	8286	8290	**8286**	8942	9198	8412
DSJC500.1	2850	2882	**2850**	3229	3205	2951
DSJC500.5	10910	11187	**10910**	12717	–	11717
DSJC500.9	29912	30097	**29912**	32703	–	30872
DSJC1000.1	9003	9520	**9003**	10276	–	10123
DSJC1000.5	37598	40661	**37598**	45408	–	43614

Table 3. Comparative results between our BLS algorithm and five reference approaches on the set of COLOR02 instances

Name	Sum*	BLS	EXSCOL [13]	HLS [4]	MRLF [10]	PGA [9]	MDSLS [6]
anna	276	**276**	283	–	277	281	**276**
david	237	**237**	**237**	–	241	243	**237**
huck	243	**243**	**243**	**243**	244	**243**	**243**
jean	217	**217**	**217**	–	**217**	218	**217**
queen5.5	75	**75**	**75**	–	**75**	**75**	**75**
queen6.6	138	**138**	150	**138**	**138**	**138**	**138**
queen7.7	196	**196**	**196**	–	**196**	**196**	**196**
queen8.8	291	**291**	**291**	–	303	302	**291**
games120	443	**443**	**443**	446	446	460	**443**
miles250	325	327	328	343	334	347	**325**
miles500	709	710	**709**	755	715	762	712
myciel3	21	**21**	**21**	**21**	**21**	**21**	**21**
myciel4	45	**45**	**45**	**45**	**45**	**45**	**45**
myciel5	93	**93**	**93**	**93**	**93**	**93**	**93**
myciel6	189	**189**	**189**	**189**	**189**	**189**	**189**
myciel7	381	**381**	**381**	**381**	**381**	382	**381**

3 Discussions

One observes that among the different ingredients of the BLS algorithm (see Alg. 1), only the neighborhood relations, evaluation functions and perturbation moves (see sections 2.2 and 2.3) are specific to the MSCP. In fact, BLS is a generic search framework which inherits and combines features from iterated local search [11], tabu search [5] and simulated annealing [8]. We briefly discuss the similarities and differences between our BLS approach and these methods.

Following the general framework of ILS, BLS uses local search to discover local optima and perturbation to diversify the search. However, BLS distinguishes itself from most ILS algorithms by the combination of multiple perturbation strategies of different intensities, triggered according to the search status. In particular, a distinction of BLS is in the way a perturbation type is selected. As explained in Section 2.3, BLS applies a perturbation of weaker diversification with a higher probability P as the search progresses toward improved new local optima. This probability is progressively decreased as the number of consecutively visited non-improving local optima ω increases. The idea of an adaptive change of probability P is inspired by the popular acceptance criterion used in simulated annealing, which ensures that neighboring solutions (even those of bad quality) are accepted with higher probability when the temperature is high, and with lower probability as the temperature decreases.

In order to direct the search toward more promising regions of the search space, BLS employs directed perturbation strategies based on the notion of tabu list from tabu search. However, unlike tabu search, BLS does not consider the tabu list during its local search phases. As such, BLS and tabu search may explore quite different trajectories during their respective search, leading to different local optima. In fact, we believe that diversification during the descent-based improving phase is unnecessary. The compromise between search exploration and exploitation is critical only once a local optimum is reached. Other studies supporting this idea can be found in [1,7].

To validate the generality of BLS, in addition to the MSCP presented in this paper, we have applied BLS to solve several other classical combinatorial optimization problems (quadratic assignment, maximum clique [2], and maximum cut) and observed very competitive performances on benchmark instances.

4 Conclusion

In this paper, we have presented the Break Local Search algorithm for solving the minimum sum coloring problem. The computational evaluation of the proposed algorithm on two sets of 27 DIMACS and COLOR02 benchmark instances has revealed that BLS is able to improve the best-known results for 4 DIMACS instances and attain the best-known results for 15 instances while failing to reach the best ones for 8 instances. These results are competitive compared with most of the state of art approaches for the MSCP.

Acknowledgment. We are grateful to the referees for their comments and questions which helped us to improve the paper. The work is partially supported by the Pays de la Loire Region (France) within the RaDaPop (2009-2013) and LigeRO (2010-2013) projects.

References

1. Battiti, R., Protasi, M.: Reactive search, a history-based heuristic for max-sat. ACM Journal of Experimental Algorithmics 2 (1996)
2. Benlic, U., Hao, J.K.: Breakout local search for maximum clique problems. Operations Research 40(1), 192–206 (2013)
3. Bouziri, H., Jouini, M.: A tabu search approach for the sum coloring problem. Electronic Notes in Discrete Mathematics 36, 915–922 (2010)
4. Douiri, S.M., Elbernoussi, S.: New algorithm for the sum coloring problem. International Journal of Contemporary Mathematical Sciences 6, 453–463 (2011)
5. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
6. Helmar, A., Chiarandini, M.: A local search heuristic for chromatic sum. In: MIC 2011, pp. 161–170 (2011)
7. Kelly, J.P., Laguna, M., Glover, F.: A study of diversification strategies for the quadratic assignment problem. Computers and Operations Research 21(8), 885–893 (1994)
8. Kirkpatrick, S., Gelett, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 621–630 (1983)
9. Kokosiński, Z., Kwarciany, K.: On Sum Coloring of Graphs with Parallel Genetic Algorithms. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007, Part I. LNCS, vol. 4431, pp. 211–219. Springer, Heidelberg (2007)
10. Li, Y., Lucet, C., Moukrim, A., Sghiouer, K.: Greedy algorithms for minimum sum coloring algorithm. In: Proceedings of LT 2009 (2009)
11. Lourenco, H.R., Martin, O., Stützle, T.: Iterated local search. Handbook of Metaheuristics. Springer, Heidelberg (2003)
12. Malafiejski, M.: Sum coloring of graphs. In: Kubale, M. (ed.) AMS Graph Colorings, pp. 55–65 (2004)
13. Wu, Q., Hao, J.K.: An effective heuristic algorithm for sum coloring of graphs. Computers & Operations Research 39(7), 1593–1600 (2012)

XCS with Adaptive Action Mapping

Masaya Nakata[1], Pier Luca Lanzi[2], and Keiki Takadama[1]

[1] Department of Informatics
The University of Electro-Communications, Tokyo, Japan
[2] Dipartimento di Elettronica e Informazione
Politecnico di Milano, Milano, Italy

Abstract. The XCS classifier system evolves solutions that represent complete mappings from state-action pairs to expected returns therefore, in every possible situation, XCS can predict the value of all the available actions. Such complete mapping is sometimes considered redundant as most of the applications (like for instance, classification), usually focus only on the best action. In this paper, we introduce an extension of XCS with an adaptive (state-action) mapping mechanism (or XCSAM) that evolves solutions focused actions with the largest returns. While UCS evolves solutions focused on the best available action but can only solve supervised classification problems, our system can solve both supervised and multi-step problems and, in addition, it can adapt the size of the mapping to the problems: Initially, XCSAM starts building a complete mapping and then it slowly tries to focus on the best actions available. If the problem admits only one optimal action in each niche, XCSAM tends to focus on such an action as the evolution proceeds. If more actions with the same return are available, XCSAM tends to evolve a mapping that includes all of them. We applied XCSAM both to supervised problems (the Boolean multiplexer) and to multi-step maze-like problems. Our experimental results show that XCSAM can reach optimal performance but requires smaller populations than XCS as it evolves solutions focused on the best actions available for each subproblem.

1 Introduction

The XCS classifier system is a method of genetics-based machine learning [7] that combines robust reinforcement learning techniques [11] with evolutionary computation [6] to solve both classification and reinforcement learning problems. In particular, XCS evolves solutions that represent accurate and complete mappings from state-action pairs to expected returns so that the system can predict the value of all the available actions in every possible situation. The evolution of a complete mapping ensures that XCS can always evolve optimal, maximally accurate, maximally general solution to a given problem [8]. However, a complete mapping is sometimes considered redundant as most applications (e.g., classification) focus only on the best action (the best classification). To avoid a complete mapping, Bernado et al. [1] introduced UCS, an extension of XCS that can only solve classification (one-step) problems, and it is trained using supervised learning instead of reinforcement learning as XCS. While UCS has been very successful in data mining, its approach cannot generalized to broader machine learning applications.

In this paper, we introduce an extension of XCS, dubbed XCS with Adaptive Action Mapping (or XCSAM), that provides a trade-off between XCS and UCS with respect to the evolved state-action mapping. Like XCS, our system can solve both reinforcement learning (multi-step) problems and classification (one-step) problems. Similarly to UCS, our system does not evolve a complete mapping but focuses on the best actions while the learning proceeds. XCSAM adaptively learns only the *actions* leading to the highest returns. To achieve this, XCSAM extends the original XCS by (i) including a mechanism to adaptively identify the actions that are likely to be included in a best action mapping and by (ii) including a mechanism to get rid of redundant actions while still exploring the space of viable optimal solutions. The results we present show that XCSAM can evolve optimal solutions with much smaller populations than XCS. The analysis of our adaptation mechanism suggests that XCSAM can evolve mappings that covers only the best actions in every possible situation like UCS.

2 The XCS Classifier System

The XCS classifier system maintains a population of rules (the classifiers) which represents the solution to a reinforcement learning problem [10]. Classifiers consist of a condition, an action, and four main parameters [12,5]: (i) the prediction p, which estimates the relative payoff that the system expects when the classifier is used; (ii) the prediction error ε, which estimates the error of the prediction p; (iii) the fitness F, which estimates the accuracy of the payoff prediction given by p; and (iv) the numerosity *num*, which indicates how many copies of classifiers with the same condition and the same action are present in the population.

At time t, XCS builds a *match set* [M] containing the classifiers in the population [P] whose condition matches the current sensory input s_t; if [M] does not contain all the feasible actions *covering* takes place and creates a set of classifiers that matches s_t and cover all the missing actons. This process ensures that XCS can evolve a complete mapping so that in any state it can predict the effect of every possible action in terms of expected returns.[1]

For each possible action a in [M], XCS computes the *system prediction* $P(s_t, a)$ which estimates the payoff that the XCS expects if action a is performed in s_t. The system prediction $P(s_t, a)$ is computed as the fitness weighted average of the predictions of classifiers in [M] which advocate action a:

$$P(s_t, a) = \sum_{cl_k \in [M](a)} p_k \times \frac{F_k}{\sum_{cl_i \in [M](a)} F_i} \qquad (1)$$

[1] In the algorithmic description [5], covering is activated when match set contains less than θ_{nma} actions; however, θ_{nma} is always set to the number of available actions so that the match covers all the actions.

where, [M](a) represents the subset of classifiers of $[M]$ with action a, p_k identifies the prediction of classifier cl_k, and F_k identifies the fitness of classifier cl_k. Then, XCS selects an action to perform; the classifiers in [M] which advocate the selected action form the current *action set* [A]. The selected action a_t is performed, and a scalar reward r_{t+1} is returned to XCS together with a new input s_{t+1}. When the reward r_{t+1} is received, the estimated payoff $P(t)$ is computed as follows:

$$P(t) = r_{t+1} + \gamma \max_{a \in [M]} P(s_{t+1}, a) \tag{2}$$

where γ is the discount factor [11]. Next, the parameters of the classifiers in [A] are updated in the following order [5]: prediction, prediction error, and finally fitness. Prediction p is updated with learning rate β $(0 \leq \beta \leq 1)$:

$$p_k \leftarrow p_k + \beta(P(t) - p_k) \tag{3}$$

Then, the prediction error ϵ and classifier fitness are updated as usual [13,5]. On regular basis (dependent on parameter θ_{ga}), the genetic algorithm is applied to classifiers in [A]. It selects two classifiers, copies them, with probability χ performs crossover on the copies, and with probability μ it mutates each allele. The resulting offspring classifiers are inserted into the population and two classifiers are deleted to keep the population size constant.

3 XCS with Adaptive Action Mapping

The XCS classifier system [12] (as well as all the other models derived from it), evolves solutions that represent accurate and complete mappings from state-action pairs to expected returns; accordingly, XCS can predict in every possible situation the expected payoff of all the actions available. The maintenance of a complete mapping ensures that XCS can evolve optimal solutions since all the possible state-action pairs can be explored in every possible situations, as required by reinforcement learning theory [11]. On the other hand, the complete mapping increases the population size required to evolve an optimal solution and results in final populations containing information about actions never used in practice: in fact, in most of the applications, at the end only the best actions are applied. To avoid a complete mapping, Bernado et al. [1] introduced UCS, an extension of XCS that can only solve classification (one-step) problems, trained using supervised learning instead of reinforcement learning as XCS.

The system we propose, XCSAM or XCS with Adaptive Action Mapping, provides a trade-off between XCS and UCS. Like XCS, our system can solve both reinforcement learning (multi-step) and supervised classification (one-step) problems; similar to UCS, XCSAM does not evolve a complete mapping but tries to focus only on the best actions while learning how to solve the problem. In practice, XCSAM adaptively learns only the *actions* leading to the highest returns and, for this purpose, it extends the original XCS by (i) including a mechanism to adaptively identify the redundant actions from those that should be included in a best action mapping and by (ii) getting rid of redundant actions while still exploring the space of viable optimal solutions.

3.1 Identifying the Actions for the Best Mapping

In a typical reinforcement learning problem involving delayed rewards, the expected future reward at the current state $maxP(s_t, a)$ (Equation 2), tends to be higher than the expected future reward at the previous state $maxP(s_{t-1}, a)$ (because of the discount factor γ). Accordingly, since the action corresponding to a higher reward also corresponds to a shorter state sequence, the best actions will tend to have a value of $maxP(s_t, a)$ larger than $maxP(s_{t-1}, a)$. More precisely, $maxP(s_{t-1}, a)$ converges to $\gamma maxP(s_t, a)$ at the next state, while $maxP(s_t, a)$ converges to the $maxP(s_{t-1}, a)/\gamma$. Thus, in XCS the prediction of the accurate classifiers in $[A]_{-1}$ tend to converge to $\gamma maxP(s_t, a)$. For this reason, in XCS we can identify the actions that are likely to be part of the best mapping by comparing $maxP(s_t, a)$ against $\zeta \times maxP(s_{t-1}, a)/\gamma$ (where ζ is a learning rate added to guarantee convergence). If $maxP(s_t, a)$ is greater than the threshold $\zeta \times maxP(s_{t-1}, a)/\gamma$, then a is a good candidate for the best mapping and should be maintained.

3.2 Identifying Classifiers for Best Action Mappings

After having identified good action candidates for the best mapping, we need to adaptively identify classifiers that may be good candidate for the final best action mapping. Accordingly, we add a parameter *eam* to classifiers, or *effect of adaptive mapping*, that for classifier cl_i is updated according to Equation 4, where *nma* represents the number of available actions. The value of *eam* converges to 1, if the classifier is a good candidate for the final best action mapping, otherwise, *eam* converges to *nma*. Therefore, classifiers with an *eam* close to one are good candidates to represent the final best action mapping while classifiers with an *eam* close to *nma* are less likely to be maintained as they are probably advocating actions with lower expected return.

$$eam_i \leftarrow \begin{cases} eam_i + \beta(1 - eam_i) \text{ if } & maxP(s_t, a) \geq \zeta \times maxP(s_{t-1}, a)/\gamma \\ eam_i + \beta(nma \ - eam_i) & \text{otherwise.} \end{cases} \quad (4)$$

3.3 Focusing Evolution on the Best Actions

To focus evolution on the best actions, XCSAM acts on the covering operator to prevent the generation of classifiers that are not likely to be included in the final solution. In particular, XCSAM tunes the activation threshold of the genetic algorithm θ_{nma} based on the actions' predicted reward and the *eam* parameters. Initially, θ_{nma} is set to the number of feasible actions (the same value used in XCS). When $[M]$ is generated (Section 2), XCSAM computes the prediction array before the covering is applied (whereas XCS computes it only after covering). Then, XCSAM computes the current θ_{nma} as the average *eam* of the classifiers in [M] weighted for the expected future return $maxP(s_t, a)$. If the number of different actions in $[M]$ is smaller than the computed θ_{nma}, covering is called and the prediction array is computed again. If solving a single-step problem, XCSAM computes the selection array $S(a)$ that associate a selection probability to each action as follows:

$$S(a_j) = \frac{\sum_{cl_k \in [M]|a_j} p_k \times F_k}{\sum_{cl_k \in [M]} F_k} \tag{5}$$

After action selection is performed, XCSAM generates both the action set [A] (as XCS) and also the *not action set* $[\bar{A}]$ consisting of the classifiers in $[M]$ not advocating the selected action. When the executed action is considered a candidate best action, the parents in GA are selected from $[A]$ to promote the evolution of classifiers that are likely to be in the final best action mapping while the deleted classifiers are selected from $[\bar{A}]$ to get rid of classifiers that are not likely to be part of the final solution. Otherwise, if there is not enough information about the executed action, or $[\bar{A}]$ is empty, XCSAM applies deletion in $[P]$ as in XCS. When the executed action is not identified as a candidate best action, the parents are selected from $[\bar{A}]$ to explore the solution space even further and deletion is applied to the population as in XCS. It is important to note that, in single-step problems the selection $S(a)$ is only used to select which action to apply and all the other XCSAM mechanisms work the same independently from the type of problem considered.

4 Experimental Results

We tested XCSAM by applying it both to single-step problems (the Boolean multiplexer [12]) and to multi-step problems (the woods environments [12]). In particular, we compared the performance of XCSAM to XCSTS [4] using the same experimental settings used in previous papers.

4.1 Design of Experiments

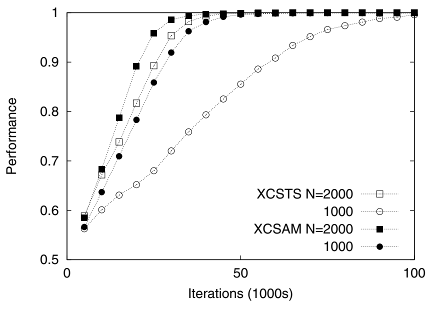
Each experiment consists of a number of problems that the system must solve. Each problem is either a *learning* problem or a *test* problem. During *learning* problems, the system selects actions randomly from those represented in the match set. During *test* problems, the system always selects the action with highest expected return and no update is performed. When the system performs the correct action (either it returns the correct class for a classification problem or it reaches the goal for a multi-step problem), it receives a 1000 reward, 0 otherwise. The genetic algorithm is enabled only during *learning* problems, and it is turned off during *test* problems. The covering operator is always enabled, but operates only if needed. Learning problems and test problems alternate. The performance is reported as the moving average over the last 5000 tests in single step problems and as the moving average over the last 50 steps in multi-step problems. All the reported plots are averages over 10 experiments.

4.2 Boolean Multiplexer

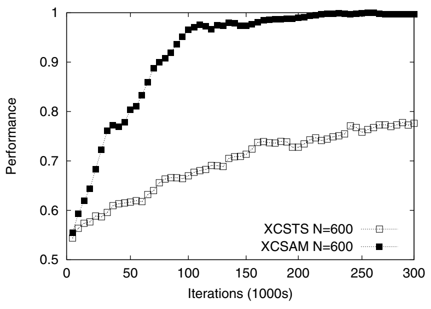
The Boolean multiplexer function of size k is defined over a binary string of $k+2^k$ bits; the first k bits represent and address pointing to the rest of 2^k bits. For example, the 6-multiplexer function ($k = 2$), applied to 110001 returns 1, when it is applied to 110110 returns 0. In the experiments, we applied the standard parameter settings for XCS [3]: $\epsilon_0 = 10$, $\mu = 0.04$, $P_{\#} = 0.65$, P_{explr} =1.0, $\chi = 0.8$, $\beta = 0.2$, $\alpha = 0.1$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 25$, $\theta_{del} = 20$, $\theta_{sub} = 20$, $\tau = 0.4$, tournament selection is applied, GA subsumption and AS subsumption are also applied; for XCSAM, we applied the same parameters of XCSTS and in addition we set $\zeta = 0.99$.

Figure 1 compares the performance of XCSAM and XCS on 20-multiplexer problems with $N = 600, 1000$ and 2000. As it can be noticed, with 2000 classifiers, XCSAM reaches optimality a little bit faster than the standard XCS; when the population size is decreased to 1000 classifiers, XCSAM clearly learns much faster than XCS, in fact after 100000 experiments XCS still did not reach full optimality. When the population size is decreased even further down to 600 classifiers, we note than XCSAM reaches a near-optimal performance around 250000 problems whereas XCS performance is significantly degraded remaining around the 70% after 300000.

Figure 2 compares the performance of XCSAM and XCS on the more difficult 37-multiplexer when 5000 (Figure 2a) and 3000 (Figure 2b) classifiers are used. All the parameters are set as in the standard experiments while for XCSAM we set $\zeta = 0.99$. As can be noticed, with the standard settings [3], XCSAM learns faster than XCS and both systems reach optimality. When the population size is almost halved down to 3000 classifiers, the performance of XCS is significantly degraded and after one million examples, its performance is still below 60%.

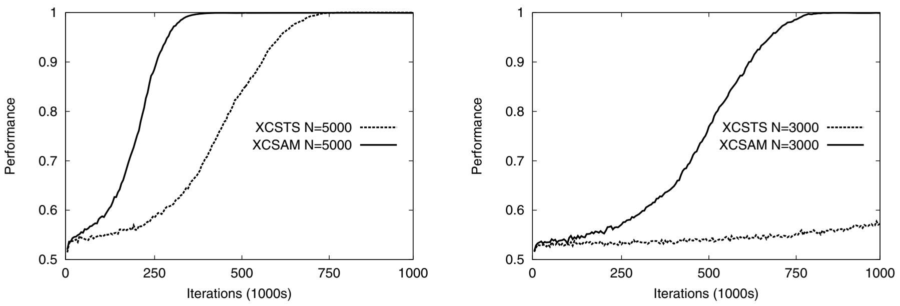


a) N = 2000 and 1000



b) N =600

Fig. 1. XCSAM and XCSTS applied to the 20-multiplexer problem with 2000, 1000, and 600 classifiers



a) $N = 5000$ b) N =3000

Fig. 2. XCSAM and XCSTS applied to the 37-multiplexer problem with 5000 and 3000 classifiers

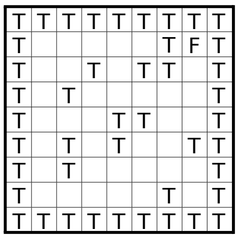


Fig. 3. The `Maze6` environment

4.3 Multi-step Problems

We performed a second set of experiments to compare the performance of XCS with gradient descent [2] and our XCSAM on the woods environments a typical multi-step problems introduced by Wilson [12]. These are grid-like mazes in which each cell can be either empty, can contain an obstacle (represented with a "T" symbol), or food (represented by a "F" symbol). The system always occupies an empty position and perceives the eight surrounding cells. In each experiment, the system is randomly placed in one of the empty positions and has to reach a food position while avoiding obstacles. When it reaches a food position, the system receives a reward of 1000 and zero otherwise. Figure 3 shows the `Maze6` environment, firstly introduced in [9].

We compared XCS and XCSAM using `Maze6` and the standard configuration used in [2]: $\epsilon_0 = 1$, $\mu = 0.01$, $P_{\#} = 0.3$, P_{explr} =1.0, $\chi = 0.8$, $\beta = 0.2$, $\alpha = 0.1$, $\delta = 0.1$, $\gamma = 0.7$, $\nu = 5$, $\theta_{GA} = 100$, $\theta_{del} = 20$, $\theta_{sub} = 20$, Tournament selection, GA Subsumption and ASsubsumption = 0 are turned off. Gradient descent is always on; the XCSAM parameter is $zeta = 0.99$ performance is computed as the moving average number of steps to a goal position over the last 50 problems. All experimental result are averaged over 20 experiments.

Figure 4 compares the performance of XCSAM and XCS when population size is 3000 (Figure 4a) and 1000 (Figure 4b) classifiers. With a large population both XCSAM and XCS reach optimal performance (i.e., 5.19 steps) whereas, when the population is decreased to 1000 classifiers, XCSAM can still reach a near-optimal performance whereas XCS (event with gradient) cannot reach stable performance. These results confirm the previous findings showing that XCSAM can reach optimal performance with much smaller populations.

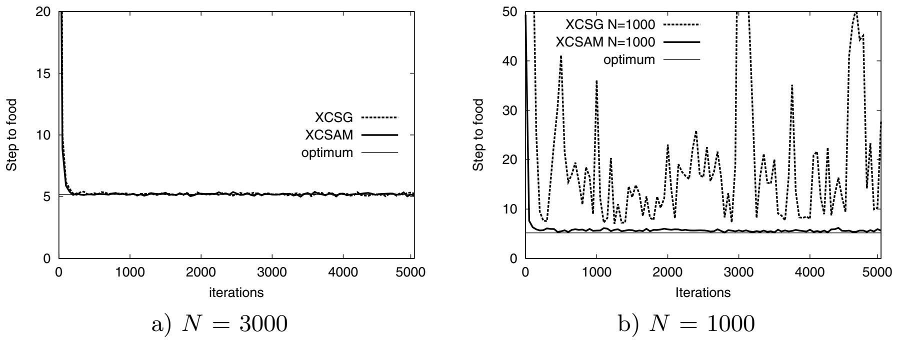


Fig. 4. XCSAM and XCS applied to `Maze6` with a population of (a) 3000 classifiers and (b) 1000 classifiers

4.4 Evolution of the Best Action Mapping

The previous results demonstrates that, by adaptively focusing on the best action mapping, XCSAM can learn optimal policies with smaller populations. We now analyse the adaptation toward the best mapping. Figure 5 shows the average *eam* of the classifiers contributing $maxP(s_t, a)$ in the 37-multiplexer problem and in `Maze6`. As can be noted, in the 37-multiplexer, XCSAM starts from an almost complete mapping (in fact the average *eam* is around 1.5) then, as evolution proceeds, XCSAM can successfully identify the best actions and thus the average *eam* converges to a value near 1, that is, the final solution basically covers only the best action. A similar result is also shown for `Maze6` (Figure 5b) where the

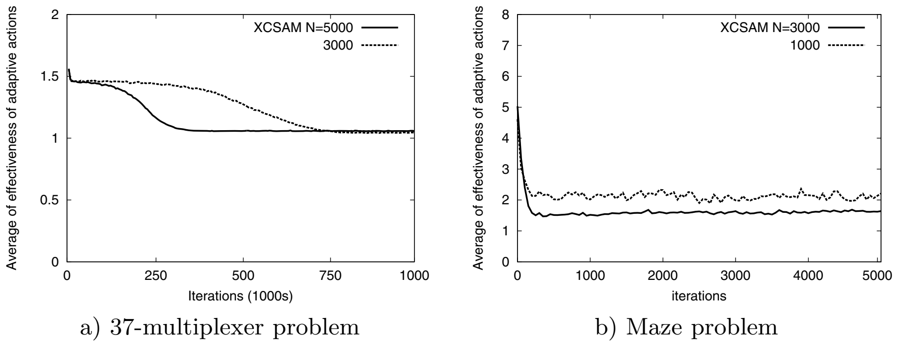


Fig. 5. Average *eam* in (a) 37-multiplexer and (b) `Maze6`

average *eam* starts from a value around 5 and rapidly converges to a value near 1.5 thus showing that XCSAM can correctly distinguish the best actions adaptively during learning. In fact, `Maze6` has 16 cells each one with two best actions, 1 cell with three best actions, 19 cells with only one best action which amounts to 36 empty cells. Therefore the average of number of best action in each cells is equal to 1.5 (=$(2 \times 16 + 3 + 1 \times 19)/36$). The results in Figure 5b thus show that XCSAM can correctly identify the best actions even in problems in which states can have more than one best actions.

5 Conclusion

We introduced XCSAM, an extension of XCS that adapts the state-action mapping to focus on the best action in every possible subproblem. We applied the proposed system both to classification (single-step) problems (the 20-multiplexer and the 37-multiplexer) and to a reinforcement (multi-step) problem (`Maze6`). Our results show that XCSAM can evolve optimal solutions with much smaller populations than XCS. The analysis of the adaptation mechanism suggests that XCSAM can evolve mappings that covers only the best actions in every possible situation.

References

1. Bernadó-Mansilla, E., Garrell, J.M.: Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. Evolutionary Computation 11, 209–238 (2003)
2. Butz, M.V., Goldberg, D.E., Lanzi, P.L.: Gradient Descent Methods in Learning Classifier Systems: Improving XCS Performance in Multistep Problems. Evolutionary Computation 9(5), 452–473 (2005)
3. Butz, M.V., Kovacs, T., Lanzi, P.L., Wilson, S.W.: Toward a Theory of Generalization and Learning in XCS. IEEE Transactions on Evolutionary Computation 8(1), 28–46 (2004)
4. Butz, M.V., Sastry, K., Goldberg, D.E.: Tournament Selection: Stable Fitness Pressure in XCS. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1857–1869. Springer, Heidelberg (2003)
5. Butz, M.V., Wilson, S.W.: An algorithmic description of xcs. Journal of Soft Computing 6(3-4), 144–153 (2002)
6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley (1989)
7. Holland, J.H.: Escaping Brittleness: The Possibilities of General Purpose Learning Algorithms Applied to Parallel Rule-based system. Machine Learning 2, 593–623 (1986)
8. Kovacs, T.: Evolving optimal populations with XCS classifier systems. Technical Report CSR-96-17 and CSRP-96-17, School of Computer Science, University of Birmingham, Birmingham, U.K. (1996), Available from the technical report archive, `ftp://ftp.cs.bham.ac.uk/pub/tech-reports/1996/CSRP-96-17.ps.gz`

9. Lanzi, P.L.: An Analysis of Generalization in the XCS Classifier System. Evolutionary Computation Journal 7(2), 125–149 (1999)
10. Lanzi, P.L.: Learning classifier systems from a reinforcement learning perspective. Soft Computing - A Fusion of Foundations, Methodologies and Applications 6(3), 162–170 (2002)
11. Sutton, R.S., Barto, A.G.: Reinforcement Learning – An Introduction. MIT Press (1998)
12. Wilson, S.W.: Classifier Fitness Based on Accuracy. Evolutionary Computation 3(2), 149–175 (1995)
13. Wilson, S.W.: Classifier Fitness Based on Accuracy. Evolutionary Computation 3(2), 149–175 (1995), `http://prediction-dynamics.com/`

DEAL: A Direction-Guided Evolutionary Algorithm

Cuong C. Vu[1], Lam Thu Bui[1], and Hussein A. Abbass[2]

[1] Le Quy Don Technical University, Vietnam
`{cuongvcc,lam.bui07}@gmail.com`
[2] University of New South Wales, Australia
`h.abbass@adfa.edu.au`

Abstract. In this paper, we propose a real-valued evolutionary algorithm being guided by *directional information*. We derive direction of improvement from a set of elite solutions, which is always maintained overtime. A population of solutions is evolved over time under the guidance of those directions. At each iteration, there are two types of directions that are being generated: (1) *convergence direction* between an elite solution (stored in an external set) and a second-ranked solution from the current population, and (2) *spreading direction* between two elite solutions in the external set. These directions are then used to perturb the current population to get an offspring population. The combination of the offsprings and the elite solutions is used to generate a new set of elite solutions as well as a new population. A case study has been carried out on a set of difficult problems investigating the performance and behaviour of our newly proposed algorithm. We also validated its performance with 12 other well-known algorithms in the field. The proposed algorithm showed a good performance in comparison with these algorithms.

Keywords: direction of improvement, evolutionary algorithms.

1 Introduction

Evolutionary algorithms (EAs) have been popular tools for approximating solutions of optimization problems. For real-parameter EAs such as real-parameter GA, differential evolution (DE), evolutionary strategies (ES) and evolutionary programming (EP), a real-valued representation of genes is used. In the literature of evolutionary computation, the use of elitism has been the most popular one. For it, usually a number of good solutions, *the elite set*, is (either implicitly or explicitly) maintained over time. Our motivation is that this set can contribute information to the evolutionary process much more than just storing some solutions to the next generation.

Our proposal is as follows: we can derive from the elite set some directions of improvement and use these directions to guide the evolutionary process. Note that when designing an optimization algorithm, it is desirable to find a good direction that guides the search process; the *steepest gradient descent* method is the most typical example of getting advantage from a good direction to guide the search. However, in general, defining a good direction is not a trivial task, especially in the case of non-linear or black-box functions. For evolutionary computation, the use of directions has been shown quite promising. An example is the case of Differential Evolution (DE), which uses the direction between two randomly-selected parents to guide the newly-generated offsprings [17]; DE has been very effective in solving continuous optimization problems.

There are several ways to maintain the elite set (in short, ETS). In this proposal, we select a number of best solutions for ETS. Our main design will focus on how to use and refine ETS. Its size is set as half of the main population. ETS is used not only to contribute solutions to the next generation, but also to generate directions for offspring production. At each generation, a pool of offsprings (having the same size as the main population) is produced. This is done via a perturbation process in which randomly-selected parents (from the main population) are perturbed by directions of improvement. There are two types of directions presented in this work: (1) convergence direction that is defined as the direction between an elite solution from ETS and a second-ranked solution from the main population and (2) spreading direction between two elite solutions in ETS. This pool of offsprings is combined with ETS in order to generate the new content of ETS and the next generation. The combined population is then sorted and copied to ETS as well as the main population. To validate the newly proposed algorithm, we carried out a case study on 6 benchmark problems with high modality. The results from these problems showed that our algorithm performed quite well. Further, we also obtained the results from 12 other well-known algorithms. The results indicated that our algorithm was very competitive with these algorithms.

The paper is organized in seven sections. Section 2 is dedicated to background literature and followed by the methodology section. A number of test problems are presented, solved and studied in Section 4 to demonstrate the concept, and the paper concludes in Section 5.

2 Background

Evolutionary Algorithms (EAs) have been well recognized as a major class of heuristic techniques in computational optimization and machine learning. They have been applied widely in many aspects of human life from social and engineering problems to security and military domains. In principle, an EA is a process that imitates evolution in Nature. It works with a population of solutions in searching for the problem's optima where the population undergoes an evolutionary process of many cycles using nature-mimicking crossover, mutation and/or selection operators. After each cycle, a new population is formed and is called a *generation*. With this population-based computational approach, EAs offer a potential paradigm for solving global optimization problems [9,1,15,16,10]. Originally, there were four classes of EAs; including Genetic Algorithms (GAs), Evolutionary Strategies (ES), Evolutionary Programming (EP), and Genetic Programming (GP). To date, there are several paradigms that have emerged as alternatives for the conventional EAs, such as Particle Swarm Optimization (PSO) [12], Ant Colony Optimization (ACO) [7], Differential Evolution (DE) [17], Estimation of Distribution Algorithms (EDA) [14], and Artificial Immune Systems (AIS) [3]. For them, mutation and crossover operators might be replaced by some specific operator inspired by a different phenomenon in nature.

Real-parameter evolutionary algorithms can be classified into different streams. In general, the difference between these streams is in the way to employ and implement the evolutionary operators. The first stream is the real parameter GA that has the same framework as the binary-coded GA with a focus on crossover. As an example, Deb et al [5] introduced a version of the real parameter GA using the SBX crossover operator that simulates the binary crossover operator. A reasonable overview of real parameter GA can be found in [4]. Meanwhile, ES [18] and EP [8,20] concentrate more on the mutation operator. The child is generated by disturbing a selected solution with Gaussian or Cauchy distributed random deviations. After this phase, all solutions have to undergo a selection process.

In the case of simple DE, it uses one main parent and two supportive parents [19,2] for generating a child. Basically, the main parent is disturbed by adding a step length multiplied by the difference between the two supportive parents. The resultant solution is called the trial/protoype solution. The prototype is then crossed-over with another pre-selected solution to generate a child. Elitism is implemented implicitly in the way that the child is inserted into the population if it outperforms the pre-selected solution. By using difference vectors, DE takes into account direction information. In some cases, good directions will be generated and DE will generate good solutions. In other cases, bad directions will be generated which will deteriorate the solution quality. This poses a question on whether or not we can systematically maintain good directions?

Several other real-valued versions can be listed here such as covariance matrix adaptation evolution strategy (CMAES) [11] being implemented with an adaptive covariance matrix to model a second-order approximation of the objective function, and generalized generation gap model with generic parent-centric recombination operator (G3PCX) [6] using elite-preservation and the parent-centric recombination operator. Recently, real-coded version of chemical reaction optimization emerges as a new paradigm for real-valued optimization [13].

3 Methodology

3.1 Overview

It has been demonstrated that elitism is useful for an EA. However, the issue is how to use it effectively? Therefore, we will focus our work on this issue when designing a new EA; especially we will address: (1) *Interaction between an ETS and the main population* and (2) *updating the ETS.*

Our methodology proposes to maintain an ETS during the optimization process. This ETS will contribute to the evolutionary process not only the elitist solutions, but also the directional information (which we call as direction of improvement). A direction of improvement is considered as a useful piece of information during optimization process. Our proposal is that at every generation, the main population will be perturbed by these directions in order to produce offsprings. Theses offsprings are then combined with the current ETS to form a temporary population, called "*the combined population*". This combined population is used subsequently to fill in the new version of ETS. Based on this merit, we call our algorithm as Direction-guided Evolutionary ALgorithm or DEAL.

3.2 Directional Information

We propose to use two types of directional information: convergence and spreading.

- *Convergence direction*: It is defined as the direction from a solution to a better one. We consider it as the direction between second-ranked solution and an elite one. If elite solutions are maintained globally, it is considered as the global direction of convergence. If a solution is guided following this direction, it will find a better area.
- *Spreading direction*: It is defined as the direction between two peers. In this context, it is the direction between two elite solutions. If solutions are perturbed along these directions, a better spreading within the population will be obtained.

3.3 General Structure

A step-wise structure of the proposed algorithm is given as follows:

```
– Step 1: Initialize the main population $P$ with size $N$
– Step 2: Evaluate the population $P$
– Step 3: Copy elite solutions to ETS (that has the half size of population $P$)
– Step 4: Report the elite solutions in ETS
– Step 5: Generate a mixed population M with size of $N$, and set $index = 0$
– Loop {
    • Copy $P(index)$ to $M(index)$
    • Select a random parent $P_r$
    • Generate a convergence direction $d_1$ from a randomly-selected low-rank
      solution in the population P to a randomly-selected solution from ETS.
    • Generate a spreading direction $d_2$ between two randomly selected solu-
      tions in ETS.
    • Generate two offspring solutions $S_1$ and $S_2$ by perturbing the parent
      solution using two newly generated directions.
        * For each dimension $i$
            · If $U(0,1) < pc$ then $S_1(i) = P_r(i) + \sigma_1 * d_1(i)$
            · Else $S_1(i) = P_r(i)$
            · If $U(0,1) < pc$ then $S_2(i) = P_r(i) + \sigma_2 * d_2(i)$
```

```
          · Else $S_2(i) = P_r(i)$
        ∗ End for
        Where $U(0, 1)$ is the random function returning values between 0 and 1,
        $pc$ is crossover rate, $\sigma_1 = U(0, 1)$; $\sigma_2$= a small constant (i.e 0.5).
      • Evaluate $S_1$
      • $S_1$ is better than $M(index)$ then replace $M(index)$ by $S_1$
      • Mutate $S_2$ with a predefined rate $pm$
      • Evaluate $S_2$
      • $S_2$ is better than $M(index + 1)$ then replace $M(index + 1)$ by $S_2$
    $index = index + 2$
  – } Until (the mixed population is full)
  – Step 6: Combine the mixed population M with ETS to form a combined
    population C (or M+A → C)
  – Step 7: Sort $C$ using fitness values
  – Step 8: Determine the new members of ETS by copying first $N/2$ elite
    solutions from the combined population C)
  – Step 9: Determine the new population P by copying solutions from $M$)
  – Step 10: Go to Step 4 if stopping criteria is not satisfied
```

Note that ETS can be maintained implicitly (without an explicit data structure). The current main population can be classified into two parts: the first half is the elite solutions copied from the combined population C and this part is actually ETS and the second half is the lower ranked (or second-ranked) solutions. σ is the step length for perturbation. For σ_1, our finding is that the best strategy is $\sigma_1 = U(0, 1)$ and σ_2 is 0.5 (that basically reduces half of the vector's magnitude). The step 5 is the main element in this structure. It shows that for the offsprings, half of them are created for convergence purpose (exploitation) while the other half to make it more diverse (exploration).

The main computational cost comes from the task of filling ETS. Filling solutions requires sorting the combined population C. In general, a sorted procedure requires complexity of $O(NlogN)$. So, the overall complexity of the algorithm is $O(NlogN)$.

4 A Case Study

4.1 Testing Problems

We considered to test a set of 6 popular continuous test problems with high-dimensionality and high modality [13]. The only reason for us to select these problems is that these problems illustrate the highest difficulty facing optimization algorithms: multi-modality. They are reported in Table 1.

4.2 Experimental Setup

We selected other well-known algorithms for validating ours with settings used by authors in [13]: Real-coded version of Chemical Reaction Optimization (RC-CRO), Genetic Algorithm (GA), Fast evolutionary programming(FEP), Classical evolutionary programming (CEP), Fast evolutionary strategy (FES), Conventional evolutionary strategy (CES), Particle Swarm Optimization (PSO),

Table 1. Lists of test problems used for experiments in this paper

ID	n	Description	Name	Range	f_{min}
F1	30	$f(x) = -\sum_{i=1}^{n}(x_i sin(\sqrt{\lvert x_i \rvert}))$	Generalized Schwefel's problem	$[-500, 500]$	-12569.5
F2	30	$f(x) = \sum_{i=1}^{n}(x_i^2 - 10cos(2PIx_i) + 10)$	Generalized Rastrigin's problem	$[-5.12, 5.12]$	0
F3	30	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i^2)})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n}(cos(2PIx_i)) + 20 + e$	Ackley's problem	$[-32, 32]$	0
F4	30	$f(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i^2) - \prod_i^n cos(\frac{x_i}{\sqrt{i}}) + 1$	Generalized Griewank's problem	$[-600, 600]$	0
F5	30	$f(x) = \frac{PI}{n}\{10sin^2(PIy_1) + \sum_{i=1}^{n-1}(y_i - 1)^2$ $[1 + 10sin^2(PIy_{i+1})] + (y_n - 1)^2\}$ $+\sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & \text{otherwise.} \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	Generalized Penalized problem	$[-50, 50]$	0
F6	30	$f(x) = 0.1\{sin^2(3PIx_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1+$ $sin^2(3PIx_{i+1})] + (x_n - 1)^2[1 + sin^2(2PIx_n)]\}$ $+\sum_{i=1}^{n} u(x_i, 5, 100, 4)$	Generalized Penalized problem	$[-50, 50]$	0

Group search optimizer (GSO), Real-coded biogeography-based optimization (RCBBO), Differential evolution (DE), Covariance matrix adaptation evolution strategy (CMAES), and Generalized generation gap model with generic parent-centric recombination operator (G3PCX).

The experiments for our algorithm are carried out on all 6 test problems and with the following parameters: The population size was also 100 solutions, the number of evaluations are 150000, 250000, 150000, 150000, 150000, and 150000 for all problems respectively, the mutation rate was kept at the same small rate of 0.01, and the crossover rate was 0.9. Further, there were 100 runs with different random seeds for testing each problem.

4.3 Results and Discussion

Behavior Analysis: To analyze the behavior of DEAL, we first tested it on a spherical problem, the easiest problem: $f(x) = \sum_{i=1}^{n} x_i^2$, with $n = 30, x \in [-100, 100]$ (the optimal point is at the origin and we call it as the zero point). We recorded the objective value of the best solution found over time. After 150000 evaluations, DEAL obtained a near-zero average objective value (in different 100 runs) of 1.558E-12 (standard deviation is 6.158E-12).

From Figure 1, it is obvious that DEAL stably converged towards the optimal solution among all 100 runs. After 250 generations, solutions found by DEAL's 100 runs were almost close to the optimal point (the left graph). The right graph is magnified by logarithmic transformation and shows the constant convergence towards the zero point. In the decision space, all the points have x_i values being around the zero point with radius of 10e-8.

Another look at can be seen at Figure 2 where we displayed the behavior of DEAL on a multi-modal problem: the Akley problem (F3). The difficulty

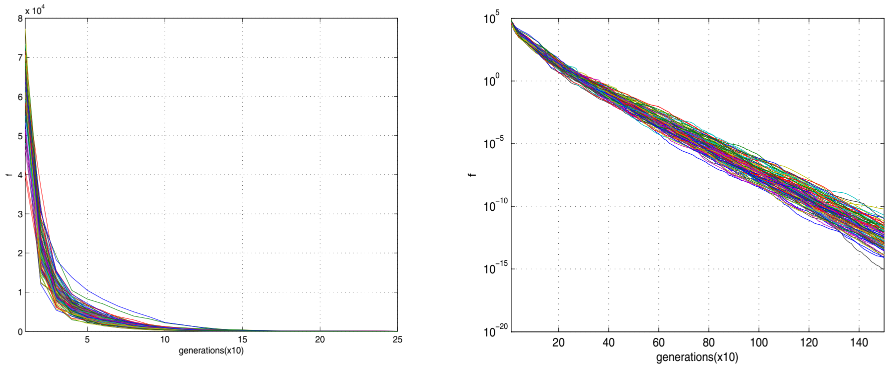


Fig. 1. Visualization of the best solution for the spherical problem found by DEAL in all 100 runs. Left graph: the convergence curve during the first 250 generations. Right graph: the convergence curve with log-transformation during all 1500 generations.

of multi-modality clearly made DEAL longer to converge. In contrast the case of the spherical problem, at generation 25rd (where DEAL converged for the spherical problem), DEAL stilled far from the optimal point. In all 100 runs, it converged almost at generation 500rd. After that the optimization still refining its best solution until the end (see the right logarithmic-transformed graph)

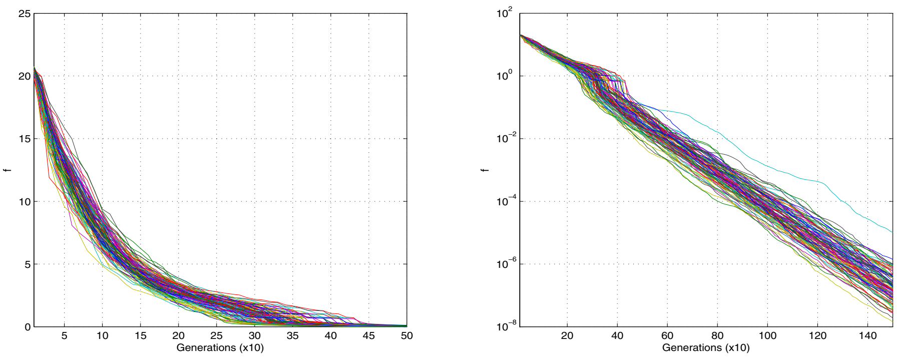


Fig. 2. Visualization of the best solution for Akley problem found by DEAL in all 100 runs. Left graph: the convergence curve during the first 500 generations. Right graph: the convergence curve with log-transformation during all 1500 generations.

Comparison with Others: In comparison with other approaches, we recorded the best objective value obtained by approaches for all problems in Table 1 and then calculated the mean and standard deviation. They are reported in Table 2 together with results obtained from [13]. From the table we can see that there is no clear winer among all 13 approaches. However, it indicates that 8 approaches (GA, FEP, CEP, FES, CES, PSO, RCBBO, and G3PCX)were inferior in all 6 test problems. The remaining 5 algorithms are seemed better in which each had at leat the result on one problem with 1st rank. Within this set of approaches, DEAL and DE emerged with more competitive results. While DEAL has two problems ranked No1 (the most case), DE has 1 problem ranked 1st and 3 problems ranked 2rd. The interesting note here is that both DEAL and DE used direction explicitly: while DE used direction between two randomly-selected parents, DEAL used direction of improvement.

Table 2. Obtained results for all test problems (Mean, Standard deviation and rank)

		F1	F2	F3	F4	F5	F6
DEAL	Mean	-1.085E+04	3.482E-15	3.097E-07	1.305E-02	3.619E-07	1.099E-04
	Std	3.996E+02	5.888E-15	8.720E-07	1.721E-02	1.268E-06	1.099E-03
	Rank	9	1	1	5	3	5
RCCRO1	Mean	-1.257E+04	9.077E-04	1.944E-03	1.117E-02	2.074E-02	7.048E-07
	Std	2.317E-02	2.876E-04	4.190E-04	1.622E-02	5.485E-02	5.901E-07
	Rank	2	3	5	3	7	1
GA	Mean	-1.257E+04	6.509E-01	8.678E-01	1.004E+00	4.372E-02	1.681E-01
	Std	2.109E+00	3.594E-01	2.805E-01	6.755E-02	5.058E-02	7.068E-02
	Rank	2	7	9	3	10	10
FEP	Mean	-1.255E+04	4.600E-02	1.800E-02	1.600E-02	9.200E-06	1.600E-04
	Std	5.260E+01	1.200E-02	2.100E-02	2.200E-02	6.140E-05	7.300E-05
	Rank	8	5	7	6	4	6
CEP	Mean	-7.917E+03	8.900E+01	9.200E+00	8.600E-02	1.760E+00	1.400E+00
	Std	6.345E+02	2.310E+01	2.800E+00	1.200E-01	2.400E+00	3.700E+00
	Rank	11	12	12	9	12	12
FES	Mean	-1.256E+04	1.600E-01	1.200E-02	3.700E-02	2.800E-02	4.700E-05
	Std	3.253E+01	3.300E-01	1.800E-03	5.000E-02	8.100E-11	1.500E-05
	Rank	7	6	6	8	8	4
CES	Mean	-7.550E+03	7.082E+01	9.070E+00	3.800E-01	1.180E+00	1.390E+00
	Std	6.314E+02	2.149E+01	2.840E+00	7.700E-01	1.870E+00	3.330E+00
	Rank	12	11	11	11	11	11
PSO	Mean	-9.660E+03	2.079E+01	1.340E-03	2.323E-01	3.950E-02	5.052E-02
	Std	4.638E+02	5.940E+00	4.239E-02	4.434E-01	9.142E-02	5.691E-01
	Rank	10	9	4	10	9	9
GSO	Mean	-1.257E+04	1.018E+00	2.655E-05	3.079E-02	2.765E-11	4.695E-05
	Std	2.214E-02	9.509E-01	3.082E-05	3.087E-02	9.167E-11	7.001E-04
	Rank	2	8	2	7	1	3
RCBBO	Mean	-1.257E+04	2.620E-02	2.510E-02	4.820E-01	3.280E-05	3.720E-04
	Std	2.200E-05	9.760E-03	5.510E-03	8.490E-02	3.330E-05	4.630E-04
	Rank	2	4	8	12	5	7
DE	Mean	-1.257E+04	7.261E-05	7.136E-04	9.054E-05	1.886E-07	9.519E-07
	Std	2.333E-05	3.376E-05	6.194E-05	3.402E-05	4.266E-08	2.021E-07
	Rank	2	2	3	1	2	2
CMAES	Mean	-9.873E+07	4.950E+01	4.607E+00	7.395E-04	5.167E-03	1.639E-03
	Std	8.547E+08	1.229E+01	8.725E+00	2.389E-03	7.338E-03	4.196E-03
	Rank	1	10	10	2	6	8
G3PCX	Mean	-2.577E+03	1.740E+02	1.352E+01	1.127E-02	4.593E+00	2.349E+01
	Std	4.126E+02	3.199E+01	4.815E+00	1.310E-02	5.984E+00	2.072E+01
	Rank	13	13	13	4	13	13

Effect of the Step length: In this section, we will discuss the effect of the step length σ on the performance of our proposed approach. We call the above version of DEAL is as Option 1 where $\sigma_1 = U(0, 1)$ and $\sigma_2 = 0.5$. We tested 3 other options as follows:

- Option 2: $\sigma_1 = 1$ and $\sigma_2 = 0.5$
- Option 3: $\sigma_1 = U(0, 1)$ and $\sigma_2 = U(0, 0.5)$
- Option 4: $\sigma_1 = 1$ and $\sigma_2 = U(0, 0.5)$

Table 3. Obtained results for all test problems from all options of DEAL

		F1	F2	F3	F4	F5	F6
Option1	Mean	-1.085E+04	3.482E-15	3.097E-07	1.305E-02	3.619E-07	1.099E-04
	Std	3.996E+02	5.888E-15	8.720E-07	1.721E-02	1.268E-06	1.099E-03
Option2	Mean	-1.064E+04	8.419E-05	3.977E-05	1.546E-02	1.063E-03	6.594E-04
	Std	3.479E+02	5.599E-04	3.433E-04	1.737E-02	1.037E-02	2.622E-03
Option3	Mean	-1.097E+04	1.785E-14	4.213E-06	1.350E-02	6.433E-06	8.790E-04
	Std	3.028E+02	1.594E-14	4.641E-06	1.423E-02	3.494E-05	2.996E-03
Option4	Mean	-1.064E+04	8.419E-05	3.977E-05	1.546E-02	1.063E-03	6.594E-04
	Std	3.479E+02	5.599E-04	3.433E-04	1.737E-02	1.037E-02	2.622E-03

We can observe from Table 3 that there were no large change in the results obtained by all options, except the slightly better results of Option 1. This indicates that the use of either random or fixed value of σ does not significantly effect on the performance of DEAL.

5 Conclusion

In this paper, we introduced a novel technique for employing directions of improvement for EAs; we call it *a direction-guided evolutionary algorithm*. With this new algorithm, a population of solutions is evolved over time under guidance of directions of improvement. At each generation, there are two types of directions are generated: (1) convergence direction between an elite solution and a solution from the current population, and (2) spreading direction between two elite solutions in ETS. These directions are then used to perturb the current population to get a temporary population of offsprings. The combination (combined population) of this offspring population and the current ETS is used to generate the next content of ETS and the main population.

A case study has been carried out to investigate the performance and behaviour of our newly proposed algorithm. We also validated its performance with 12 other well-known algorithms in the field. Our algorithms showed a good performance in comparison with these algorithms.

Acknowledgement. The authors gratefully acknowledge the financial support from the Vietnam Institute for Advanced Study in Mathematics (VIASM) and the University of New South Wales at Australian Defence Force Academy.

References

1. Back, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
2. Corne, D., Dorigo, M., Glover, F.: New Ideas in Optimization. McGraw Hill, Cambridge (1999)
3. Dasgupta, D.: Artificial Immune Systems and Their Applications. Springer, Berlin (1998)
4. Deb, K.: Multiobjective Optimization using Evolutionary Algorithms. John Wiley and Son Ltd., New York (2001)

5. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
6. Deb, K., Anand, A., Joshi, D.: A computationally efficient evolutionary algorithm for real-parameter optimization. Evolutionary Computation 4, 371–395 (2002)
7. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, USA (2004)
8. Fogel, L.J., Angeline, P.J., Fogel, D.B.: An evolutionary programming approach to self-adaptation in finite state machines. In: McDonnell, J.R., Reynolds, R.G., Fogel, D.B. (eds.) Proc. of Fourth Annual Conference on Evolutionary Programming, pp. 355–365. MIT Press, Cambridge (1995)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
10. Goldberg, D.E.: The design of innovation: lessons from and for competent genetic algorithms. Kluwer Academic Publishers, Massachusetts (2002)
11. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9, 159–195 (2001)
12. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
13. Albert, Y.S., Lam, V.O.K.: Li, and James J.Q. Yu. Real-coded chemical reaction optimization. IEEE Transactions on Evolutionary Computation (accepted for publication, 2012)
14. Larraanaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Norwell (2002)
15. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, London (1996)
16. Mitchell, T.: Machine Learning. McGraw Hill, Singapore (1997)
17. Price, K., Storn, R., Lampinen, J.: Differential Evolution - A Practical Approach to Global Optimization. Springer, Berlin (2005)
18. Rudolph, G.: Evolution strategy. In: Handbook of Evolutionary Computation. Oxford University Press (1997)
19. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report tr-95-012. Technical report, ICSI (1995)
20. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3(2), 82–102 (1999)

Introduction of a Mutation Specific Fast Non-dominated Sorting GA Evolved for Biochemical Optimizations

Susanne Rosenthal, Nail El-Sourani, and Markus Borschbach

University of Applied Sciences, FHDW
Faculty of Computer Science, Chair of Optimized Systems,
Hauptstr. 2, D-51465 Bergisch Gladbach, Germany
{Susanne.Rosenthal,Markus.Borschbach}@fhdw.de

Abstract. In many physiochemical and biological phenomena, molecules have to comply with multiple optimized biophysical feature constraints. Mathematical modeling of these biochemical problems consequently results in multi-objective optimization. This study presents a special fast non-dominated sorting genetic algorithm (GA) incorporating different types of mutation (referred to as MSNSGA-II) for resolving multiple diverse requirements for molecule bioactivity with an early convergence in a comparable low number of generations. Hence, MSNSGA-II is based on a character codification and its performance is benchmarked via a specific three-dimensional optimization problem. Three objective functions are provided by the BioJava library: Needleman Wunsch algorithm, hydrophilicity and molecular weight. The performance of our proposed algorithm is tested using several mutation operators: A deterministic dynamic, a self-adaptive, a dynamic adaptive and two further mutation schemes with mutation rates based on the Gaussian distribution. Furthermore, we expose the comparison of MSNSGA-II with the classic NSGA-II in performance.

Keywords: multi-objective biochemical optimization, characater-encoded GA, mutation variants, aggregateselection.

1 Introduction

Especially pharmaceutical research comprises multiple diverse requirements for molecule bioactivity which needs to be optimized simultaneously. The process of resolving the resulting objective functions is termed multi-objective (m.-o.) optimization. Unfortunately, these objective functions usually range from competing to contradictory: An optimal solution with respect to one condition often yields inacceptable results with respect to the other conditions. In most cases an overall optimum for all conditions is non-existent. In the last 20 years, a variety of m.-o. GAs has been developed for these kinds of optimization problems and are widely used tools nowadays, proving themselves as effective and robust solving methods. The performance of this optimization process is characterized by three components: recombination, mutation and selection. Thus, a m.-o. GA can be configured in many ways. However, different configurations expose strong effects on the quality of the solutions. The most important component of a GA is the crossover operator [31]. The main goal of this operator is to exploit parts of good solutions with regard to the different conditions to generate new non-dominated solutions in unexplored parts of the Pareto front. Mutation generally improves diversity of the solution set, but it may also neglect the important (because responsible) parts of a good solution. In classic GA implementations the mutation rate is usually constant, small and it depends on the length of the individuals [8], [34], [35]. The suitable choice of components and component-specific parameters in a m.-o. GA is a challenging task, mostly defined by empirical analysis. Various works have been published, proposing guidelines and recommendations for a quantity of m.-o. optimization problems (recent works [20],[16]). In this work, we present a m.-o. GA based on NSGA-II in its procedure, but it optionally includes for the first time (to the best of our knowledge) three known mutation methods and two further ones based on the Gaussian distribution as well as a newly introduced selection function termed 'Aggregate Selection'. This GA is especially evolved for optimization of molecular features by customizing NSGA-II in the components encoding, selection and mutation. Corresponding to the principle of a most intuitive encoding for an optimized GA performance, a character-string encoding is implemented Furthermore, the the different mutation variants in MSNSGA-II are compared to each other in performance. Three objective functions have been selected from the BioJava library to classify these performance. Additionally, the performance of MNSGA-II for this optimization problem has been compared to the one of NSGA-II.

2 Review of m.-o. GAs

The classic GA has been invented by the inspiration to imitate the evolution process of living things (Holland [1]). Goldberg [15] proved that the GA is able to successfully apply to a real engineering problem. A solution of the optimization problem is termed *individual* and is assumed to be a binary string. The collections of individuals is termed *population* and the start population is normally randomly initialized. Rosenberg first discerned the capability of the traditional GA to solve m.-o. optimization problems in the 1960's. The aim of m.-o. optimization problems is to find a set of adequate good solutions that suit the predefined conditions sufficiently. In the m.-o. sense, a good solution is a n-dimensional vector which is non-dominated by any other solution, meaning that at least one vector component is superior. The *Pareto front* is formed by solution vectors which are non-dominated by other solutions of the search area. The main goal of the m.-o. optimization process is to yield solutions, which approximate the Pareto front as well as possible. The single-objective GA can be modified to simultaneously search the solution space for non-dominated solutions. The first m.-o. GA was introduced by Schaffer [2], termed vector evaluated GA (VEGA), followed by the m.-o. genetic algorithm (MOGA) [10]. Commonly used GAs for m.-o. optimization are the Non-dominated Sorting GA (NSGA) [3] and the Fast Non-dominated Sorting GA (NSGA-II) [4]. Zitzler and Thiele proposed an elitist evolutionary algorithm termed Strength Pareto Evolutionary Algorithm (SPEA)

([28], [27]). Ten years later Zitzler, Laumann and Thiele proposed a variant of SPEA termed SPEA2 [29]. These two variants differ in fitness assignment and a tournament operator. State-of-the-art m.-o. GAs are IBEA [11] and SEAMO2 which reveal remarkable performance. In the field of m.-o. problems with highly correlated objectives, MOEA/D achieved remarkable results [30]. The paper of Konak et. al. [5] provides a comparative overview of the well-known m.-o. GAs. A lot of research has been published analyzing and benchmarking the performance of various m.-o. GAs: In [13], the hybrid adaptive method MO-SHERPA [14] is tested on the standard benchmark problem ZDT functions compared to NSGA and NSGA-II. MO-SHERPA dramatically outperforms these two GAs. Another related works are [27], [30] and [26]. Related works, using single-objective GA to optimize biological activity and other molecular targets, report that the configuration of an unusual low number of generations (partly under ten) is sufficient [32], [33]. This fact is termed early convergence in this paper. Unfortunately, there is not sufficient literature available for application of m.-o. GAs in the simultaneously optimization area of molecular features [23], [24], [25].

3 Introduction of MSNSGA-II

As MSNSGA-II has especially been evolved for application in molecular biology, the individuals are encoded as character-strings instead of the GA-typical binary encoding. The character-string encoding symbolizes the 20 canonical amino acids. This has two advantages: It suits the proposed optimization problem in a natural way and avoids a possible drawback of the binary encoding - unfavorable breakpoints. Hence, individuals are implemented as 20-character strings. As MSNSGA-II corresponds to NSGA-II in its procedure, a brief description is given in the following. Furthermore, only the components of MSNSGA-II differing to the ones of NSGA-II are described below.

NSGA-II. The fast elitist m.-o. GA was first introduced by Deb et. al. [4]. It finds many solutions well-spread across the Pareto optimal front in a low number of computation operations ($O(mN^2)$, where m is the number of objectives and N the population size). NSGA-II incorporates an elite operator to preserve and utilize previously found good solutions in the subsequent generations, the non-dominated sorting concept [3], [15] and the crowded tournament selection operator to preserve the diversity of non-dominated solutions up to a large number of generations in order to obtain a good distribution of solutions over the Pareto front. This mechanism provides that solutions in the same non-dominated front are selected by their crowding distance, a measure of solution density in the immediate environment. A solution in a less crowded area, hence a larger crowding distance is prioritized by the selection operator. The crowding distance is a critical component and encouraging research work has been done to improve the performance of m.-o. GAs by variations of the crowding distance scheme[17]. A practical problem customized best application practice has not been established yet and is subject to ongoing research [19].

Fitness Calculation. As mentioned above, three functions of the BioJava library are selected as objective functions. The encoding of the individuals as character strings enables the use of *Needleman Wunsch* algorithm (NMW) [7], *Molecular Weight* (MW) and *hydrophilicity* (hydro) [6] as the three objective functions. The fitness values of a string sequence of the length l for MW and hydro are calculated from the amino acids (a_i) for $i = 1, ..., l$ [6]: Molecular weight is computed as the sum of mass of each amino acid plus a water molecule: $\sum_{i=1}^{l} mass(a_i) + 17.0073(OH) + 1.0079(H)$. (According to the periodic system of elements: Oxygen (O), hydrogen (H)) Hydrophilicity: $\frac{1}{l} \cdot (\sum_{i=1}^{l} hydro(a_i))$. The NMW algorithm performs a global sequence alignment of two sequences by an iterative matrix method of calculation. More precisely, NMW as an objective function in MSNSGA-II is a measure in order to represent similarity of an individual to a pre-defined reference individual. For a closer understanding see [7]. The source codes of these three objective functions are available at [6].

Recombination Operator. The n-point recombination described in [37] is used. Three parent are selected and the number n of recombination points is determined on the basis of a Gaussian distribution. Hence, the parameters of the recombination are the actuarial expectation $\rho = 2$ (the most frequent number for n) and the standard deviation $\sigma = 2.5$. Only positive values $n \geq 0$ are permitted: In the cases of negative values, the random generator is restarted.

Mutation Operators. Five different types of mutation operators are compared with regard to the performance of the GA for this m.-o. problem. Non-fixed mutation types are categorized in three classes: deterministic dynamic, dynamic adaptive and dynamic self-adaptive. A promising mutation operator of each class is selected as well as two alternatives, mutation rates which depend on the Gaussian distribution.
Bäck and Schütz introduced the following deterministic dynamic mutation operator [8]. The mutation rates are calculated by the function with $a = 2$

$$p_{BS} = (a + \frac{l-2}{T-1}t)^{-1}, \tag{1}$$

whereas T is the total number of generations the GA is run for, l is the length of the individual and t the actual number of generation. The basic idea of this operator is that higher mutation rates in early generations lead to a good exploration and the lower rates in later generations provide a good exploitation of the local fitness landscape. The mutation rate is bounded by $(0; \frac{1}{2}]$. As the GA includes both components, mutation and recombination, the function of Bäck and Schütz has been adapted to a lower initial mutation rate: $a = 4$ in (1). The mutation rate here is bounded by $(0; \frac{1}{4}]$. Both functions will be compared with regard to the solution-quality.
Thierens introduced the dynamic adaptive mutation scheme [9]. The goal is to try three different mutation rates on the current individual. The comparison of the fitness values of the three offspring gives a rough hint, whether the current mutation rate should be increased or decreased. The modification of the current mutation rate is carried out proportionally by multiplying or dividing the cur-

rent rate with the constant learning factor α. During the evaluation, a factor ω called exploration factor is used. Usually $\omega > \alpha > 1$ to avoid oscillations of the mutation rates. Formally $M(x, p_m) \longrightarrow (x^*, p_m^*)$ which means, that the individual x with mutation rate p_m generates the offspring x^* with the new mutation rate p_m^*. The mutation scheme of Thierens:

1. Mutate the current individual (x, p_m):
   $M(x, p_m/\omega) \longrightarrow (x_1, p_m/\alpha)$
   $M(x, p_m) \longrightarrow (x_2, p_m)$
   $M(x, p_m \cdot \omega) \longrightarrow (x_3, p_m \cdot \alpha)$
2. Select the fittest individual of
   $\{(x, p_m), (x_1, p_m/\alpha), (x_2, p_m), (x_3, \alpha \cdot p_m)\}$

Appropriate values are $\alpha = 1.1$ and $\omega = 1.5$. The start mutation rate is selected as $p_0 = 0.2$. The self-adaptive mutation scheme for binary strings was also proposed by Bäck and Schütz [8]. Self-adaption gives no direct feedback to the quality of the mutation rate. The goal of this operator is that individuals with good parameters receive an evolutionary advantage. The mutation rate is calculated by the function:

$$p_m(t+1) = (1 + \frac{1 - p_m(t)}{p_m(t)} \cdot e^{-\gamma N(0,1)})^{-1}, \tag{2}$$

where $N(0, 1)$ is a normal distributed random number and the learning rate γ controls the adaption steps of the mutation rate. A traditional choice for the learning rate is $\gamma = 0.22$. The start population rate is also selected as $p_0 = 0.2$. Two further mutation operators, referred as *Random* and *AAweighted* in the following are based on the Gaussian distribution, similar to the recombination: The number of mutations of each individual is chosen randomly, according to the Gaussian distribution. The difference between these two operators is the selection of the characters. In AAweighted, each of these 20 characters (canonical amino acids) has its specific frequency to be mutated to (according to their natural incidence). So the selection here is implemented as Stochastic-Universal-Sampling. In Random - mutation, each character has the same mutation-probability to be inserted. In MSNSGA-II the choice of the Gaussian interval length σ and the actuarial expectation ρ for both mutation variants is set to: $\sigma = 5$ and $\rho = 0.2$

Selection Operator. Taking the standard NSGA-II proposed selection method as base the individual diversity rapidly declined in early consecutive test-runs. This induced the need of careful examination and tweaking of the selection function to preserve a high individual diversity in potential parent selection to not undermine the important aspect of solution space broadness in GA optimization. The specialty in this GA application is posed by the low number of overall generations to be calculated as well as an overall low number of individuals generated (due to manual fitness evaluation) while still aiming for high quality solutions. Common for peptide design approaches the fitness-value imposed weakness of single individuals is not representative of their sub-qualities that can be induced to later generation individuals via recombination. Thus, one main goal of the aggregateselection-operator is to just very slightly steer in the "right" direction regarding the solution space. Suppose μ individuals are to be selected:

```
1. select x (= tournament size) random individuals from population
2. Pareto-rank the tournament individuals
   (a) 0.5 probability
   (b) preselect front 0
   (c) randomly chose one to put into selection pool

       or opposite

   (a) 0.5 probability
   (b) SUS by front size to preselect individuals frome some front
   (c) randomly chose one to put into selection pool
3. if (selection pool size = $\mu$) then done else back to 1.
```

This pseudo-code demonstrates the workings of the standard aggregateselection-operator. However, during development the need to prioritize one of the multi-objective optimization goals emerged. To achieve this while allowing current, supposedly bad individuals to survive, the procedures above were changed each at (c); instead of randomly accepting one individual from the preselection-pool to the selection pool, the individual with the better fitness regarding the prioritized objective is chosen to be put into the selection pool. This selection method is a product of extensive analysis and different μ, λ sizes are subject to ongoing work. The principle selection workflow was derived from earlier work, developed for solving the Rubik's Cube's discrete optimization problem [38].

4 Experiments

This section provides the series of experiments comparing on the one hand different mutation settings for MSNSGA-II and on the other hand a benchmarking with NSGA-II. The remaining parameter settings are the same:

Start mutation rate: $p_0 = 0.2$ except for the adapted dynamic mutation scheme of Bäck and Schütz; the parameters of recombination: $\rho = 2$, $\sigma = 2.5$. The start population has a size of 100 randomly initialized individuals of 20 characters.

The three objective functions used for these experiments present a real-life problem: Simultaneously optimization - in this case minimization - of Needleman Wunsch, representing a sequence alignment and two molecular features, molecular weight and hydrophilicity. The main goal of these experiments is to exhibit the early convergence of MSNSGA-II in a special m.-o. biophysical application. Each configuration of MSNSGA-II is run 30 times until the 18-th generation. Due to the high costs of determining fitness values in practice, the evolution process is limited to only a small number of generations and thereby realizes an early convergence. The objective function values reflect the distance of the individuals' objective function values to the one of a non-varying reference individual.

M.-o. optimization has two different aims: The convergence of the population to the Pareto optimal front and the diversity of the solution set. Therefore, as a measure of density the Spacing-metric of Deb [4] is used. This metric states a measure for uniform distribution of the population in the feasible region:

$$SD = \frac{\sum_{i=0}^{n} |d_i - \bar{d}|}{n} \tag{3}$$

whereas n denotes the number of individuals of the generation and $\bar{d}$ is the average distance of all individuals. The S-metric or hypervolume, first proposed by Zitzler [36] is utilized for measuring the convergence. The S-metric measures the size of the objective space spanned by a set of non-dominated solutions

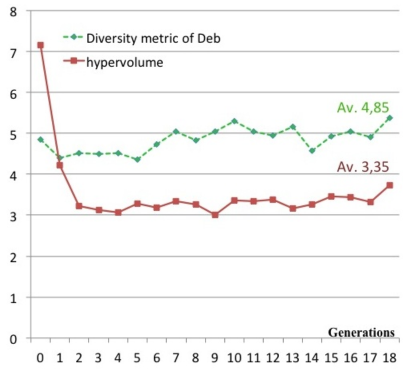


Fig. 1. Performance of MSNSGA-II with mutation variant Bäck& Schütz

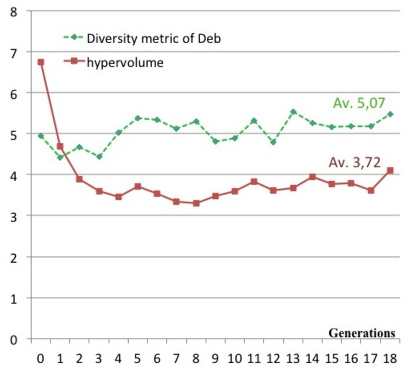


Fig. 2. Performance of MSNSGA-II with mutation variant Random

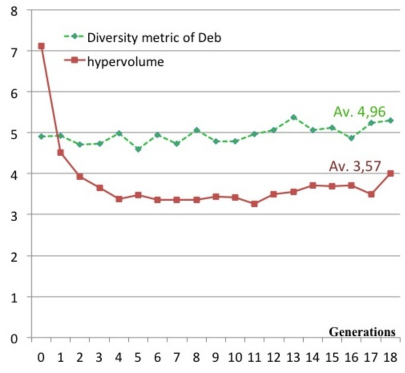


Fig. 3. Performance of MSNSGA-II with mutation variant AAweighted

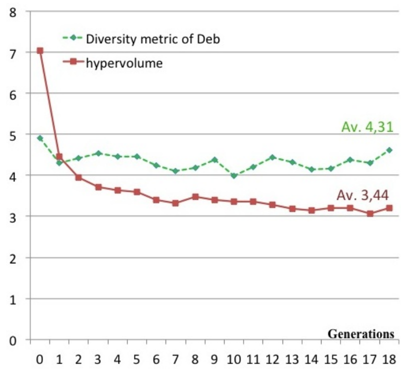


Fig. 4. Performance of MSNSGA-II with self-adaptive mutation, $\gamma = 0.35$

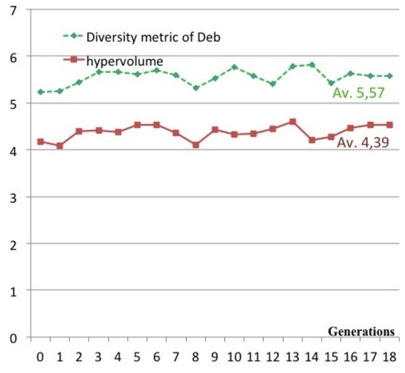


Fig. 5. Performance of MSNSGA-II with Thierens constant gain mutation

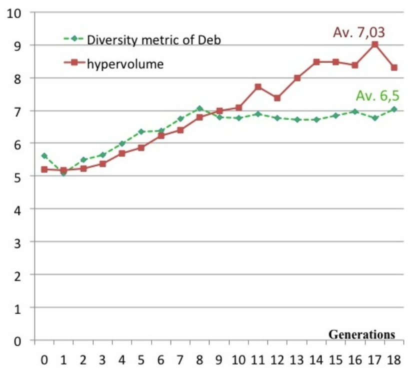


Fig. 6. Performance of character-encoded NSGA-II

and a pre-defined reference point. As empirical results, the Spacing-metric and the hypervolume with reference point $\mathbf{0}$ is determined of each generation. The average metric values over 30 runs for each configuration are depicted in figure 1 to 6. The metric values are scaled.

Apparently, MSNSGA-II is able to obtain a good solution set within the 18 generations, especially with the dynamic deterministic mutation scheme of Bäck and Schütz (Fig. 1). MSNSGA-II with the self-adaptive mutation scheme of Bäck and Schütz (Fig. 4) achieves better convergence results at the cost of diversity. Different values for the control parameter γ in the self-adaptive scheme were tested, the best performance was achieved with $\gamma = 0.35$. The configuration with the mutation Random (Fig. 2) features oscillating behavior although converging. The similar mutation AAweighted (each character has its specific frequency to be mutated) decreases this oscillating behavior (Fig. 3). The mutation scheme of Thierens (Fig. 5) yields the worst performance in the sense of m.-o.. By way of comparison, the results of the character-encoded NSGA-II run are depicted in Fig. 6: In all trials NSGA-II exhibit no convergence within the 18 generations.

The comparison of the metric values over all MSNSGA-II runs discloses the following hypothesis: The runs with almost constant metric values below or equal 4.5 realize the most effective convergence. Constant high metric values above 5.0 result in oscillating convergence behavior.

5 Conclusion

This paper presents a special m.-o. GA based on NSGA-II in its procedure, but it is especially adapted to optimize diverse molecular features. Hence, MSNSGA-II is character-encoded and the peculiar achievement is effected by the interaction of a newly introduced selection function and different types of mutation variants. MSNSGA-II was applied on a synthetic three-dimensional optimization problem. Running MSNSGA-II with the different mutation settings on these three objective functions from the BioJava library reveal on the one hand that a good solution set in m.-o. sense is achieved within 18 generations. On the other hand, all configurations of MSNSGA-II featured convergence behavior, though some exhibit oscillating behavior. The comparison of the experimental results disclose throughout good convergence behavior of MSNSGA-II with the mutation variants of Bäck and Schütz. A few insights could be gain with regard to the metric values defined by Deb: Steady good convergence behavior can be expected with constant metric values throughout all generations. As this approach seems interesting for biochemical research, a closer look is necessary to gain deep inside understanding of the convergence velocity. After this empirical study, theoretical analysis of the different configurations is planned. Other fields of research are the state-of-the-art GAs IBEA, SEAMO2 and MOEA/D promising remarkable results. But it is an open issue if these GAs adequately perform like MSNSGA-II by adapting their encoding respectively their other components. Additional research will concentrate on improving the mutation and recombination of MSNSGA-II. Furthermore, we have to focus on to the critical component crowding distance and we will analyze the convergence performance of possible alternatives via optimization of crowding-distance behavior.

References

1. Holland, J.H.: Adaption in natural and artificial systems, pp. 287–299. University of Michigan Press, Ann Arbor (1975)
2. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proceeding of the International Conference on the Genetic Algorithm and Their Applications (1985)
3. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. J. Evol. Comput. 2(3), 221–248 (1994)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 182–197 (2002)
5. Konak, A., Coit, D.W., Smith, A.E.: Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety 91, 992–1007 (2006)
6. BioJava: CookBook, release 3.0, http://www.biojava.org/wiki/BioJava
7. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48(3), 443–453 (1970)
8. Bäck, T., Schütz, M.: Intelligent mutation rate control in canonical genetic algorithm. In: Proc. of the International Symposium on Methodology for Intelligent Systems, pp. 158–167 (1996)
9. Thierens, D.: Adaptive mutation rate control scheme in genetic algorithm. In: Proceedings of the 2002 IEEE World Congress on Computational Intelligence: Congress on Evolutionary Computation, pp. 980–985 (2002)
10. Fonseca, C.M., Fleming, P.J.: Multiobjective genetic algorithm. In: IEE Colloquium on Genetic Algorithms for Control Systems Engineering (Digest No. 1993/130), London, May 28, pp. 6/1–6/5 (1993)
11. Zitzler, E., Künzli, S.: Indicator-Based Selection in Multiobjective Search. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
12. Mumford-Valenzuela, C.L.: A Simple Approach to Evolutionary Multi-Objective Optimization. In: Evolutionary Computation Based Multi-Criteria Optimization: Theoretical Advances and Applications. Springer (2004)
13. Chase, N., Rademacher, M.: A benchmark of multi-objective optimization methods. Red chedar Technology, BMK-3021 Rev. 06.09
14. HEEDS of ProSIM, http://www.pro-sim.com/heeds.html
15. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, p. 432. Addison-Wesley, Reading (1989)
16. Kötzinger, T., Sudholt, D.: How crossover helps in Pseudo-Boolean Optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, pp. 989–996 (2011)
17. Li, M., Zheng, J., Wu, J.: Improving NSGA-II Algorithm Based on Minimum Spanning Tree. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 170–179. Springer, Heidelberg (2008)
18. Liu, H., Gu, F.: A improved NSGA-II algorithm based on sub-regional search. In: IEEE Congress on Evolutionary Computation, pp. 1906–1911 (2011)
19. Luo, B., Zheng, J.: Dynamic crowding distance? A new diversity maintenance strategy for MOEA's. In: Fourth International Conference on Natural Computation, ICNC, vol. 1, pp. 580–585 (2008)

20. Sato, H., Aquire, H.: Improved S-CDAS using Crossover Controlling the Number of Crossed Genes for Many-Objective Optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, pp. 753–760 (2011)
21. Takahashi, R., Vansconcelos, J.A.: A multiobjective methodology for evaluating genetic operators. IEEE Transactions on Magnetics 39(3), 1321–1324 (2003)
22. Van Veldhuizen, D.A., Lamont, G.: Evolutionary computation and convergence to a Pareto front. Stanford University, Califoria, CiteSeerX, pp. 221–228 (1998)
23. Suzuki, H., Sawai, H.: Chemical genetic Algorithms - Coevolution between Codes and Code translation. Artificial Life VIII, pp. 164–172. MIT Press (2002)
24. Ekins, S., Honeycutt, J.D.: Evolving molecules using multi-objective optimization: applying to ADME/Tox. Grug Discovery Today 15(11/12), 451–460 (2010)
25. Yamamichi, S., Kurata, H.: Optimization of a Large-Scale Dynamic Model of the Cell Cycle Network using Multi-Objective Genetic Algorithm. Genome Informatics 16 (2005)
26. Grosan, C., Dumitrescu, D.: A Comparison of Multi-Objective Evolutionary Algorithms. Acta Universitatis Apulensis (2010)
27. Zitzler, E., Deb, K., Thiele, L.: Comparison of multi-Objective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
28. Zitzler, E., Thiele, L.: An evolutionary algorithm for multiobjective optimization: The strength Pareto approach. Technical report 43, Computer engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology, ETH (1999)
29. Zitzler, E., Laumann, M., Thiele, L.: Improving the Strength Pareto Evolutionary Algorithm. Technical report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich (2001)
30. Ishibuchi, H., Hitotsuyanagi, Y., Ohyanagi, H., Nojima, Y.: Effects of the Existence of Highly Correlated Objectives on the Behavior of MOEA/D. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 166–181. Springer, Heidelberg (2011)
31. Deb, K., Agrawal, S.: Understanding interactions among genetic algorithm parameters. Foundations of Genetic Algorithms 5, 265–286 (1999)
32. Singh, J., Ator, M.A., Jaeger, E.P.: Application of genetic Algorithms to Combinatorial Synthesis: A Computational Approach to Lead Identification and Lead Optimization. J. Am. Chem. Soc. 118, 1669–1676 (1996)
33. Weber, L., Wallaum, S., Broger, C.: Optimizing biological activity of combinational compound libraries by a genetic algorithm. Optimierung der biologischen Aktivität von kombinatorischen Verbindungsbibliotheken durch einen genetischen Algorithmus. Angew. Chem. 107, 2452–2454 (1995) (in German)
34. Zarges, C.: Rigorous Runtime Analysis of Inversely Fitness Proportional Mutation Rates. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN X. LNCS, vol. 5199, pp. 112–122. Springer, Heidelberg (2008)
35. Nijssen, S., Bäck, T.: An analysis of the behaviour of simplified evolutionary algorithms on trap functions. IEEE Trans. on Evol. Comp. 7(1), 11–22 (2003)
36. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN V. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
37. Borschbach, M.: Neural Classification of Biological Properties and Genetic Operators Configuration Issues. Trans. on Information Science 12(2), 324–329 (2005) ISSN 1790-0832
38. El-Sourani, N., Borschbach, M.: Design and Comparison of two Evolutionary Approaches for Solving the Rubik's Cube. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 442–451. Springer, Heidelberg (2010)

Using Hybrid Dependency Identification with a Memetic Algorithm for Large Scale Optimization Problems

Eman Sayed, Daryl Essam, and Ruhul A. Sarker

School of Engineering and Information Technology at Australian Defence Force Academy, UNSW, Canberra, Australia
e.hasan@student.adfa.edu.au,
{d.essam,r.sarker}@adfa.edu.au

Abstract. Decomposing a large scale problem into smaller subproblems is one of the approaches used to overcome the usual performance deterioration that occurs in EA because of the large dimensionality. To achieve a good performance with a decomposition approach, the dependent variables need to be grouped into the same subproblem. In this paper, the Hybrid Dependency Identification with Memetic Algorithm (HDIMA) model is proposed for large scale optimization problems. The Dependency Identification (DI) technique identifies the variables that must be grouped together to form the subproblems. These subproblems are then evolved using a Memetic Algorithm (MA). Before the end of the evolution process, the subproblems are then aggregated and optimized as a complete large scale problem. A newly designed test suite of problems has been used to evaluate the performance of HDIMA over different dimensions. The evaluation shows that HDIMA is competitive to other models in the literature in terms of both consuming less computational resources and better performance.

Keywords: Large Scale Problems Optimization, Evolutionary Algorithms, Memetic Algorithms, Problem Decomposition, Dependency Identification.

1 Introduction

Solving large scale optimization problems has become a challenging area, due to both the increased need for high quality decision making for large scale optimization problems in real life; and also the need to optimize the available computing resources. Recently developed optimization algorithms use Evolutionary Algorithms (EAs) to solve large scale optimization problems. However, the performance of EAs eventually deteriorates with the increased dimensionality of optimization problems [1]. There are two solutions to overcome this dimensionality problem; applying a decomposition approach, and using a hybridized approach. The first attempt for applying the decomposition approach is the Cooperative Coevolution (CC) approach that breaks the large scale problem into smaller subproblems [2]. The CC approach loses its efficiency with nonseparable optimization problems [2]. If one variable is optimized in one subproblem and it depends on other variables which are optimized in different subproblems, the overall performance will deteriorate [2]. These variables are known as interdependent variables. To improve the performance of the CC, the separation of these interdependent variables need to be minimized. This demands a new technique that would detect the best arrangement of variables by minimizing the separation of the interdependent variables. The recently developed grouping approaches cannot provide a complete grouping for the interdependent variables which should be optimized in one subproblems [2]. This motivates the development of an optimization model that hybridizes the CC approach and the complete large problem optimization approach. To overcome the dimensionality drawback of EA when solving large scale optimization problems, it has been hybridizing with Local Search (LS) technique [3]. LS is one of these techniques that can be hybridized with EAs to enhance its performance. The hybridizations of EA and LS are called Memetic Algorithms (MA).

A newly developed model, Hybridized Dependency Identification with Memetic Algorithm (HDIMA), is proposed in this paper and its performance is investigated over different large scale dimensions. HDIMA hybridizes the decomposition and the aggregation approaches. In the proposed models, the large scale problem is decomposed into subproblems and is then optimized for two thirds of the evolution process, and then the subproblems are aggregated into one problem for the final third. HDIMA performance is evaluated on a recently designed test suite of 12 unconstrained large scale optimization problems. A copy of this test suite can be downloaded from the author's page[1]. The results are analyzed against other models in the literature and show the competitive performance of HDIMA over the 1000 dimension scenarios and the saving of computational resources for the 4000 dimension set, but unfortunately this saving decreased the performance on the larger dimension.

Section 2 of this paper is a literature review of grouping techniques. Section 3 discusses the proposed model. The Experiment and results are presented in sections 4 and 5. Finally, the conclusion and future work is in section 6.

2 Literature Review

The first attempt for problem decomposition is the Cooperative Coevolution (CC) approach [2]. Problem decomposition helps to enhance the performance of the optimization algorithms by decomposing the large scale problems into small subproblems. The initial strategies that were developed for CC are the one-dimension based strategy, and the splitting-in-half strategy [4]. The one-dimension based strategy decomposed a N dimension large scale problem into N subproblems. The second approach produces two subproblems of size $\frac{N}{2}$ (which are large scale subproblems if N is very large). The common decomposition approach is to decompose the large scale problem into predefined subproblems. The subproblem size of 100 has been found to be convenient for both the separable and the nonseparable problems [5]. Besides the problem size, the performance of the optimization model depends also on identifying

[1] `https://docs.google.com/file/d/0B7q30xCpYVy3R1dkNkxzdHhMUWM/edit?pli=1`

the variables dependency. If the dependent variables are grouped into different subproblems, the performance of the optimization algorithm will decrease. These variables are referred to in this paper as interdependent variables. This motivates the development of decomposition techniques that consider grouping the dependent variables into subproblems to minimize the interdependent variables.

One of the recently developed decomposition techniques is Random Grouping (RG) [5], where variables are grouped randomly into smaller subproblems. RG has achieved good performance in the literature [1, 5], but it does not have a systematic way to group the variables or detecting their dependencies. The grouping technique of correlation based Adaptive Variable Partitioning [6] was developed afterwards. This technique doesn't detect the nonlinear dependencies amongst the variables and also uses large computational resources. Following that technique, the Delta Grouping technique [7] was developed. For it, the variables are grouped according to the magnitude of change that happens through one generation during the evolution of the subproblems. Although this technique seems to be reasonable, it is less efficient when the large scale problem has more than one nonseparable group [7]. A recently developed grouping technique is the Variable Interactive Learning (VIL) [8] technique. In that, the large scale optimization problem is decomposed into one-dimension subproblems and the current and previous populations are tested after optimizing each subproblem. After that, the subproblems which have changed and affected each other are merged. One problem with this, is that starting with one-dimension based decomposition is not suitable for large scale problems. Moreover, VIL consumes up to 60% of the available computational resources [8]. The most recently developed technique is the Dependency Identification (DI) [9] which groups the variables in an arrangement that minimizes the interdependent variables. DI uses approximately 1.16% of the total evaluations and follows a systematic way to group the variables. However it can't group all the dependent variable in one subproblem if they exceed the allowed subproblem size. These drawbacks in the grouping techniques in the literature and the requirement to minimize the interdependent variables, motivates the need for hybridizing the decomposition and the aggregation approaches into one model. Furthermore, increasing the subproblems size will decrease the interdependent variables.

3 Proposed Model

The three main stages of HDIMA are dependency identification and decomposition; subproblems optimization and information exchange; and subproblems aggregation and optimization. This model can decrease the computational resources that need to be used for dependency identification and information exchange. The detailed pseudo code of the HDIMA is shown in Fig. 1.

3.1 Dependency Identification (DI) and Problem Decomposition

The first stage of HDIMA applies the DI technique [9] for two thirds of the evolution process which in this paper is two thirds of the Fitness Evaluations (FE). The DI technique is inspired from the definitions of problem separability in the literature [8, 10]. A fully separable problem [10] is one that can be written in the form of linear combinations of subproblems of individual variables, where $F(x) = \sum_{i=1}^{N} f(x_i)$. A partially separable problem is defined in [6] as $F(x) = \sum_{k=1}^{m} f_k(x_v)$, $v = [1, V], N = m * V$, where N is the total number of variables of the large scale optimization problem, $F(x)$, which is decomposed into m equal size subproblems. Each subproblem has V dependent variables and no variable is represented in more than one subproblem.

From these definitions it is obvious that the best grouping to decompose a large scale optimization problem into subproblems is one that minimizes the number of interdependent variables, where interdependent variables are variables that have more than one instance in two or more subproblems. The DI technique finds the arrangement of variables that produces the least square difference (sq_{diff}) between $F(x)$ and $\sum_{k=1}^{m} f_k(x_v), v = [1, V]$, as defined in equation (1). This is done by running a simple randomized search. $F(x)$ is m multiplied by the total of the two solutions of all the variables, $x_i = c_1, \forall\ i = [1, N], and\ c_1 \neq 0$, and all the variables $x_i = c_2, \forall\ i = [1, N], and\ c_2 \neq 0$. The N variables are arranged and decomposed into m subproblems, where each subproblem consists of V variables. Also $\sum_{k=1}^{m} f_k(x_v)$ is the total of $f\left(x_{v,c_1}\right)$ and $f\left(x_{v,c_2}\right)$ for all the m subproblems. To do this, the V variables of one subproblem k are set to c_1, and the rest of the $(N - V)$ variables are set to the other value c_2, and vice versa. Lastly, and while recalling that the technique aims to minimise sq_{diff}:

$$sq_{diff} = [F(x) - \textstyle\sum_{k=1}^{m} f_k(x_v)]^2 \ , v = [1, V] \tag{1}$$

$$x_{v,c_1} = \begin{cases} c_1\ \forall\ v = [1, V] \\ c_2\ otherwise \end{cases} \quad , \quad x_{v,c_2} = \begin{cases} c_2\ \forall\ v = [1, V] \\ c_1\ otherwise \end{cases}$$

3.2 Subproblem Optimization and Information Exchange

The subproblems from the previous stage are optimized at this stage using MA. MA is more reliable than EA [11] and it is capable of searching noisy search spaces efficiently [12]. MA uses Genetic Algorithms (GA) [13] and self-directed LS. The crossover operator in GA is the Simulated Binary Crossover (SBX) [14], and the mutation operator is an adaptive nonuniform mutation [15]. The self-directed LS guides the search to the most promising solution area. It uses variant search steps, d', based on the improvement of the search process as in equation (2). The advantage of using a self-directed local search is that of decreasing the greediness of LS so as to avoid premature convergence [16].

$$d' = \begin{cases} d + e^{\ 1/d} & if\ f_{d'}(x) < f_d(x) \\ d + e^{-1/d} & otherwise \end{cases} \tag{2}$$

Although the DI can find a good arrangement for the subproblems, there is always a probability of having interdependent variables. For example, one subproblem may have an instance of the interdependent variable that is optimized in another subproblem. The values of the variables will thus differ from their initial values during the optimization. An Information Exchange Mechanism (IEM) is used to maintain the representation of only one instance for each interdependent variable during the optimization process. The IEM is activated after optimizing each subproblem to update the value of the interdependent variables that were optimized in that subproblem.

```
1. Initialize population $NP$ for $N$ variables.
2. Create initial sequential arrangement $S_v$ for the $N$ variables
3. Initially decompose the $N$ variables into $m$ subproblems of $s\_size$
4.**for** (swp < 10) randomly swap two variables $x_{r_1}$, $x_{r_2}$ in $S_v$ to get $S_{\bar{v}}$
    $S_v = \{x_1, ..., x_{r_1}, ..., x_{r_2}, ..., x_N\}$,
    $S_{\bar{v}} = \{x_1, ..., x_{r_2}, ..., x_{r_1}, ..., x_n\}$
5. calculate $sq\_diff$ as in equation (1)
6. update $S_v$ to the arrangement that achieved minimum $sq\_diff$
7. follow $S_v$ to decompose the $N$ variables into $m$ subproblems
8. **for** $k = 1$ to $k = m$ **do**
        4.5.1 Optimise the subproblems
        4.5.2 Information Exchange Mechanism
9. If FE< 2*max_FE/3 go to step 4
10. Aggregate the $m$ subproblems into one N_size problem
11. **for** (FE< max_FE) optimize the N_size problem
```

Fig. 1. Pseudo code of DIMA

3.3 Subproblems Aggregation and Optimization

At the last stage of HDIMA, the subproblems are aggregated and optimized using MA as one large scale problem for the final third of the evolution process, and the IEM is deactivated. Subproblems aggregation can overcome the drawback of the grouping techniques which is imposed by the subproblem size limitation. Thus optimizing a complete large scale problem without decomposition is a compromise between the performance, and saving the resources that would have been used for DI and IEM.

4 Experiments

The performance of HDIMA is evaluated on large scale problems to investigate the compromise between slightly lower performance and the ability to save computational resources. Two experiments are carried out on 2 different dimensions, of 1000 and 4000, and HDIMA is compared to DIMA, RGMA, and MA. DIMA is similar to HDIMA but without the aggregation stage. RGMA uses the decomposition technique RG which achieved good performance in the literature [1, 5]. The MA model does not follow the CC decomposition approach. The objectives of the experiments are to investigate how the hybridization of the decomposition and aggregation approaches can decrease the computational resources; and how that decrease may affect the performance when the dimension is increased. These two experiments are conducted on a new test suite of 12 unconstrained large scale optimization problems that are based on the basic 6 problems of CEC2008 [17]. This test suite is published on the author's page. This test suite covers different degrees of separability and overlapping among the variables to represent real life problems where the variables are rarely independent. The experiments have been conducted using the same parameter settings as in the literature[7-8], with population size $NP = 50$, subproblem size (s_size) =100, η=2 as used by most researchers for SBX [18], maximum number of FEs (max_FE) = 3E+6, $c_1 = 1$ and $c_2 = 2$. The results are taken over 25 independent runs for each problem.

5 Results and Analysis

The results of the experiments with 1000 and 4000 dimensions are represented in Table 1 and Table 2 with 3 fraction points. The best value is formatted in bold to distinguish it from other similar approximated results. For D=1000, HDIMA outperformed DIMA in 2 nonseparable problems, 1 *overlapping* nonseparable, 1 *overlap ping* partially nonseparable F_6, and 3 *overlapping spliced* partially nonseparable problems. HDIMA has higher performance than DIMA at 9 problems in terms of standard deviation. HDIMA achieved less performance than that of RGMA for only the overlapping nonseparable problem, F_3, but it was also competitive for the other 11 problems. HDIMA was better at 7 problems in terms of the mean value and 8 problems based on the standard deviation value in comparison to MA. HDIMA achieved better performance than MA at F_1, F_2, F_4, and the overlapping partially nonseparable problems F_6, F_7, F_8, and the overlapping spliced partially nonseparable problems, F_{12}.

Table 1. Models on D=1000

		F1	**F2**	**F3**	**F4**	**F5**	**F6**
	Best	**4.910E+02**	**4.925E+02**	9.927E+02	**5.878E+02**	5.451E+02	**5.449E+02**
HDIMA	Mean	1.799E+04	**1.324E+04**	1.214E+03	**1.017E+04**	5.400E+03	7.846E+03
	Std	9.383E+03	1.086E+04	2.257E+02	6.184E+03	4.808E+03	4.675E+03
	Best	4.912E+02	4.934E+02	9.925E+02	5.887E+02	**5.449E+02**	5.449E+02
DIMA	Mean	2.039E+04	1.110E+05	**1.073E+03**	4.670E+04	**2.367E+03**	9.639E+03
	Std	5.950E+04	1.305E+05	1.649E+02	6.208E+04	4.150E+03	2.961E+04
	Best	4.929E+02	5.015E+02	**9.923E+02**	5.913E+02	5.454E+02	5.453E+02
RGMA	Mean	2.331E+05	2.398E+05	1.272E+03	9.088E+04	1.162E+05	9.515E+04
	Std	1.652E+05	1.492E+05	3.781E+02	7.987E+04	7.469E+04	7.284E+04
	Best	4.912E+02	4.935E+02	9.923E+02	5.881E+02	5.453E+02	5.451E+02
MA	Mean	**1.527E+04**	1.990E+04	1.344E+03	1.013E+04	7.826E+03	**7.380E+03**
	Std	1.193E+04	9.876E+03	3.542E+02	6.131E+03	4.246E+03	5.386E+03
		F7	**F8**	**F9**	**F10**	**F11**	**F12**
	Best	5.450E+02	5.466E+02	5.298E+02	5.237E+02	5.234E+02	**5.244E+02**
HDIMA	Mean	**5.886E+03**	**7.189E+03**	7.251E+03	**6.825E+03**	**5.269E+03**	**6.583E+03**
	Std	4.894E+03	4.708E+03	3.137E+03	3.640E+03	4.357E+03	3.881E+03
	Best	**5.449E+02**	**5.465E+02**	5.295E+02	5.251E+02	5.251E+02	5.249E+02
DIMA	Mean	8.698E+03	3.272E+04	2.356E+04	1.341E+04	3.791E+04	3.318E+04
	Std	2.423E+04	5.319E+04	3.621E+04	2.821E+04	4.199E+04	3.658E+04
	Best	5.499E+02	5.468E+02	5.324E+02	5.266E+02	5.286E+02	5.268E+02
RGMA	Mean	9.529E+04	8.527E+04	7.158E+04	5.718E+04	6.765E+04	6.821E+04
	Std	7.052E+04	6.917E+04	4.428E+04	3.827E+04	4.076E+04	5.080E+04
	Best	5.451E+02	5.467E+02	**5.292E+02**	**5.227E+02**	**5.231E+02**	5.254E+02
MA	Mean	8.911E+03	8.415E+03	**5.868E+03**	7.741E+03	6.344E+03	7.380E+03
	Std	3.940E+03	4.632E+03	4.151E+03	3.782E+03	4.495E+03	3.526E+03

Table 2. Optimization models on D=4000

		F1	F2	F3	F4	F5	F6
HDIMA	Best	1.977E+03	1.979E+03	3.978E+03	**2.377E+03**	2.182E+03	2.192E+03
	Mean	3.440E+07	3.962E+07	2.363E+06	3.050E+07	2.674E+07	2.697E+07
	Std	1.906E+07	1.583E+07	1.301E+06	1.328E+07	7.648E+06	9.812E+06
DIMA	Best	**1.977E+03**	**1.979E+03**	**3.978E+03**	2.377E+03	**2.182E+03**	**2.182E+03**
	Mean	**1.978E+03**	1.979E+03	**3.978E+03**	**2.378E+03**	**2.182E+03**	**2.182E+03**
	Std	3.583E-01	**4.949E-01**	2.070E-01	7.868E-01	2.481E-01	4.543E-01
RGMA	Best	1.978E+03	1.979E+03	3.978E+03	2.378E+03	2.182E+03	2.182E+03
	Mean	8.069E+06	4.726E+06	4.740E+03	6.517E+06	3.278E+06	1.358E+05
	Std	2.083E+07	1.410E+07	3.317E+03	1.613E+07	1.082E+07	3.623E+05
MA	Best	1.979E+03	2.781E+07	3.981E+03	2.378E+03	2.186E+03	9.173E+03
	Mean	4.195E+07	4.580E+07	3.490E+06	3.617E+07	2.705E+07	3.447E+07
	Std	2.645E+07	1.419E+07	2.563E+06	1.625E+07	1.476E+07	1.387E+07
		F7	**F8**	**F9**	**F10**	**F11**	**F12**
HDIMA	Best	2.182E+03	2.183E+03	2.139E+03	**2.125E+03**	**2.126E+03**	**2.127E+03**
	Mean	2.409E+07	2.392E+07	2.240E+07	2.508E+07	2.461E+07	2.309E+07
	Std	8.394E+06	1.188E+07	9.299E+06	7.693E+06	1.009E+07	8.251E+06
DIMA	Best	**2.182E+03**	**2.183E+03**	2.138E+03	2.126E+03	2.126E+03	2.128E+03
	Mean	**2.182E+03**	**2.183E+03**	**2.160E+03**	**2.127E+03**	**2.128E+03**	**2.129E+03**
	Std	8.649E-02	5.491E-02	9.571E+01	1.291E+00	2.260E+00	1.228E+00
RGMA	Best	2.182E+03	2.184E+03	2.139E+03	2.126E+03	2.127E+03	2.128E+03
	Mean	1.049E+07	8.562E+06	2.710E+06	3.074E+06	8.115E+06	2.389E+06
	Std	2.073E+07	1.957E+07	8.774E+06	8.928E+06	1.532E+07	7.896E+06
MA	Best	2.183E+03	2.192E+03	**2.138E+03**	2.151E+03	2.162E+03	1.316E+07
	Mean	3.199E+07	3.011E+07	2.828E+07	2.381E+07	2.809E+07	3.003E+07
	Std	1.119E+07	1.083E+07	1.119E+07	1.026E+07	1.280E+07	1.080E+07

For the second experiment of D=4000, HDIMA was better than DIMA and RGMA at 4 problems and 11 problems in terms of the mean value, and was not competitive to them at any problem in terms of the standard deviation value. HDIMA performed better than MA in 11 problems and 10 problems in terms of the mean value and the standard deviation value consecutively. The hybridization of the decomposition and the aggregation approaches in HDIMA decrease the percentage of evaluations (FE_{DI}) to 0.1072% instead of 0.1604% at DIMA and 60% at VIL [8]. This is calculated from $max\_FE \approx FE_{init} + FE_{DI} + FE_{opt} + \mathrm{FE}_{\mathrm{MA}}$, where $FE_{init} = NP + 2m$, $FE_{DI} = (2m.Iter_{DI})$, $FE_{opt} = m.Iter_{DI}(NP + FE_{LS})$, $FE_{MA} = Iter_{MA}.(NP + FE_{LS})$, FE_{LS} = s_size$(2 + L)$,$max\ \_FE$ =3E+6, s_size={100,D} for FE_{LS} in FE_{opt} and FE_{MA} respectively, the LS iteration L=10 and $m = \mathrm{D}/100$.

Wilcoxcon nonparametric test has been conducted at the significance level of 5%. The "+" means that the first algorithm "a" is significantly better than the second algorithm "b"; "-" means algorithm "a" is significantly worse than "b"; and "≈" means that there is no significant difference. The analysis of Table 3 shows that HDIMA is significantly better than DIMA and RGMA over dimension of 1000, but worse for the 4000 dimension. There was no significant difference between HDIMA and MA at dimension of 1000, but HDIMA was significantly better for D=4000. DIMA and RGMA are significantly worse than MA at D=1000. This reveals the ability of HDIMA to benefit from the decomposition approach while the other two compared models failed to compete with the MA model over D=1000.

Table 3. Wilcoxcon Nonparametric test

	DIMA	RGMA	MA	DIMA	RGMA	MA
b						
a		1D			4D	
HDIMA	+	+	≈	-	-	+
DIMA		+	-		+	+
RGMA			-			+

The convergence of the 12 problems has been observed for the four models over the 2 different dimensions (1000 and 4000). A graph for one problem from each category of the test suite problems is included for dimensions 1000 and 4000 (Figures 2 to 5). For F_1 over D=1000, HDIMA converges quickly at the beginning and is the best at the end, while RGMA used all of the allowed FE without reaching that level of solution. MA performed well over D=4000 at the beginning only, and HDIMA was the second best after DIMA. Although F_4 is a very complex problem (overlapping spliced nonseparable) and MA is expected to work well, HDIMA was the best over the dimension of 1000 after 1E+6 FEs until the end, and was able to compete with MA starting from 10E+4 FEs over the dimension of 4000. This emphasises the impact of DI on enhancing the performance and achieving a reasonable solution at less than a third of the FEs that the other two models require. F_8 is the relaxed version of F_4 as it is an overlapping partially nonseparable problem. The convergence of F_8 on the 2 dimensions shows that MA suffers from a slow convergence and HDIMA was better and continued to be better. The overall performance of all the models that follow the decomposition approach is better than MA. Moreover DIMA and HDIMA was the best among the all. HDIMA achieved the best convergence for F_{12}, followed by DIMA, then RGMA, while MA was away by 1E+7. The convergence of RGMA is slow at the beginning for most of the problem over the 2 dimensions and the convergence of MA is weaker over the larger dimensions. This shows the need for the decomposition approach with better identification than that given by random approach.

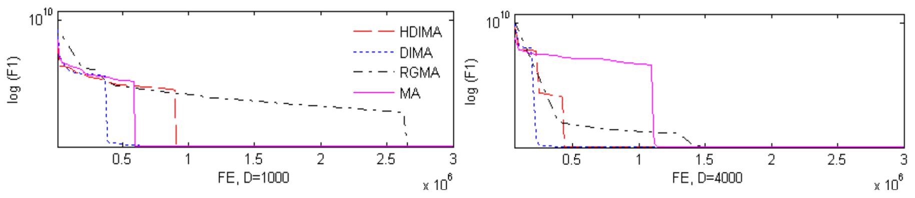


Fig. 2. F1 convergence for 1000 D and 4000 D

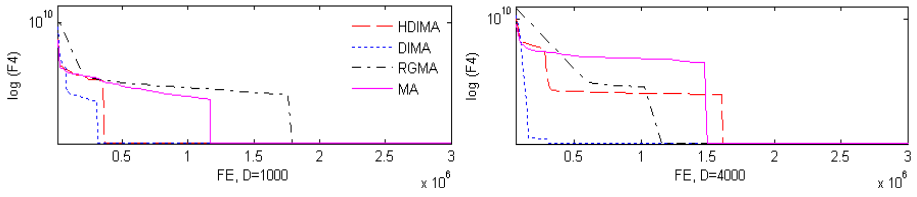


Fig. 3. F4 convergence for 1000 D and 4000 D

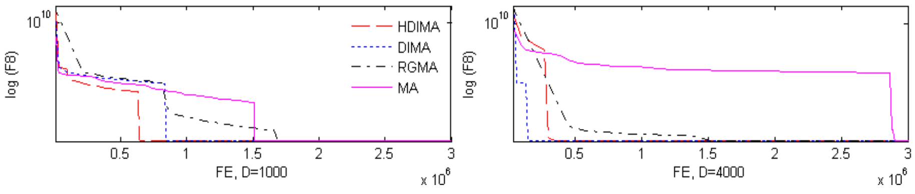


Fig. 4. F8 convergence for 1000 D and 4000 D

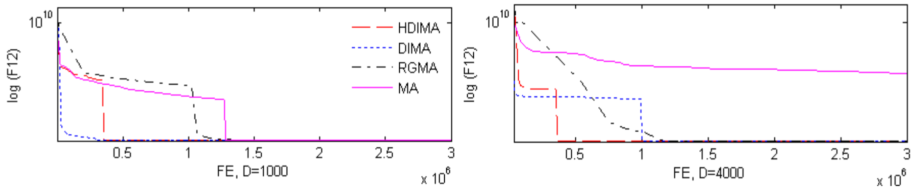


Fig. 5. F12 convergence for 1000 D and 4000 D

6 Conclusion and Future Work

Hybridizing the aggregation approach with the decomposition approach in HDIMA made it competitive to two of the latest developed models DIMA, RGMA on dimension of 1000 and made it even competitive to MA for the 4000 dimension. HDIMA achieved better performance than DIMA and, RGMA over dimension 1000 for almost all the *overlapping spliced* partially nonseparable problems. Another advantage of HDIMA is that of reaching a reasonable solution three times earlier than the other algorithms in the literature when solving large scale problems of complex structure. The recognizable contribution of HDIMA is reducing the computational resources to 0.1072% and 0.1092% for 1000 and 4000 dimensions. This achievement makes HDIMA a recommended optimization model when the computational resources are limited. This calls for future experiments and analysis to investigate the hybridization of DI with other EAs on larger dimensions to merge the advantages of DI and the refining abilities of other EAs. Applying HDIMA in a parallel computing environment is another interesting experiment in which HDIMA could achieve further improvement.

References

1. Omidvar, M., Li, X., Yang, Z., Yao, X.: Cooperative co-evolution for large scale optimization through more frequent random grouping (2010)
2. Potter, M., Jong, K.D.: A Cooperative Coevolutionary Approach to Function Optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN III. LNCS, vol. 866. Springer, Heidelberg (1994)
3. Goldberg, D., Voessner, S.: Optimizing global-local search hybrids. In: Proceedings of the Genetic and Evolutionary Computation Conference, San Mateo, California, pp. 220–228 (1999)

4. Potter, M., Jone, K.D.: Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. Evolutionary Computation 8, 1–29 (2000)
5. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Information Sciences 178, 2986–2999 (2008)
6. Ray, T., Yao, X.: A Cooperative Coevolutionary Algorithm with Correlation Based Adaptive Variable Partitioning. In: IEEE Congress on Evolutionary Computation, pp. 983–989 (2009)
7. Omidvar, M., Li, X., Yao, X.: Cooperative Co-evolution with Delta Grouping for Large Scale Non-separable Function Optimization. In: 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain, pp. 18–23 (2010)
8. Chen, W., Weise, T., Yang, Z., Tang, K.: Large-Scale Global Optimization Using Cooperative Coevolution with Variable Interaction Learning. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 300–309. Springer, Heidelberg (2010)
9. Sayed, E., Essam, D., Sarker, R.: Dependency Identification Technique for Large Scale Optimization Problems. In: Proceedings of the 2012 IEEE Congress on Evolutionary Computation, Brisbane, Australia, pp. 1442–1449 (2012)
10. Mosk-Aoyama, D., Shah, D.: Fast Distributed Algorithms for Computing Separable Functions. IEEE Transactions on Information Theory 54, 2997–3007 (2008)
11. Merz, P.: Memetic Algorithms for Combinational Optimization Problems: Fitness Landscapes and Effective Search Strategies. PhD, Gesamthochschule Siegen, University of Siegen, Germany (2000)
12. Molina, D., Lozano, M., García-Martínez, C., Herrera, F.: Memetic Algorithms for Continuous Optimisation Based on Local Search Chains. Evolutionary Computation 18, 27–63 (2010)
13. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc. (1989)
14. Deb, K., Agrawal, R.: Simulated Binary Crossover for Continuous Search Space. Complex Systems 9, 115–148 (1995)
15. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York (1992)
16. Molina, D., Lozano, M., Sánchez, A., Herrera, F.: Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SW-Chains. In: Soft Computing - A Fusion of Foundations, Methodologies and Applications, pp. 1–20 (2010)
17. Tang, K., Yao, X., Suganthan, P.N., MacNish, C., Chen, Y.-P., Chen, C.-M., Yang, A.Z.: Benchmark Functions for the CEC 2008 Special Session and Competition on Large Scale Global Optimization. University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory, NICAL, Héféi, Ānhū, China (2007)
18. Talbi, E.: Metaheuristics: from design to implementation, pp. 124–127. John Wiley & Sons, Hoboken (2009)

Application of Cooperative Convolution Optimization for ^{13}C Metabolic Flux Analysis: Simulation of Isotopic Labeling Patterns Based on Tandem Mass Spectrometry Measurements

Rohitash Chandra[1,3], Mengjie Zhang[1], and Lifeng Peng[2]

[1] School of Engineering and Computer Science
[2] School of Biological Sciences
Victoria University of Wellington, P.O. Box 600, Wellington, 6140, New Zealand
[3] Department of Computer Science and Information Technology,
University of Fiji, Saweni Campus, Lautoka, Fiji
c.rohitash@gmail.com, mengjie.zhang@ecs.vuw.ac.nz,
lifeng.peng@vuw.ac.nz

Abstract. Metabolic fluxes are the key determinants of cellular metabolism. They are estimated from stable isotope labeling experiments and provide the informative parameters in evaluating cell physiology and causes of disease. Metabolic flux analysis involves in solving a system of non-linear isotopomer balance equations by simulating the isotopic labeling distributions of metabolites measured by tandem mass spectrometry, which is essentially an optimization problem. In this work, we introduce the cooperative coevolution optimization method for solving the set of non-linear equations that decomposes a large problem into a set of subcomponents. We demonstrate that cooperative coevolution can be used for solving the given metabolic flux model. While the proposed approach makes good progress on the use of evolutionary computation techniques for this problem, there exist a number of disadvantages that need to be addressed in the future to meet the expectation of the biologists.

Keywords: Metabolic flux analysis, metabolic pathways, cooperative coevolution, evolutionary algorithms.

1 Introduction

^{13}C Metabolic flux analysis (MFA) deals with a mathematical model that describes the relationship between the relative abundances of metabolites and isotopomers of metabolites with metabolic fluxes for a given metabolic network, expressed as a system of metabolite and isotopomer mass balance equations [1,2,3]. This system is usually overdetermined for solving metabolic fluxes, and in essence, involves a large-scale non-linear parameter fitting to minimize the difference between experimentally measured and computationally simulated isotopic labelling patterns.

Metabolic fluxes, i.e., the reaction rates in the metabolic pathways in a biological system, are key determinants of cell physiology and informative parameters in evaluating cellular mechanisms and causes of diseases and identifying possible genetic targets for optimization of a particular phenotype [1,2]. Metabolic flux analysis involves in (1) introducing isotope tracers (e.g., ^{13}C) into a cell culture and allowing the system to reach at metabolic steady state, (2) measuring the relative abundances of the isotopic isomers, i.e. distinct labelling patterns (isotopomers) of intracellular metabolites in the metabolic pathways, and (3) computationally simulating these measurements to estimate metabolic fluxes [3]. Recent advance in mass spectrometry (MS) technology, tandem MS (MS/MS), provides the opportunity for MFA towards resolving detailed fluxes for complex metabolic network. Jeffrey et al.[4] demonstrated that tandem MS was able to yield detailed positional labelling information of isotopomers of metabolites through measuring the mass distributions of their daughter fragments with superior sensitivity and the MS/MS data can be computed allowing estimation of fluxes with better precision [5].

In the past, genetic algorithms have been used for solving metabolic pathways in conjunction with some numerical methods such as the gradient based Newton's method [6,7]. Genetic algorithms provide a means to optimize the model without the need of information on the derivatives of the equations. Zhang and Yao [8] used a hybrid genetic algorithm where sequential simplex technique was used as a local search method to simulate the flux distribution of central metabolism of *Saccharomyces cerevisiae*. However, these methods are limited to resolving large and complex biological networks due to heavy computational burden.

Cooperative coevolution (CC) is an evolutionary computation optimisation method that decomposes a problem into subcomponents and solves them independently in order to collectively solve the problem. The subcomponents are implemented as subpopulations that are evolved separately and the cooperation only takes place at fitness evaluation for the respective individuals in each subpopulation. CC has shown promising results in general function optimisation problems [9,10]. We have used CC for training feed forward and recurrent neural networks [11,12,13,14]. More relevantly, CC has been used for flux balance analysis (FBA), which is different from ^{13}C MFA in the present work as FBA employs no ^{13}C label and thus contains no isotopomer balance equations in the mathematical model, for maximum ATP production for *Bacillus subtilis* in the past [15]. The advantage of CC over standard genetic algorithms is that they provide a mechanism for evolving parts of the problem in isolation, i.e., a large problem can be effectively broken down into smaller problems using cooperative coevolution, and different forms of constraint handling and fitness approximation can be assigned to the respective subpopulations. It is, therefore, promising that CC can be used to resolve large and complex biological networks to provide detailed insights into biological systems.

This work applies cooperative coevolution for ^{13}C MFA for simulating the isotopic labeling patterns of the metabolites in an example metabolic network selected from literature [5]. The problem is expressed as a system of non-linear equations, which is an over-determined system and solved by optimization using the algorithm developed in this paper. The goal is to achieve the least errors of the isotopomer balance equations and simulate the isotopic labelling patterns of the metabolites.

The rest of the paper is organised as follows. Section 2 presents the background of ^{13}C MFA. Section 3 gives details of the proposed approach where CC is used for MFA. Section 4 presents the results and section 5 concludes the work with a discussion on future work.

2 Mathematical Models for ^{13}C MFA

2.1 Metabolic Model

We used an example metabolic model as described by Choi and Antoniewicz [5] to simulate the isotopic labeling patterns of the metabolites that are expressed in the format of MS/MS measurement data through linear grouping isotopomers of the same number of labels in a metabolite pool. The model consists of gluconeogenesis, TCA cycle and glyoxylate shunt as shown in Figure 1. The biochemical reactions and the atom transitions for these reactions involved in the model are taken from [5]. In this network, [^{13}C]ASP and AcCoA are the two substrates and Gluc and CO_2.ext are the two products; the seven intermediary intracellular metabolites (Cit, AKG, Suc, Fum, OAC, Glyox and CO_2) were assumed to be at the metabolic and isotopic steady state. The labelled substrate [^{13}C]ASP contained 25% [1,2-^{13}C] ASP, 25% [4-^{13}C] ASP, 25% [2,3,4-^{13}C] ASP and 25% unlabelled ASP. The assumed fluxes for the network are shown in Figure 1.

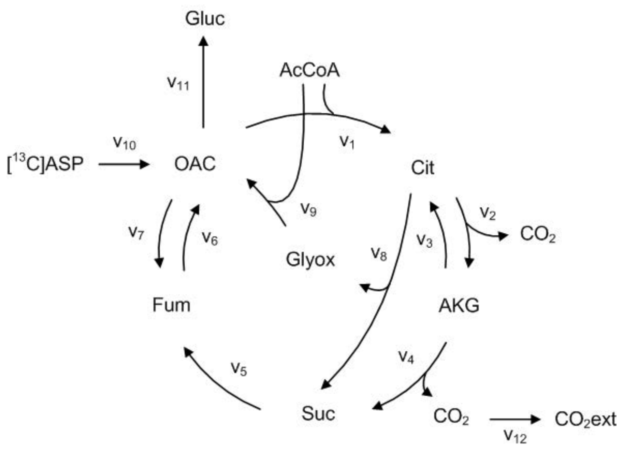


Fig. 1. Metabolic network model [5]. Abbreviations of metabolites: Asp, aspartate; OAC, oxaloacetate; AcCoA, acetyl coenzyme A; Cit, citrate; AKG, alpha-ketoglutarate; Fum, fumarate; Suc, succinate; Glyox, glyoxylate. Assumed fluxes (arbitrary units): $v_1 = 100, v_2 = 220, v_3 = 150, v_4 = 70, v_5 = 100, v_6 = 140, v_7 = 40, v_8 = 30, v_9 = 30, v_{10} = 60, v_{11} = 90, v_{12} = 140$.

2.2 Mathematical Description of the Model

By assuming the isotopic steady state, a set of isotope isomer (isotopomer) balance equations for the intracellular metabolites can also be formulated. The isotopomer balance equations were constructed based on the method by Schmidt *et al* [16]. First, the atom mapping matrices (AMMs) were created to describe the carbon transitions in the reactions according to the protocol by Zupke and Stephanopoulos [17], followed by the construction of isotopomer mapping matrices (IMMs) derived from AMMs using the algorithm by Schmidt *et al* [16]. The IMMs describe the transformation of isotopomers of the reactant into the product. An IMM is constructed for each pair of reactant and product molecules in the reaction. For example, IMM$_{OAC>Cit}$ is the transformation matrix that describes how the isotopomers of reactant OAC are transformed into the isotopomers of the product Cit, which are matrices filled with the value of "1" at the appropriate positions where the transitions of isotopomer from the reactant to the product occur and all the remaining elements are tilled with "0". We have created the AMM and IMM databases for the given metabolic model. Isotopomer distribution vectors (IDVs) collect the fractional abundances of all isotopomers of the metabolites. For a metabolite with n carbons, the IDV has 2^n elements.

By using IMMs and IDVs, the isotopomer balance equations can be formulated in which the sum of incoming isotopomer fluxes is equal to the sum of isotopomer fluxes out of the metabolite pool. The following equations are used to express the network model shown in Figure 1.

$$\begin{aligned} OAC: \quad & v_6\left(\frac{1}{2}IMM_{Fum(abcd)>OAC} + \frac{1}{2}IMM_{Fum(dcba)>OAC}\right).IDV_{Fum} \\ & + v_9(IMM_{AcCoA>OAC}.IDV_{AcCoA}) \otimes (IMM_{Glyox>OAC}.IDV_{Glyox}) \\ & + (IMM_{ASP>OAC}.IDV_{ASP}) - (v_1+v_7+v_{11})IDV_{OAC} = 0 \end{aligned} \tag{1}$$

$$\begin{aligned} Cit: \quad & v_1(IMM_{AcCoA>Cit}.IDV_{AcCoA}) \otimes (IMM_{OAC>Cit}.IDV_{OAC}) \\ & + v_3(IMM_{AKG>Cit}.IDV_{AKG}) \otimes (IMM_{CO_2>Cit}.IDV_{CO_2}) \\ & - (v_2+v_8)IDV_{Cit} = 0 \end{aligned} \tag{2}$$

$$AKG: \; v_2(IMM_{Cit>AKG}.IDV_{Cit}) - (v_3+v_4)IDV_{AKG} = 0 \tag{3}$$

$$\begin{aligned} Suc: \quad & v_4\left(\frac{1}{2}IMM_{AKG>Suc(bcde)} + \frac{1}{2}IMM_{AKG>Suc(edbc)}\right).IDV_{AKG} \\ & + v_8\left(\frac{1}{2}IMM_{Cit>Suc(cdef)} + \frac{1}{2}IMM_{Cit>Suc(fedc)}\right).IDV_{Cit} \\ & - (v_5)IDV_{Suc} = 0 \end{aligned} \tag{4}$$

$$\begin{aligned} Fum: \quad & v_5\left(\frac{1}{4}IMM_{Suc(abcd)>Suc(abcd)} + \frac{1}{4}IMM_{Suc(abcd)>Suc(dcba)}\right).IDV_{Suc} \\ & + v_5\left(\frac{1}{4}IMM_{Suc(dcba)>Suc(abcd)} + \frac{1}{4}IMM_{Suc(dcba)>Suc(dcba)}\right).IDV_{Suc} \\ & + v_7\left(\frac{1}{2}IMM_{OAC>Fum(abcd)} + \frac{1}{2}IMM_{OAC>Suc(dcba)}\right).IDV_{OAC} \\ & - V_6.IDV_{Fum} = 0 \end{aligned} \tag{5}$$

$$Glyox: \; v_8(IMM_{Cit>Glyox}.IDV_{Cit}) - v_9.IDV_{Glyox} = 0 \tag{6}$$

$$\begin{aligned} CO_2: \quad & v_2(IMM_{Cit>CO_2}).IDV_{Cit} \\ & + v_4\left(\frac{1}{2}IMM_{AKG>Suc(bcde)} + \frac{1}{2}IMM_{AKG>Suc(edcb)}\right).IDV_{AKG} \\ & - (v_3+v_{12})IDV_{CO_2} = 0 \end{aligned} \tag{7}$$

where, $\otimes$ is the column-wise multiplication of the vectors. Equations 1- 7 describe the relationship between the metabolic fluxes and isotopomer distributions of the metabolites in the network that can be solved by simulation computationally.

2.3 Correlation between Isotopomer Distributions (IDVs) and Tandem MS Measurements

Tandem mass spectrometer measures the mass distribution of isotopomers of the metabolites (called mass distribution vector, MDV) and the mass distribution of the fragments of particular precursor isotopomers (called fragment mass distribution vector FMDV). The MDV and FMDV can be calculated from the IDV based on the theory established in [5]. Taking the four-carbon metabolite OAC in the metabolic model Figure 1 as an example, the correlation between the IDV and FMDV is expressed in Figure 2. We have created the software that automatically computes the FMDVs for all possible fragments of a molecule with any given number of carbons, which is available online [18].

$$\begin{array}{c}\text{FMDV}\\(\text{OAC}_{1234}>\text{OAC}_{34})\end{array} = \begin{bmatrix} M0>m0\\ M1>m0\\ M2>m0\\ M1>m1\\ M2>m1\\ M3>m1\\ M2>m2\\ M3>m2\\ M4>m2 \end{bmatrix} = \begin{bmatrix} OAC(0000)\\ OAC(0100)+(1000)\\ OAC(1100)\\ OAC(0001)+(0010)\\ OAC(0101)+(0110)+(1001)+(1010)\\ OAC(1101)+(1110)\\ OAC(0011)\\ OAC(0111)+(1011)\\ OAC(1111) \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\ 0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&1&0&0&1&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\ 0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix} \cdot \text{IDV}_{\text{OAC}}$$

Fig. 2. Correlation between the IDV and FMDV

3 Cooperative Coevolution for Solving the Metabolic Flux Model

3.1 Conversion to an Optimisation Problem

The metabolic flux model is expressed as a system of non-linear equations given in Section 2.2. Given fixed values for the IMMs, the set of IDVs and fluxes need to be found in order to solve the equations. To use cooperative coevolution for optimisation, the problem of solving a system of non-linear equations has to be converted into an optimisation problem. This is done by defining a fitness function that will constitute of errors that need to be minimised in order to solve the equations. This problem also has constraints as all the elements of the IDVs should be within the range of [0,1] that should sum up to 1.

The fitness of a solution is calculated using Equation 8.

$$GF = \sqrt{\sum_{i=0}^{k} f_i} \tag{8}$$

where

$$f_i = \frac{\sum_{j=0}^{n} X_{ij}^2}{n} \tag{9}$$

where, X_i is the vector that is obtained after simplifying the left hand side of Equations 1 - 7. f_i represents each of the equations from Equations 1 - 7, n is the size of X_i and k is the number of equations.

3.2 Optimisation Using Cooperative Coevolution

The entire framework consists of the set of non-linear equations that rely on the IMMs generated from the AMMs for each of the metabolites. Figure 3 shows the flowchart of the framework that employs cooperative coevolution for the simulation of the IDVs and FMDV.

```
1) Decompose the problem into $k$ subcomponents. All IDVs have different sub-populations.
2) Initialise and cooperatively evaluate each sub-population (SP)
while until termination do
  for each SP do
    for $n$ Generations do
      Cooperatively evaluate selected individuals in SP
      Evolve using genetic operators
    end for
  end for
end while
```

Algorithm 1. Cooperative Coevolution for Metabolic Flux Analysis

The general framework for metabolic flux analysis using cooperative coevolution is given in Algorithm 1. The algorithm begins by initialising and cooperatively evaluating each of individuals of the respective sub-populations. After the initialisation and evaluation phase, the evolution proceeds. All the sub-populations (SP) are evolved in a round-robin fashion for the depth of n generations. A cycle is complete when all the sub-populations have been evolved. The algorithm terminates until the maximum number of function evaluations are reached.

Figure 3 shows how the original problem is decomposed into sub-populations for the specific problem described in Section 2.2.

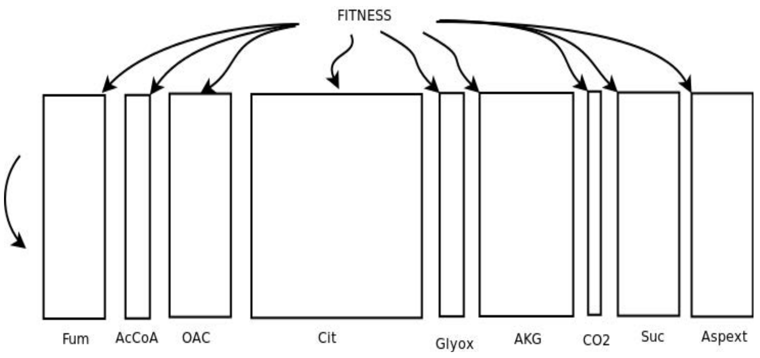


Fig. 3. Problem decomposition in cooperative coevolution for metabolic flux analysis. Note that the values for the IDV of OAC will be transformed into FMDV for comparison when given with different sets of fluxes.

3.3 Sub-populations of Cooperative Coevolution

The G3-PCX (generalised generation gap with parent centric crossover operator) algorithm is used in the sub-populations of cooperative coevolution [19].

The details of the G3-PCX are given as follows. The generalised generation gap differs from a standard genetic algorithm in terms of selection and the creation of new individuals. In G3-PCX, the whole population is randomly initialised and evaluated similarly to the standard genetic algorithm. The difference lies in the optimisation phase where a small sub-population is chosen. At each generation, n best fit and m random individuals are chosen from the main population to make up a sub-population. The sub-population is evaluated at each generation and the evaluated individuals are added to the main population. In this way, over time, the individuals of the main populations are evaluated.

The best individual in the population is retained at each generation. The parent-centric crossover operator is used in creating an offspring based on orthogonal distance between the parents [19]. The parents are made of female and male components. The offspring is created in the neighbourhood of the female parent. The male parent defines the range of the neighbourhood. The neighbourhood is the distance of the search space from the female parent which is used to create the offspring. The genes of the offspring extract values from intervals associated in the neighbourhood of the female and the male using a probability distribution. The range of this probability distribution depends on the distances among the genes of the male and the female parent. The parent-centric crossover operator assigns a larger probability to create an offspring near the female than anywhere else in the search space.

4 Simulation and Analysis

In this section, the cooperative coevolution optimization algorithm is used to solve the system of non-linear equations given in Section 2.2. The G3-PCX algorithm [19] is employed in the sub-populations of cooperative coevolution. The G3-PCX algorithm employs a mating pool size of 2 offspring and a family size of 2 parents for all cases. This set-up has been taken from [19] and also used in previous work [11]. The sub-populations are seeded in the range of [0, 1] for the IDVs in all the experiments. Each of the 9 IDVs contain the following number of variables; Fum = 16, AcCoA = 4, OAC = 16, Cit= 64, Glyox = 4, AKG = 32, CO˙2 = 2, Suc= 16, and ASP = 16. Note that the IDV for ASP will not be optimized as it is fixed. The sub-populations for the IDVs are constrained to have solutions between the range of [0,1]. This is done by monitoring the new solutions in the respective sub-populations. If the values in the new solution are less than 0, a small real random number is added to 0. If the values in the new solution greater than 1, than a small real random number is subtracted from 1. We will apply cooperative coevolution to find all the IDVs, and the simulated TMDV for OAC given a set of predefined fluxes. The tandem MS is simulated in this stage as there are no real measurements available for the given network model. The training is terminated when the maximum number of function evaluations is reached.

4.1 Simulation of IDVs from Given Predefined Flux Values

In this section, the flux is given and the cooperative coevolution algorithm finds all the IDVs and then calculate the FMDV for OAC. A total of 8 sub-populations were used in order to represent each IDV. Each run is initialised with different real random numbers for the search to begin. After the end of the optimisation process, the values for the IDV of OAC will be transformed into FMDV for comparison when given with different sets of fluxes. This reflects on the sensitivity of the model and optimisation algorithm.

Table 1 shows the results for 30 independent experimental runs showing the mean and 95% confidence interval. The results evaluate the optimal population size for the problem for the duration of 1000 000 fitness function evaluations. The population size of 1500 gives the best performance in terms of the least error that shows that the equation system has been solved.

Table 1. Results show the global fitness given different population size

Problem	Population Size	Global Fitness
$FMDV(OAC_{1234} > OAC_{34})$	500	6.253e-03±2.856e-04
	1000	5.85e-03 ± 2.50e-04
	1500	5.56e-03 ± 2.36e-04

Table 2 shows the results of the simulated FMDV for OAC for two sets of fluxes (two cases) which test the sensitivity of the cooperative coevolution optimisation method. Case 1 employs the set of fluxes shown in the caption of Figure 1 and Case 2 employs the same set except for two different flux values, i.e. $v_6 = 200$ and $v_7 = 100$. The difference in these two cases and the fitness is given. Note that the results have been rounded off to four decimal places. The differences between the two cases show that the cooperative coevolution optimisation method has been able to balance the metabolic flux equations to a small error and the model has been sensitive to respond to different sets of fluxes.

Table 2. Simulated Results

		Case 1	Case 2	Difference
FMDV	M0>m0	11.3397	12.0894	-0.7497
$OAC_{1234} > OAC_{34}$	M1>m0	16.6009	15.1822	1.4187
	M1>m1	04.7857	05.7403	0.9547
	M2>m0	10.0141	10.7398	0.7257
	M2>m1	17.6064	17.4824	0.0124
	M2>m2	15.7889	13.5307	2.2582
	M3>m1	11.3982	09.6425	1.7556
	M3>m2	09.4825	10.5090	1.0264
	M4>m2	02.9836	05.0835	2.0999
Fitness		5.8456e-3	6.8728e-2	

4.2 Further Discussions

The results have shown that cooperative coevolution can be used for metabolic flux analysis. However, there are a number of aspects that need to be noticed. Firstly, due to the stochastic nature of evolutionary computation, the proposed method produces multiple solutions for the same problem at different experiment runs. In many situations, this is good as many problems (particularly the multi-modal problems) could have multiple (sub)optima and all of them can meet the requirements of the problem. However, some problems might only have one optimal point and it is important to obtain that point. For metabolic flux analysis problems on this point, there are different views — while some think this problem has a single optimal solution, some others think multiple (sub)optimal solutions exist. The current literature does not have solid evidence for or against either of them. This needs to be investigated in the future.

Secondly, it is no doubt that many biological scientists would prefer a single solution to be achieved in different simulation runs, at least similar results can be obtained from different runs. For example, the results obtained from this paper are different from those from [5], and from the biological point of view it would be desire to have the same results as (or close to) Choi's results from the simulations. We believe that it is possible to achieve this, but this needs to be further investigated in the future.

Thirdly, this work so far has not provided the evidence to support the hypothesis that MS/MS is more sensitive in terms of the changes of fluxes than MS, which is the on-going work.

Finally, the approach of simulating the set(s) of unknown fluxes and IDVs simultaneously has not been extensively explored using evolutionary computation methods. In the literature, the IDVs have mostly been optimised using numerical methods such as the Gauss-Siedel method. In future, we will use the cooperative coevolution method for finding unknown fluxes.

5 Conclusions and Future Work

This work applied cooperative coevolution based optimization to metabolic flux analysis. The sub-populations in cooperative coevolution were assigned to all the different IDVs that were simultaneously evolved. The results suggest that cooperative coevolution can be used solve the system of non-linear equations and that the approach is sensitive when given different values for the set of flux due to the stochastic nature of evolutionary computation.

While this approach was successful as preliminary work and it opened the door of applying evolutionary optimisation methods to this domain, it does have a number of disadvantages compared with the traditional methods. For example, this method produces multiple (sub) solutions for the same problem, while many biological scientists would prefer a single solution. In addition, this work so far does not provide evidence to show such problems are single-modal or multi-modal optimisation ones. We have not been able to provide definite interpretations on why this approach provides different solutions from the literature [5]. These issues will need to be investigated in the future.

In addition, the method will also be used to approximate the measured data for the FMDV, constraints will be assigned to the respective sub-populations, and the set(s) of fluxes also need to be found. We will also compare this approach to other evolutionary computation algorithms (such as [15]) for this problem in the future.

References

1. Bailey, J.: Toward a science of metabolic engineering. Science 252(5013), 1668–1675 (1991)
2. Nielsen, J.: It is all about metabolic fluxes. J. Bacteriol. 185(24), 7031–7035 (2003)
3. Sauer, U.: Metabolic networks in motion: 13C-based flux analysis. Mol. Syst. Biol. 2, 62 (2006)
4. Jeffrey, F.M., Roach, J.S., Storey, C.J., Sherry, A.D., Malloy, C.R.: 13C isotopomer analysis of glutamate by tandem mass spectrometry. Anal. Biochem. 300(2), 192–205 (2002)
5. Choi, J., Antoniewicz, M.R.: Tandem mass spectrometry: A novel approach for metabolic flux analysis. Metab. Eng. 13(2), 225–233 (2011)
6. Peng, L., Arauzo-Bravo, M.J., Shimizu, K.: Metabolic flux analysis for a ppc mutant Escherichia coli based on 13C-labelling experiments together with enzyme activity assays and intracellular metabolite measurements. FEMS Microbiol. Lett. 235(1), 17–23 (2004)
7. Zhao, J., Shimizu, K.: Metabolic flux analysis of escherichia coli k12 grown on 13c-labeled acetate and glucose using gc-ms and powerful flux calculation method. Journal of Biotechnology 101(2), 101–117 (2003)
8. Zhang, H., Yao, S.: Simulation of flux distribution in central metabolism of saccharomyces cerevisiae by hybridized genetic algorithm. Chinese Journal of Chemical Engineering 15(2), 150–156 (2007)
9. Potter, M.A., Jong, K.A.D.: A Cooperative Coevolutionary Approach to Function Optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN III. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994)
10. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Inf. Sci. 178(15), 2985–2999 (2008)
11. Chandra, R., Frean, M., Zhang, M.: An Encoding Scheme for Cooperative Coevolutionary Feedforward Neural Networks. In: Li, J. (ed.) AI 2010. LNCS, vol. 6464, pp. 253–262. Springer, Heidelberg (2010)
12. Chandra, R., Frean, M., Zhang, M.: On the issue of separability for problem decomposition in cooperative neuro-evolution. Neurocomputing 87, 33–40 (2012)
13. Chandra, R., Frean, M., Zhang, M., Omlin, C.W.: Encoding subcomponents in cooperative co-evolutionary recurrent neural networks. Neurocomputing 74(17), 3223–3234 (2011)
14. Chandra, R., Zhang, M.: Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction. Neurocomputing 86(0), 116–123 (2012)
15. Supudomchok, S., Chaiyaratana, N., Phalakomkule, C.: Co-operative co-evolutionary approach for flux balance in bacillus subtilis. In: IEEE Congress on Evolutionary Computation, pp. 1226–1231. IEEE (2008)
16. Schmidt, K., Carlsen, M., Nielsen, J., Villadsen, J.: Modeling isotopomer distributions in biochemical networks using isotopomer mapping matrices. Biotechnology and Bioengineering 55(6), 831–840 (1997)
17. Zupke, C., Stephanopoulos, G.: Modeling of isotope distributions and intracellular fluxes in metabolic networks using atom mapping matrices. Biotechnology Progress 10(5), 489–498 (1994)
18. Online Tandem MS Software (May 2012), http://softwarefoundationfiji.org/research/bioinfor/tandem/
19. Deb, K., Anand, A., Joshi, D.: A computationally efficient evolutionary algorithm for real-parameter optimization. Evol. Comput. 10(4), 371–395 (2002)

An Efficient Two-Phase Ant Colony Optimization Algorithm for the Closest String Problem

Hoang Xuan Huan[1], Dong Do Duc[1], and Nguyen Manh Ha[2]

[1,2] University of Engineering and Technology, VNU, Hanoi, Vietnam
{huanhx,dongdoduc}@vnu.edu.vn,
manhhacz@gmail.com

Abstract. Given a finite set S of strings of length m, the task of finding a string t that minimizes the Hamming distance from t to S, has wide applications. This paper presents a two-phase Ant Colony Optimization (ACO) algorithm for the problem. The first phase uses the Smooth Max-Min (SMMAS) rule to update pheromone trails. The second phase is a memetic algorithm that uses ACO method to generate a population of solutions in each iteration, and a local search technique on the two best solutions. The efficiency of our algorithm has been evaluated by comparing to the Ant-CSP algorithm.

Keywords: Memetic algorithm, Ant Colony Optimization, Closest String Problem, Local Search, Pheromone update rule.

1 Introduction

Given a finite set S of strings of the same length, the task of Closest String Problem (CSP) is to find a string t so that the Hamming distance $d_H(t, S)$ is minimal. This problem plays an important role in coding theory [12,13] and Genetics [16,17]. Therefore, it has been studied extensively [1,2,5,6,13-19,21], and it has been proved [12,17] to be NP-hard. Besides heuristics methods to quickly find good enough solutions [6,14,15], several approximate algorithms [13,17-19] had been developed for CSP. Recently, Faro and Papalando [11] proposed an ACO algorithm called Ant-CSP. The experiment showed that Ant-CSP is not only much better in quality of the solutions, but also in the runtime comparing to two state-of-the-art algorithms: genetic and simulated annealing algorithm [11]. However, Ant-CSP's construction graph is too simple, it applied neither heuristic information nor local search.

For NP-hard problems, there are two methods that are used widely: local search, and population-based search. Moscato [20] had proposed a framework to combine these two methods called memetic algorithm to significantly improve quality of solutions.

This paper presents a two-phase algorithm named ACOM-CSP which combines two algorithms: ACO and Memetic. The first phase is the ACO phase with the SMMAS pheromone update rule. The second phase is a memetic phase. In each loop of this phase, ACO method is applied to generate a population of solutions, then local search technique is applied on only two best solutions in one loop. If only the algorithm in the first phase is used, the algorithm is called ACO-CSP. The experiment had showed that our new algorithms are better than Ant-CSP algorithm both in terms of quality of solutions and runtime. Although the runtime of ACOM-CSP is slightly higher than ACO-CSP of the same number of loops, ACOM-CSP is more stable. When the two algorithms run in same runtime, ACOM-CSP produces solutions of better quality.

The rest of this paper is organized as follows. Section 2 defines the CSP and presents related works: ACO method, memetic framework. Our new algorithm is introduced in section 3. Section 4 describes the experiment comparing our algorithm to Ant-CSP. Conclusions are presented in the last section.

2 Closest String Problem and Related Works

2.1 Closest String Problem

Considering a finite set of X strings, over an alphabet $\sum$, each string of length m, Hamming distance is defined as follows:

Definition 1. (Hamming Distance)

i) For any two strings $x = x_1 \dots x_m$ and $y = y_1 \dots y_m$, the distance $d_H(x,y)$ is the number of mismatches between x_i and y_i $(i \leq m)$.

ii) For each finite set S of strings of length m, the distance from one string t of length m to set S is the maximal distance from string t to each string in S.

$$d_H(t,S) = max\{d_H(t,s): s \epsilon S\} \tag{1}$$

For example: Consider a string t = “AAAA” and a set S ={ s^1= “AAAB”, s^2 = “AAAA”, s^3 = “AABB”}, it can be seen that: $d_H(t,s^1) = 1$; $d_H(t,s^2) = 0$; $d_H(t,s^3) = 2$; so $d_H(t,S) = 2$.

Definition 2. (The closest string of one set of string) Given a finite set S of n strings, each string of length m, a string t is called the *closest string* of S if:

$$d_H(t,S) = min\{ d_H(x,S): x \epsilon X\} \tag{2}$$

For each given set S, the task of CSP is to find a closest string t of S. In other words, it is to find a string t that optimizes the objective function $d_H(t,S)$. The problem had been proved to be NP-hard [12,17].

Ant Colony Optimization and Ant-CSP Algorithm. ACO method had been proposed by Dorigo [9] to solve the Traveling Salesman problem. Until now, it had been developed into many variations and is widely used for solving hard combinatorial optimization problems (see [10]). In these algorithms, the under-examined problem is transformed into the path finding problem on a construction graph $G = (V,E,\Omega,\eta,T)$, where V is a set of vertices (with a set of start vertices C_0), E is a set of edges, Ω is a set of constraints for determining the next vertex and specifying acceptable solutions, η is a vector that denotes heuristic information on vertices or edges, T is pheromone trail vector on vertices or edges (initial value is τ_0) that represents reinforcement learning information for solution finding. Each acceptable solution is a path with bounded length. It starts from one vertex in C_0, then expands by a random walk procedure based on heuristic information and pheromone trail. These paths must satisfy the constraints Ω. ACO algorithm uses N_{ant} artificial ants.

In each loop, each ant uses the random walk procedure to construct solution. Then, those solutions are evaluated and used for updating the pheromone trails as reinforcement learning information that helps ant colony constructs solutions in the next loops. This procedure is specified in Figure 1.

```
Procedure of ACO algorithms;
Begin
    Initialize;      // initialize pheromone trail matrix and u ants
    Repeat
        Construct solutions;     // each ant constructs its own solution
        Update trail;
    Until End condition;
End;
```

Fig. 1. Specification of an ACO algorithm

There are 3 factors that affect the performance of ACO algorithm: 1) construction graph, 2) heuristic information, 3) pheromone update rule. Among them, pheromone update rule is the universal factor and often used to name the algorithm.

Faro and Papalando [11] proposed an elite-ant system called Ant-CSP algorithm. The simulation results had shown that this algorithm outperformed two state-of-the-art algorithms in both terms of quality of solutions and runtime. However, this algorithm still has some disadvantages. It did not use heuristic information and its pheromone update rule is not the best one (see [10]).

2.2 Memetic Framework

There are two methods that are widely applied in approximating solutions for NP-hard problems: Local search and population-based search. ACO method and genetic algorithms (GA) belong to the latter. To combine these two methods, Moscato [20] had proposed memetic framework. Nowadays, the use of this idea is called memetic computing [3,4]. A simple memetic procedure is specified in Figure 2.

```
Procedure Memetic algorithm
Begin
    Initialization: initialize the first population;
    While stop conditions not satisfied do
        Evaluate individuals in population;
        Evolve the population by given algorithms;
        Choose a subset $\Omega_{il}$ to evolve by local search;
        For each individual in $\Omega_{il}$ do
            Local search;
        End for;
        Choose the best individual;
    End while;
    Choose the solution;
End;
```

Fig. 2. Specification of a simple memetic algorithm

3 ACOM-CSP Algorithm

In this section we first introduce the ACO-CSP procedure which runs in the first phase. When only the first phase is used to solve CSP, it is also named ACO-CSP algorithm.

3.1 ACO-CSP

The framework of ACO-CSP has been delineated in Figure 1. Following are the details of the construction graph, the solution finding procedure and the pheromone update rule:

Construction Graph. Consider a set of n strings $S = \{s^1, \ldots, s^n\}$ of length m, where $s^i = s_1^i \ldots s_m^i$, $s_j^i \in \{\text{alphabet } \sum\}$. Denote the set $\{a \epsilon \sum : \exists s_j^i = a\}$ by V_i. Then the construction graph contains m columns and the i^{th} column is composed of vertices in V_i . E is a set of edges that connect two adjacent columns. Heuristic information η_{ij} for vertex (i, j) at column i, row j is:

η_{ij} = number of string s^k that has s_j^k coincides with the label of the vertex (i, j) (3)

With the given parameter τ_{max} and τ_{min}, pheromone trails on all vertices (i, j) are initialized to $\tau_{max} = \frac{1.0}{|\Sigma|}$.

Random Walk Procedure to Find Solution. One ant constructs its own solution by moving from the first column to the last. Ant chooses a vertex on the first column with the probability equal to the normalized product of heuristic information and pheromone trail, then expand as follows. Supposing ant is at vertex $a \epsilon V_i$, it chooses vertex $b \epsilon V_{i+1}$ as the next position of string with probability:

$$P(b) = \frac{\tau_b \eta_{ib}^{\alpha}}{\sum_{c \epsilon V_{i+1}} \tau_c \eta_{ic}^{\alpha}} \quad , \tag{4}$$

α is a positive constant value, $\alpha = 0$ means heuristic information is not used.

Pheromone Update Rule. When all ants found their own solution, pheromone trail will be updated by SMMAS rule, which had been proposed in [7]. This rule had been also applied effectively in [8] for haplotype inference problem.

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \Delta_{ij}, \tag{5}$$

$$\text{where: } \Delta_{ij} = \begin{cases} \rho\tau_{max} & (i, j) \epsilon \text{ best solution} \\ \rho\tau_{min} & \text{otherwise} \end{cases} \tag{6}$$

Typically, to improve quality of the solutions, local search is used (see[10]) with two strategies: 1) use local search for only the best solution, 2) use local search for all solutions that had been found. If the first strategy is used, the best solution is usually repeated, but if the second strategy is used, runtime increases considerably. In ACOM-CSP, memetic framework implements local search for the two best solutions in the second phase.

3.2 Memetic-CSP Algorithm

The second phase of ACOM-CSP implements memetic algorithm following the specification in Figure 3 with $k = 2$. We use ACO algorithm to build a population in each loop. Then, the two best solutions from all of the solutions, which had been found until that time, are chosen to form the set Ω_{il} on which we apply the local search procedure to improve the result before updating pheromone trail. The framework of this algorithm has been specified in Figure 2.

Local Search Technique. Local search technique is implemented as follows. For a solution $t = t_1 \ldots t_n$, the local search procedure is implemented sequentially from t_1 to t_n. When position t_i is considered, it will be replaced by the remaining characters in set V_i until a better solution is found. The new solution replaces solution t to be the best solution at that time, and the local search procedure ends.

With ACO-CSP and Memetic-CSP procedures mentioned above, ACOM-CSP algorithm uses ACO-CSP in the first 60% loops and Memetic-CSP in the 40% remaining loops.

4 Experimental Results

ACOM-CSP and ACO-CSP are compared to Ant-CSP algorithm both in terms of quality of solutions and runtime by experiments:

1) Run algorithms with a predefined number of loops to compare the effect and runtime.

2) With increasing runtime, all of three algorithms are run on the same dataset to track the convergence of the obtained objective function.

3) Comparing the stability of ACO-CSP and ACOM-CSP by running each algorithm 10 times on the same dataset.

The Ant-CSP, what was used in our paper, is the one that we had re-implemented followed the psuedo-code and specification of Ant-CSP in [11]. The experimental result had showed that the result of our re-implemented Ant-CSP was equivalent to the result of the original Ant-CSP in [11]. The experiments are performed on a computer with: CPU C2D 3.0 GHz, 2GB of RAM, running Windows XP. The sets of input data are randomly generated by Faro and Papalandro's program [11] with alphabet $\sum$ = {A, C, G, T}. Set S contains n strings with n = 10 | 20 30 | 40 | 50 and length of the strings is m = 100 | 200 | … | 1000. The number of loops in Ant-CSP is set to 1500 (as in the Faro and Papalando's paper). We run each of the three algorithms 20 times and compare the average results. The parameters had been set as follows:

- The number of ants in each loop is 10.
- $\rho = 0.06;\ \alpha = 1\quad (\rho = 0.03$ for Ant-CSP (as in [11])
- $\tau_{max} = \frac{1}{|\Sigma|};\ \tau_{min} = \frac{\tau_{max}}{ratio};\ ratio\ = |\Sigma| * length$

The difference between pheromone bounds is thus set proportionally to the number of graph nodes.

4.1 Comparing Effect and Runtime with a Predefined Number of Loops

Ant-CSP's number of the loops is 1500 as Faro et al had set in [11]. Our proposed algorithms' number of the loops is set to 600. Since the experiments show that our algorithms converge very quickly, we do not need more loops. When the number of the loops increases, quality of result only increases insignificantly comparing to runtime. The first phase of ACOM-CSP runs in the first 360 loops. The experimental results are presented in 5 below tables, the best result is displayed in bold.

Table 1. Results for input set of 10 strings of length m

n = 10	Ant-CSP		ACO-CSP		ACOM-CSP	
Length	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime
100	*65.45*	453.8	*59.65*	380.2	***58***	417.1
200	*132.8*	1012.6	*120.95*	787.4	***118***	869.7
300	*196.05*	1656.7	*178.25*	1217.3	***175***	1333.4
400	*263.2*	2396.6	*235.25*	1709.4	***230***	1835.8
500	*330.95*	3119.5	*293.35*	2191.7	***290***	2346.2
600	*399.3*	4064.3	*354.35*	2723.7	***351.7***	2925.3
700	*465.2*	4793.9	*417.65*	3320.7	***406.1***	3547.0
800	*538.7*	5823.9	*473.05*	3862.5	***470.45***	4105.1
900	*603.05*	6909.4	*526.8*	4451.6	***523.2***	4771.3
1000	*673.05*	7975.5	*587.3*	4945.5	***582***	5431.9

Table 2. Results for input set of 20 strings of length m

n = 20	Ant-CSP		ACO-CSP		ACOM-CSP	
Length	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime
100	*70.8*	626.1	*66.25*	453.7	***66***	736.5
200	*139.2*	1335.6	***128***	929.4	***128***	1500.3
300	*209.4*	2149.9	*194.8*	1444.4	***191.5***	2296.4
400	*279.9*	2804.4	*262.35*	2008.3	***256.05***	3137.8
500	*348.8*	3960.0	*325.3*	543.8	***319.1***	3989.4
600	*417.1*	4851.2	*385.05*	3143.7	***379.7***	4997.2
700	*491*	5916.7	*453.15*	3767.6	***447.45***	5927.0
800	*559.25*	7104.4	*512.05*	4463.4	***507.9***	6823.3
900	*635.45*	8253.8	*582*	5096.5	***573.55***	7817.4
1000	*706.05*	9593.6	*641.15*	5711.2	***636.7***	8814.2

Table 3. Results for input set of 30 strings of length m

$n = 30$	Ant-CSP		ACO-CSP		ACOM-CSP	
Length	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime
100	*73.25*	790.0	*70*	520.7	***67.75***	652.9
200	*143.25*	1648.0	***134***	1071.8	***134***	1306.5
300	*214.15*	2661.7	*203*	1653.8	***200***	2002.8
400	*285.9*	3679.7	*273.4*	2258.1	***266.65***	2717.7
500	*355.2*	4728.9	*335.85*	2907.1	***328.9***	3383.0
600	*429.3*	5921.4	*398.4*	3536.0	***397.95***	4055.1
700	*501.95*	7153.7	*463.45*	4228.3	***463***	4667.9
800	*573.3*	8522.5	*531.5*	4872.7	***528.15***	5420.7
900	*646.6*	9906.7	*600.75*	5766.2	***596.35***	6260.8
1000	*717.6*	11341.4	*662.55*	6474.5	***658.1***	6859.1

Table 4. Results for input set of 40 strings of length m

$n = 40$	Ant-CSP		ACO-CSP		ACOM-CSP	
Length	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime
100	*74.1*	961.3	*70.85*	591.3	***70***	748.7
200	*145.35*	1993.6	*139.1*	1209.0	***138.75***	1521.4
300	*217.05*	3116.6	*207.8*	1835.3	***204.1***	2324.8
400	*288.45*	4310.6	*274.95*	2518.3	***270.85***	3160.6
500	*360.55*	5609.9	*346.05*	3211.2	***344***	4006.2
600	*432.2*	6962.3	*409.95*	3959.3	***404.55***	4900.1
700	*507.25*	8374.9	*476.85*	4710.2	***474.05***	5801.1
800	*589.15*	9910.0	*543.15*	5490.6	***539.45***	6770.3
900	*652.75*	11555.5	*609.1*	6282.1	***606.7***	7726.1
1000	*727.55*	13090.0	*675*	7096.8	***672.9***	8745.0

Table 5. Results for input set of 50 strings of length m

$n = 50$	Ant-CSP		ACO-CSP		ACOM-CSP	
Length	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime	$d_H(t,S)$	Runtime
100	75.1	1126.4	**72**	653.5	**72**	854.9
200	146.8	2338.5	141.2	1337.2	**139.4**	1737.8
300	219.65	3641.5	212.8	2050.0	**209**	2642.1
400	292.35	5006.7	279.5	2758.9	**276.25**	3605.7
500	364.5	6465.0	345.6	3552.8	**344**	4543.3
600	437.8	7997.9	414.5	4327.7	**411.5**	5542.1
700	509.25	9669.4	483.35	5153.2	**479.65**	6564.5
800	583.65	11321.8	550.65	5979.0	**545.05**	7618.0
900	657.1	13429.1	621.3	6881.9	**617.1**	8697.3
1000	730.75	14928.0	686.95	7895.9	**685.2**	9778.8

Comment. The experimental results show that:

1) Both ACO-CSP and ACOM-CSP give result considerably better than Ant-CSP. Moreover, their runtime is less than Ant-CSP.

2) When the number of strings increases, the difference in the quality of the solutions of the two algorithms ACO-CSP and ACOM-CSP is negligible. However, according to section 4.3, when using memetic technique, quality of solution is more stable and it takes not much more runtime.

4.2 Comparing Three Algorithms in the Same Runtime

We run three algorithms on the set of 30 strings, each string of length 1000, with runtime from 1000ms to 8000ms to examine the convergence. The result is presented in Figure 3.

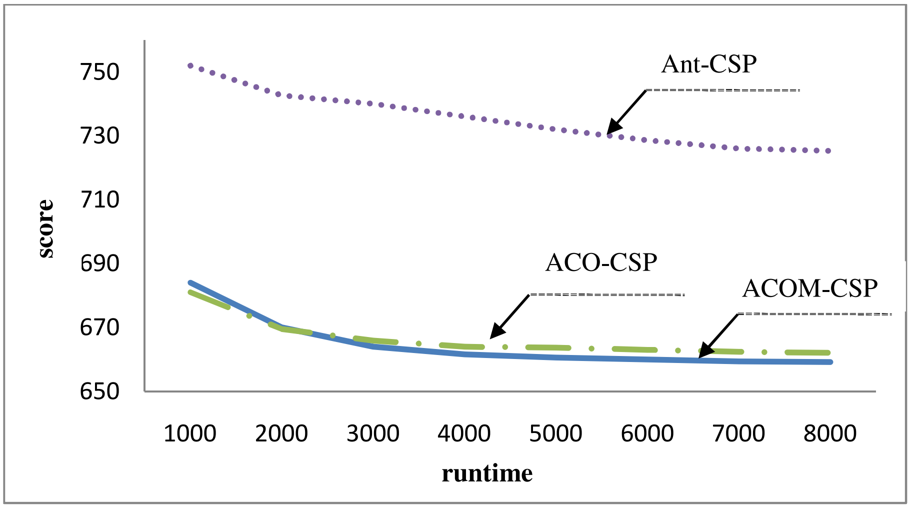


Fig. 3. Examining the convergence of three algorithms

Comment: It can be seen that ACO-CSP and ACOM-CSP converge more quickly than Ant-CSP. From 6500ms (corresponding to 600 loops), quality of solutions remains steadily. That is the reason why the number of loops had been set to 600 to optimize both quality of the solution and runtime.

4.3 Comparing the Stability of ACO-CSP and ACOM-CSP

To comparing the stability of solution quality when runtime increases, we run the two algorithms 10 times on a set of 30 strings, each string of length 1000. The experimental results are presented in Figure 4.

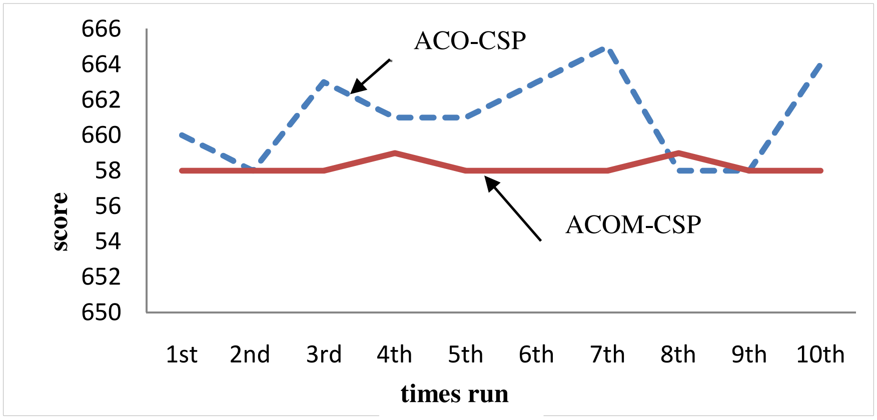


Fig. 4. Comparing the stability of ACO-CSP and ACOM-CSP

5 Conclusion

Applying Ant Colony Optimization to solve the CSP had been first proposed in the Ant-CSP in [11]. The algorithm yields better results than GA and simulated annealing algorithm do. However, it applied neither heuristic information nor local search technique.

We presented in this paper ACOM-CSP, a two-phase algorithm combining ACO technique and memetic algorithm for CSP. ACO-CSP algorithm in the first phase which uses heuristic information and effective pheromone update rule produces better solution quality comparing to Ant-CSP. Using memetic framework in the second phase helps improve the results, and significantly improve the stability of ACO-CSP algorithm.

The experiments show that our new algorithms have outstanding efficiency comparing to Ant-CSP both in terms of solution quality and runtime. Although ACOM-CSP's runtime is slightly higher than ACO-CSP's, the results of ACOM-CSP are better and more stable. When two algorithms run in the same runtime, ACOM-CSP produces better solution quality.

Our algorithms can be applied for the similar problems in genetics and encryption.

Acknowledgements. This work was supported by the Vietnam National Foundation for Science and Technology Development (NAFOSTED).

References

1. Boucher, C., Ma, B.: Closest String with Outliers. BMC Bioinformatics 12(suppl. 1), S55 (2011)
2. Boucher, C., Landau, G.M., Levy, A., Pritchard, D., Weimann, O.: On Approximating String Selection Problems with Outliers. In: Kärkkäinen, J., Stoye, J. (eds.) CPM 2012. LNCS, vol. 7354, pp. 427–438. Springer, Heidelberg (2012)

3. Chen, X.S., Ong, Y.S., Lim, M.H.: Research Frontier: Memetic Computation - Past, Present & Future. IEEE Computational Intelligence Magazine 5(2), 24–36 (2010)
4. Chen, X.S., Ong, Y.S., Lim, M.H., Tan, K.C.: A Multi-Facet Survey on Memetic Computation. IEEE Transactions on Evolutionary Computation 15(5), 591–607 (2011)
5. Chen, Z.Z., Ma, B., Wang, L.: A three-string approach to the closest string problem. J. Comput. Syst. Sci. 78(1), 164–178 (2012)
6. Chen, Z.Z., Wang, L.: Fast Exact Algorithms for the Closest String and Substring Problems with Applicationto the Planted (L. d)-Motif Model. IEEE/ACM Transaction on Computational Biology and Bioinformatics 8(5), 1400–1410 (2011)
7. Do Duc, D., Dinh, H.Q., Hoang Xuan, H.: On the Pheromone Update Rules of Ant Colony Optimization Approaches for the Job Shop Scheduling Problem. In: Bui, T.D., Ho, T.V., Ha, Q.T. (eds.) PRIMA 2008. LNCS (LNAI), vol. 5357, pp. 153–160. Springer, Heidelberg (2008)
8. Do Duc, D., Hoang Xuan, H.: ACOHAP: A novel Ant Colony Optimization algorithm for haplotype inference problem. In: Proc. of the Third International Conference on Knowledge and Systems Engineering (KSE 2011), pp. 128–134 (2011)
9. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: An autocatalytic optimizing process. Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Milano, Italy (1991)
10. Dorigo, M., Stutzle, T.: Ant Colony Optimization. The MIT Press, Cambridge (2004)
11. Faro, S., Pappalardo, E.: Ant-CSP: An Ant Colony Optimization Algorithm for the Closest String Problem. In: van Leeuwen, J., Muscholl, A., Peleg, D., Pokorný, J., Rumpe, B. (eds.) SOFSEM 2010. LNCS, vol. 5901, pp. 370–381. Springer, Heidelberg (2010)
12. Frances, M., Litman, A.: On covering problems of codes. Theory of Computing Systems 30(2), 113–119 (1997)
13. Gasieniec, L., Jansson, J., Lingas, A.: Effcient approximation algorithms for the Hamming center problem. In: Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 905–906 (1999)
14. Gramm, J., Niedermeier, R., Rossmanith, P.: Exact solutions for Closest String and related problems. In: Proc. of the 12th International Symposium on Algorithms and Computation, pp. 441–453. Springer, London (2001)
15. Gramm, J., Niedermeier, R., Rossmanith, P.: Fixed-parameter algorithms for closest string and related problems. Algorithmica, 25–42 (2003)
16. Hertz, G.Z., Hartzell, G.W., Stormo, G.D.: Identication of consensus patterns in unaligned DNA sequences known to be functionally related. Bioinformatics 6(2), 81–92 (1990)
17. Lanctot, J.K., Li, M., Ma, B., Wang, S., Zhang, L.: Distinguishing String Selection Problems. Information and Computation 185, 41–55 (2003)
18. Li, M., Ma, B., Wang, L.: On the closest string and substring problems. Journal of the ACM 49(2), 157–171 (2002)
19. Ma, B., Sun, X.: More Efficient Algorithms for Closest String and Substring Problems. SIAM J. Comput. 39(4), 1432–1443 (2009)
20. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Tech. Rep.Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA (1989)
21. Dinu, L.P., Popa, A.: On the Closest String via Rank Distance. In: Kärkkäinen, J., Stoye, J. (eds.) CPM 2012. LNCS, vol. 7354, pp. 413–426. Springer, Heidelberg (2012)

Evolution of Intrinsic Motives in Multi-agent Simulations

Kamran Shafi, Kathryn E. Merrick, and Essam Debie

School of Engineering and Information Technology, University of New South Wales, Australian Defence Force Academy, Canberra, Australia
{k.shafi,k.merrick,e.debie}@adfa.edu.au

Abstract. In humans, intrinsic motives help us to identify, prioritize, select and adapt the goals we will pursue. A variety of computational models of intrinsic motives have been developed for artificial agents including curiosity, achievement, affiliation and power motivation. Previous research has focused on using models of intrinsic motivation in individual agents, such as developmental robots or virtual agents. However, in humans, intrinsic motives evolve over time in response to personal and social factors. This paper presents evolutionary models of intrinsically motivated agents. The models are evaluated in multi-agent simulations of agents playing iterated prisoner's dilemma games.

Keywords: Intrinsic motivation, achievement, affiliation, power, cognitive agents, evolutionary game theory.

1 Introduction

Computational models of motivation are processes that artificial agents can use to identify, prioritize, select and adapt the goals they will pursue. A variety of computational models of intrinsic motives have been developed including curiosity [1,2], achievement [3], affiliation and power [4] motivation. Previous research has focused on using models of intrinsic motivation in individual agents, such as developmental robots [5] or virtual agents [2]. However, human motives evolve over time in response to both personal and social factors [6]. To create accurate simulations of motivation in artificial agents that take these factors into account, this paper presents evolutionary models of intrinsically motivated artificial agents. Section 2 reviews background literature regarding an existing approach to modelling motive profiles of achievement, affiliation and power motivation in artificial agents. Section 3 presents our new evolutionary agent models based on these profiles. In Section 4 the models are evaluated in multi-agent simulations of agents playing a well-known two-player game: the iterated prisoner's dilemma. Conclusions and future work are presented in Section 5.

2 Background

A range of models of intrinsic motives have been developed [7,8,5,1,2]. However the focus of much of this work has been on the role of motivation for controlling the behavioural responses of an individual agent, or an agent interacting with a small number of other agents. In contrast, this paper explores evolutionary models of intrinsically motivated agents. Specifically, this paper combines existing models of achievement, affiliation and power motivation with a genetic algorithm so that the resulting agents can evolve their motive profiles during interactions with other motivated agents in a society of agents. The remainder of this section is organised as follows. Section 2.1 introduces the computational motive profiles that form the basis of the evolutionary motivated agents in this paper. Section 2.2 introduces the social game used as the experimental setting for this work.

2.1 Computational Motive Profiles

This paper focuses on intrinsic motives for achievement, affiliation and power motivation. Merrick and Shafi [4] developed computational models of these motives with respect to the incentive I_s for succeeding at a goal. In their model, resultant motivational tendency for a goal T_{res} is defined as the sum of sigmoid functions for achievement, affiliation and power motivation:

$$T_{res} = T_{ach} + T_{aff} + T_{pow} \tag{1}$$

where

$$T_{ach} = \frac{S_{ach}}{1 + e^{\rho^+_{ach}(M^+_{ach} - (1 - I_s))}} - \frac{S_{ach}}{1 + e^{\rho^-_{ach}(M^-_{ach} - (1 - I_s))}}$$

$$T_{aff} = \frac{S_{aff}}{1 + e^{\rho^+_{aff}(I_s - M^+_{aff})}} - \frac{S_{aff}}{1 + e^{\rho^-_{aff}(I_s - M^-_{aff})}}$$

$$T_{pow} = \frac{S_{pow}}{1 + e^{\rho^+_{pow}(M^+_{pow} - I_s)}} - \frac{S_{pow}}{1 + e^{\rho^-_{pow}(M^-_{pow} - I_s)}}$$

Parameters of the model and their meaning and ranges are summarized in Table 1.

2.2 Prisoner's Dilemma

The Prisoner's Dilemma (PD) game [9,10] is a two-by-two mixed-motive game. Players choose an action from a list of two possible actions {C, D} between a number of actions. Once all players have chosen an action, a resulting payoff to each player is determined by taking into account the actions of all players. Player 1 is assigned a payoff value V_1 and Player 2 is assigned a payoff value V_2. The best payoff is only obtained if players act differently. This can be thought of as players acting with different (mixed) motives because the situation (game) itself is identical for all players. V_1 and V_2 can have values of T, R, P or S where $T > R > P > S$. The value R is the reward if both players choose C. In other words, R is the reward for a (C, C) outcome. T represents the temptation to defect (choose action D) from the (C, C) outcome. P is the punishment if both players defect (joint D choices leading to a (D, D) outcome). Finally, S is the sucker's payoff for choosing C when the other player chooses D

Table 1. Parameters of achievement, affiliation and power motivation and their value ranges [4]

Param	Description	Range
I_s	Incentive (value) for success	[0, 1]
M^+_{pow}	Turning point of approach component for power	$(-\infty, M^-_{pow})$
M^-_{pow}	Turning point of avoidance component for power	(M^+_{pow}, ∞)
M^+_{aff}	Turning point of approach component for affiliation	(M^-_{aff}, ∞)
M^-_{aff}	Turning point of avoidance component for affiliation	$(-\infty, M^+_{aff})$
M^+_{ach}	Turning point of approach component for achievement	$(-\infty, \infty)$
M^-_{ach}	Turning point of avoidance component for achievement	$(-\infty, \infty)$
ρ^-_{pow}	Gradient of avoidance of power	$[0, \infty)$
ρ^+_{pow}	Gradient of approach to power	$[0, \infty)$
ρ^+_{aff}	Gradient of approach to affiliation	$[0, \infty)$
ρ^-_{aff}	Gradient of avoidance of affiliation	$[0, \infty)$
ρ^-_{ach}	Gradient of avoidance of achievement	$[0, \infty)$
ρ^+_{ach}	Gradient of approach of achievement	$[0, \infty)$
S_{pow}	Relative strength of power motive	$[0, \infty)$
S_{aff}	Relative strength of affiliation motive	$[0, \infty)$
S_{ach}	Relative strength of achievement motive	$[0, \infty)$

3 Evolutionary Game Theoretic Model for Motivated Agents

The evolutionary game-theoretic model presented in this work consists of a population of N heterogeneous motivated agents, denoted as $p_1, p_2, \ldots, p_n, \ldots, p_N$. The agents interact through pair-wise multi-period games and evolve their motive profiles over time. The agents can be created by either randomly sampling from the entire range of a mixed motive profile (see Eq 1 and Table 1) that includes the three computational models of motivation (i.e., achievement, affiliation and power motivations) or according to a specific profile type. The former agent modelling approach can be considered as more generic as it allows evolving agents in the entire motivation profile without distinction. In this paper, only the latter approach is used to create agents; since it allows studying the evolution of agents in an explicit competing environment between agents with different profiles. We refer to this model as *restricted agent model* as opposed to the generic agent model.

During each interaction agents choose actions (cooperate or defect) based on their motivational tendencies. Motivational tendencies are computed according to Equation 1 based on agents' current motive profiles and a vector of incentives for each possible combination of actions (for a two-player binary decision game this corresponds to four incentive values). In this paper, the incentives directly correspond to the values given in the game's payoff matrix and no feedback from the game outcomes is applied to update incentives over time. In other words incentives remain constant throughout all encounters. We refer to this model as a *static incentive model*; noticing that a more sophisticated incentive update rule may also be applied, but not discussed in this paper. As a result, agent p_n is assigned a payoff value from the payoff table based on the combination of the two players' choices as follow:

$$v_n^d = \left\{ \begin{array}{l} P \text{ If both players choose to defect} \\ T \text{ If player } p_n \text{ chooses to defect supposing other player will cooperate} \\ S \text{ If player } p_n \text{ chooses to cooperate supposing other player will defect} \\ R \text{ If both players choose to cooperate} \end{array} \right\}$$

In this paper two models of restricted static incentive motivated agents are presented. In the first model (referred as *Model 1*) we will show experimentally that motivated agents can solve stable strategy games like the PD game. In the second model (referred as *Model 2*) we will study how well the motivated agents can adapt their behaviour to produce a more cooperative model. The objective of this model is to more closely simulate human behaviour where individuals with different motives behave differently but are still able to survive in a population. In the subsections below we provide a detailed description of these models.

3.1 Agent Representation

The same agent representation is used in both models. Each agent is represented by a single chromosome which encodes the parameters of the computational motivation model given in Equation 1. Formally, each agent can be represented using a string representation as following:

$$p_n = \left\{ \begin{array}{l} S_{ach}(n), p_{ach}^{+}(n), p_{ach}^{-}(n), M_{ach}^{+}(n), M_{ach}^{+}(n), \\ S_{aff}(n), p_{aff}^{+}(n), p_{aff}^{-}(n), M_{aff}^{+}(n), M_{aff}^{-}(n), \\ S_{pow}(n), p_{pow}^{+}(n), p_{pow}^{-}(n), M_{pow}^{+}(n), M_{pow}^{-}(n), \\ F_n, type_n \end{array} \right\}$$

The population is initialised by sampling agents from distinct motivational profiles that are created by controlling the parameters in Equation 1. Specifically, the agents with mixed motivations are created by fixing the strength parameters and choosing the other main parameters within given ranges that make sense according to a target profile definition. For instance, a *power* motive profile can be created as seen in Figure 1 by choosing $S_{ach} = 1$, $S_{aff} = 1$, $S_{pow} = 2$ and sampling other parameters from given ranges. In this paper we experiment with agents sampled from three distinct profiles. Table 2 below summarises the parameter ranges chosen to create these profiles. These parameters distribute the three motivation peaks (2 small and one high) roughly evenly over the range 0-1 of incentives. It is important to note that these parameter ranges only make sense if the payoff values of the specific PD game are also distributed evenly over the full range 0-1.

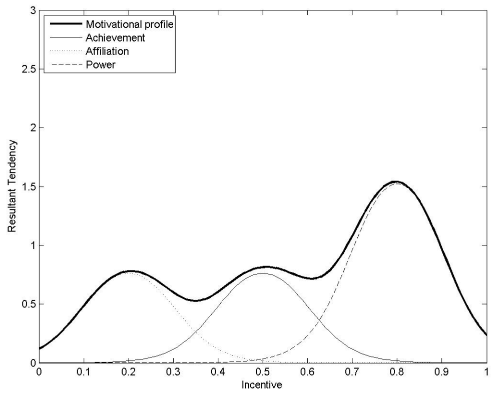


Fig. 1. An example of a motive profile for a power-motivated agent: S_{pow}= 2, S_{ach}= 1, S_{aff}= 1, M^{+}_{ach}= 0.4, M^{-}_{ach}= 0.6, M^{+}_{aff}= 0.3,M^{-}_{aff}= 0.1, M^{+}_{pow}= 0.7, M^{-}_{pow}= 0.9, $\rho^{+}_{ach} = \rho^{-}_{ach} = \rho^{+}_{ach} = \rho^{-}_{ach} = \rho^{+}_{pow} = \rho^{-}_{pow}$= 20

3.2 Evolutionary Dynamics

The evolutionary process consists of a fixed number of generations G in the range $\{1, 2, ..., g, ..., G\}$. In each generation g each agent p_n plays a fixed rounds D, in the range $\{1, 2, ..., d, ..., D\}$, of a PD game with every other agent in the population without discrimination. In each round d of the game, agent p_n deterministically chooses an action a_n using a decision function that maximizes its motivational tendency (see the discussion above for the computation of motivational tendencies).

After all agents have been evaluated through their respective games, a new generation of agents is created using selection, reproduction and mutation operators. In *Model 1*, new agents are created by iteratively selecting parents without replacement from the entire population proportionate to their fitness. In each iteration, a new offspring is created by simply copying the selected parent's genes to its chromosome. The child then undergoes mutation during which each gene of the child's chromosome is allowed to change its value within the ranges given in Table 2 with a given probability μ. The newly created child is then inserted into the population and the process is repeated until the population size is reached.

In *Model 2*, the children are created similarly except that the selection operator is modified to select parents within each profile. This is done by selecting three parents in every iteration, one from each of the three motivational profiles still proportional to their fitness, and producing three children through copying their genes from their respective parents. This niching procedure ensures reproduction opportunities is provided to the agents of all profiles. In addition, contrary to the limited mutation applied in the first model where each approach and avoidance parameter has a specific range, agent's approach and avoidance in this model are allowed to change their values within the whole range [0, 1.2]. Therefore, agents have the ability to change strategies from generation to the other.

3.3 Fitness Functions

In a generation g, fitness for an agent p_n is updated after it finishes playing $N-1$ games with all other players in the population. Fitness in *Model 1* is computed using the relation in Equation 2.

$$F_n^g = \frac{accV_n^g}{D\,(N-1)} \tag{2}$$

where $accV_n^g$ refers to p_n's accumulated payoff over all encounters and as given by Equation 3.

$$accV_n^g = \sum_{d=1}^{D(N-1)} v_n^d \tag{3}$$

Fitness in *Model 2* is computed proportional to the degree of commitment each agent has to the strategy motivated by its initial motive profile. Under this model, when agent p_n finishes $N-1$ games with all other players its fitness F_n^g is updated as follows:

$$F_n^g = \begin{cases} \frac{NumC_n^g}{D(N-1)} & \text{If initial agent strategy is cooperation.} \\ \frac{NumD_n^g}{D(N-1)} & \text{If initial agent strategy is defection.} \end{cases} \tag{4}$$

where $NumC_n^g$ ($NumD_n^g$) is the number of cooperation (defection) decisions that agent p_n^g made in generation g.

4 Experimental Setup

This section presents two experiments with different variations of our evolutionary intrinsically motivated agents. The agents interact in a social game setting defined by the iterative PD game. The impact of motivation in human decision making has previously been studied in this setting [11]. Results identify the existence of individuals with different dominant motive profiles of achievement, affiliation or power motivation. Individuals with different motive profiles act differently, and do not always conform to the predicted ESS of defection. The aim of this section is thus to investigate whether we can develop artificial motive profiles that support the survival of agents with different dominant motives (achievement, affiliation or power) within an evolving society. Based on the two fitness functions (see Section 3.3), we have two distinct experimental setups using restricted agent representation as described earlier:

Table 2. Parameter settings used for restricted agent model

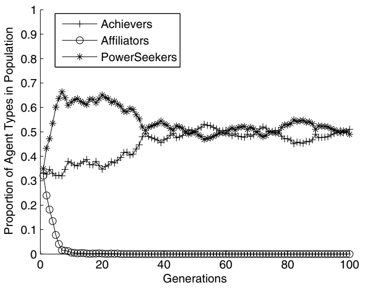
Parameters	Achievers	Affiliator	Power Seekers
S_{ach}	2	1	1
M^{+}_{ach}	0.1-0.3	0.1-0.3	0.1-0.3
M^{-}_{ach}	0.4-0.6	0.4-0.6	0.4-0.6
p^{+}_{ach}	20-100	20-100	20-100
p^{-}_{ach}	20-100	20-100	20-100
S_{aff}	1	2	1
M^{+}_{aff}	0.3-0.45	0.3-0.45	0.3-0.45
M^{-}_{aff}	0.05-0.2	0.05-0.2	0.05-0.2
p^{+}_{ach}	20-100	20-100	20-100
p^{-}_{ach}	20-100	20-100	20-100
S_{pow}	1	1	2
M^{+}_{pow}	0.8-0.95	0.8-0.95	0.8-0.95
M^{-}_{pow}	1.05-1.2	1.05-1.2	1.05-1.2
p^{+}_{ach}	20-100	20-100	20-100
p^{-}_{ach}	20-100	20-100	20-100

1. Model 1: Restricted agent model setup with fitness proportional to the average payoff overall encounters,
2. Model 2: Restricted agent model setup with fitness proportional to the degree of commitment of an agent to its original strategy

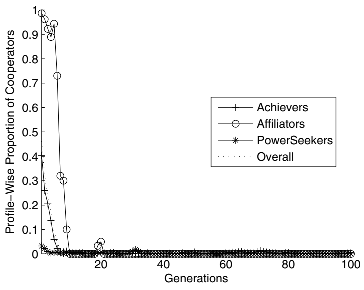
The normalised payoff matrix used for the experiments is as follows: $T = 1$, $R = 0.75$, $P = 0.5$, $S = 0.25$. Here the payoffs are chosen such that they are spread evenly to avoid any skewness that may result in biasing agent decisions other than their motivation. For all experiments, the population size is kept constant at 100 agents. The mutation rate is also kept constant at 0.025. For each experiment we vary the number of rounds from 1 to 10 and report the results accordingly.

4.1 Model 1: Experimental Results and Analysis

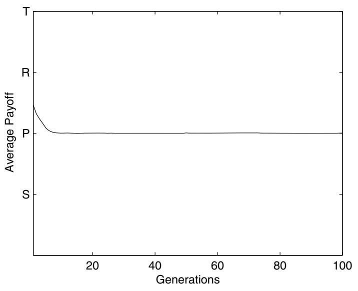
According to the payoff structure of the PD game, defection is considered the initial strategy for power-motivated agents, affiliation-motivated agents tend to cooperate most of the time, and achievement-motivated agents exhibit a mixture of cooperating and defecting strategies. It is also worth noting that defection in a PD game is evolutionary stable strategy (ESS)[12], thus those agents adopting such strategy can survive in a population of agents with different strategies. Reflecting these facts on our model, Figure 2(a) shows the evolution of agents with different motives in an explicit competing environment. As can be seen, all agents which choose to cooperate have died out of the population very quickly (before generation 20). This is represented by affiliators and the proportion of achievers whose initial strategy is cooperation. Figure 2(b) shows the proportion of co-operators of each type of agents. As is shown in the figure, the power-motivated agents never cooperates which allow them to have higher chance of reproduction while the proportion of co-operators of both affiliation and achievement-motivated agents declined rapidly before the 20th generation. In Figure 2(c), where R, S, T and P refer to Reward for mutual cooperation, Sucker's payoff, Temptation to defect and Penalty to mutual defection, it is shown that the average population payoff is 0.5 which confirms the results by showing that all agents in the population turn to defection strategy.



(a) The average proportion of three types of agents in the population.



(b) The average normalized proportion of cooperators in each profile.

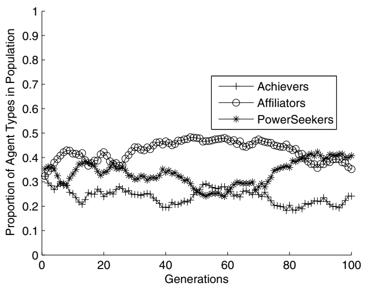


(c) The average fitness of the population over 100 generations.

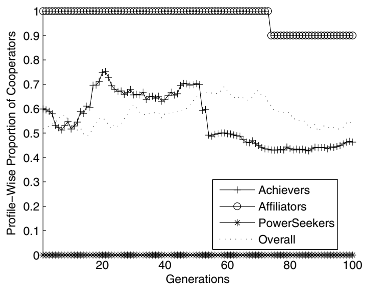
Fig. 2. Statistics for motivated agents Model 1. The population is initialized uniform randomly with the three agent types.

4.2 Model 2: Experimental Results and Analysis

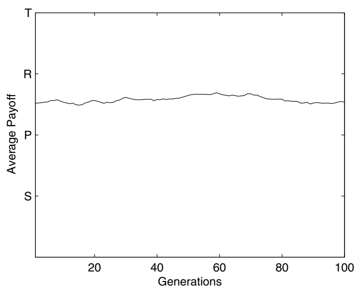
While the first model favours defecting agents over cooperating ones (in this case power seekers and a proportion of achievers) by allowing them to evolve more frequently, the second model is built in a neutral way such that agents are not evaluated based on the direct payoff they gain. However, agents are evaluated based on their commitment to the strategy defined by their initial motive profile. For example, agents whose initial strategy is defection (cooperation) and they keep defecting (cooperating) throughout the games with other agents get higher fitness. As is shown in Figure 3(a), all agents with different motives could survive throughout the simulation with approximately the same probability. Figure 3(b) shows the proportion of cooperating agents from each type where the higher proportion of cooperation is due to the survival of affiliation and achievement-based agents. Now, there does not exist any stable strategy in such a neutral environment but a mix of strategies can be found which leads to an average population payoff swings between 0.5 and 0.75 as is shown in Figure 3(c) where R, S, T and P refer to Reward to mutual cooperation, Sucker's payoff, Temptation to defect and Penalty to mutual defection.



(a) The average proportion of three types of agents in the population.



(b) The average normalized proportion of cooperators in each profile.



(c) The average payoff of the population over 100 generations.

Fig. 3. Statistics for motivated agents Model 2. The population is initialized uniform randomly with the three agent types.

5 Conclusion and Future Work

In this paper we presented two evolutionary models of intrinsically motivated agents. The first one was able to successfully solve stable strategy games while in the second model we presented a simulation of more realistic human motive interaction as reported in psychology literature where the agents strive to survive by adapting behaviour. In the experiments conducted in this paper we have used a "static incentive" model. However, in future work we will extend our study on different models where the incentives evolve over time.

Acknowledgments. This work was supported by a UNSW@ADFA Early Career Researcher Grant: UNSWA SIR30 Z6300 0000 00 PS23595.

References

1. Saunders, R.: Curious Design Agents and Artificial Creativity: A Synthetic Approach to the Study of Creative Behaviour. PhD thesis, Department of Architectural and Design Science, Faculty of Architecture, University of Sydney (2002)
2. Merrick, K., Maher, M.L.: Motivated reinforcement learning: curious characters for multiuser games. Springer, Berlin (2009)
3. Merrick, K.: A computational model of achievement motivation for artificial agents. In: Autonomous Agents and Multi-Agent Systems (AAMAS 2011), Taiwan, pp. 1067–1068 (2011)
4. Merrick, K., Shafi, K.: Achievement, affiliation and power: motive profiles for artificial agents. Adaptive Behavior 9(1), 40–62 (2011)
5. Oudeyer, P.Y., Kaplan, F., Hafner, V.V.: Intrinsic motivation systems for autonomous mental development. IEEE Transactions on Evolutionary Computation 11(2), 265–286 (2007)
6. Heckhausen, J., Heckhausen, H.: Motivation and action. Cambridge University Press (2008)
7. Baldassarre, G.: What are intrinsic motivations? a biological perspective. In: 2011 IEEE International Conference on Development and Learning (ICDL), vol. 2, pp. 1–8 (August 2011)
8. Oudeyer, P.Y., Kaplan, F.: What is intrinsic motivation? a typology of computational approaches. Frontiers in Neurorobotics 1(6) (2007) (online, open–access)
9. Rapoport, A., Chammah, A.: Prisoner's dilemma: A study in conflict and cooperation, vol. 165. Univ. of Michigan Press (1965)
10. Poundstone, W.: Prisoner's Dilemma. Doubleday, New York (1992)
11. Terhune, K.W.: Motives, situation, and interpersonal conflict within prisoner's dilemma. Journal of Personality and Social Psychology 8(3, pt. 2), 1–24 (1968)
12. Nowak, M., Sasaki, A., Taylor, C., Fudenberg, D.: Emergence of cooperation and evolutionary stability in finite populations. Nature 428(6983), 646–650 (2004)

A Hybrid Particle Swarm Optimization Approach to Bernoulli Mixture Models

Faezeh Frouzesh, Yuichi Hirose, Shirley Pledger, and Mahdi Setayesh

School of Mathematics, Statistics and Operations Research,
Victoria University of Wellington, New Zealand
{Faezeh.Frouzesh,Yuichi.Hirose,Shirley.Pledger}@msor.vuw.ac.nz,
Mahdi.Setayesh@ecs.vuw.ac.nz
http://www.vuw.ac.nz

Abstract. The use of mixture models in statistical analysis is increasing for dataset with heterogeneity and/or redundancy in the data. They are likelihood based models, and maximum likelihood estimates of parameters are attained by the use of the expectation maximization (EM) algorithm. Multi-modality of the likelihood surface means that the EM algorithm is highly dependent on starting points and poorly chosen initial points for the optimization may lead to only a local maximum, not a global maximum. The aim of this paper is to introduce a hybrid method of Particle Swarm Optimization (PSO) as a global optimization approach and the EM algorithm as a local search to overcome this problem and then it will be compared with different methods of choosing starting points in the EM algorithm.

Keywords: Maximum likelihood (ML), Expectation maximization (EM) algorithm, Clustering techniques, particle swarm optimization (PSO).

1 Introduction

Finite mixture models provide a natural representation for great flexibility in fitting models for continuous or discrete outcomes that are observed from populations including a finite number of homogeneous subpopulations [14]. The mixture models also provide a convenient formal setting for model-based clustering. There are a lot of applications of finite mixture models in the social and behavioral sciences, biological, physical and environmental sciences, engineering, finance, medicine and psychiatry among many other fields [8]. The most important advantage of mixture models is that the model can have quite a complex distribution through choosing its components to have an accurate local area in order to fit the best distribution to the model. As a result, finite mixture models can be used in situations where a single parametric family is unable to provide an adequate model for local variations in the observed data [8].

Finding estimation of parameters in finite mixture models is of practical importance in pattern recognition and other related fields. These parameters are estimated by maximizing the likelihood. The EM algorithm is a standard approach to finding maximum likelihood in a variety of problems where there is incomplete information or missing data. This algorithm was proposed by Dempster et al. in 1977 [9]. Beside all the advantages of the EM algorithm, such as its simplicity, monotonic convergence and its natural statistical interpretation, it has some general drawbacks which have been noted in the literature [14]. The EM algorithm is an iterative hill-climbing method whose performance is highly dependent on its initial points, especially in the multivariate context because of the multi-modality of the likelihood function. Therefore, its sensitivity to the initial points and trapping in local optima are the main disadvantages of the EM algorithm.

For the first time, Laird [12] proposed a method based on a grid search for choosing the initial values in 1978. Woodward et al. [19] used an ad-hoc quasi-clustering technique to obtain starting values for the EM algorithm. Bithell [7] suggested an initial classification of the data by maximizing the within sum of squares criterion. Seidel et al. [18] used a Random starts method in the EM algorithm for finite exponential mixtures. Their research showed that starting with several random initial values was preferable in order to ensure that the global maximum is obtained. In 2006, Nasser Sara [15] used the K-mean algorithm to initialise the EM algorithm and the results were promising. In 2009, Bessadok et al. [6] suggested an initiative method via variable neighbourhood search to overcome the problem of choosing starting points in the EM algorithm which often gets trapped at local maxima. To the authors' knowledge, the K-mean algorithm is the only clustering technique whose centroids have been used as starting points in the EM algorithm.

In this paper, we aim to present a novel method to overcome the problem of choosing initial point in the EM algorithm. Particle Swarm Optimisation (PSO) is a meta-heuristic method that has successfully been used to solve global optimisation problem. The ease of implementation, fewer operations, limited memory for each particle and high speed of convergence are the main advantages of PSO in comparison with other heuristic methods, such as genetic algorithms [2]. PSO has been used for the optimisation of parameters in Gaussian mixture models [4][17]. But it has never been used for optimisation in Bernoulli mixture models up to now. The main goals of this paper are as follows:

- investigate the performance of the use of different most well-known clustering techniques to choose the initial points for the EM algorithm in Bernoulli mixture models for binary datasets;
- develop a new PSO-based algorithm as a global search method combined with the EM algorithm as a local search method in order to more likely to cover all the potential solutions;
- develop a new fitness function for the Hybrid PSO-based algorithm along with a simple encoding scheme;
- compare the performance of the new proposed method to the EM algorithm which is initialised by different clustering methods in terms of the goodness of fitted statistical models.

2 Background

2.1 Data and Notations

The datasets used in this paper consist of an $n \times p$ matrix $\mathbf{Y}$ of binary data. Each row corresponds to a subject and each column corresponds to one variable. The value of each element y_{ij} is 1 if there is "success"on a Bernoulli trial with parameter θ_{ij}, and 0 if there is failure. In ecological presence/absence data, the rows may represent n species while the columns are p quadrants or samples. It is desirable to know if the rows can be grouped (clustered) into two or more relatively homogeneous groups. The subjects are more similar (in their pattern of occurrence) within groups and dissimilar between different clusters. We list notations used in this paper:

$$\begin{aligned} L &= \text{Likelihood} \\ l &= \text{log likelihood} \\ l_c &= \text{log likelihood under complete knowledge} \\ \theta_{rj} &= \text{Bernoulli parameter for } y_{ij}, i = 1, ...n, j = 1, ...p \\ z_{ir} &= \text{Indicator of row } i \text{ being in row-group } r \\ \pi_r &= \text{A priori membership probabilities } (r = 1, ...R,\ 0<\pi_r<1,\ \textstyle\sum_{r=1}^{R} \pi_r = 1) \\ \pi_{ir} &= \text{A posteriori probability row i from row-group } r \end{aligned}$$

2.2 EM Algorithm for Finite Mixture Models

For a given dataset and any statistical model, the most commonly used method to estimate the parameters of the model is Maximum Likelihood (ML) method. Therefore, we need to construct the likelihood which is the probability of observing the data set. The EM algorithm as a standard method is often used to estimate the parameters of the model in two steps: the expectation step (E-step) and the maximisation step (M-step). The E-step calculates the estimation of the log-likelihood evaluated using the current estimate for the latent variables. For a discrete distribution, the likelihood is the product of the probabilities for the individual data points. Hence, the overall likelihood is:

$$L(\theta, \pi|y) = \prod_{i=1}^{n} \left[\sum_{r=1}^{R} \pi_r \prod_{j=1}^{p} \theta_{rj}^{y_{ij}} (1 - \theta_{rj})^{1-y_{ij}} \right] \tag{1}$$

and the log likelihood is:

$$l(\theta, \pi|y) = \sum_{i=1}^{n} \log \left[\sum_{r}^{R} \pi_r \prod_{j=1}^{p} \theta_{rj}^{y_{ij}} (1 - \theta_{rj})^{1-y_{ij}} \right]. \tag{2}$$

With consideration of the missing information as an $n \times R$ group membership matrix, Z, where $z_{ir} = 1$ if case i is in group r otherwise 0. Then, the likelihood under complete knowledge is:

$$l_c(\theta, \pi|y, z) = \sum_{i=1}^{n} \sum_{r=1}^{R} z_{ir} \sum_{j=1}^{p} \left[y_{ij} \log(\theta_{rj}) \right. \tag{3}$$

$$\left. +(1 - y_{ij}) \log(1 - \theta_{rj}) \right] + \sum_{i=1}^{n} \sum_{r=1}^{R} z_{ir} \log \pi_r$$

After estimating the missing data in the E-step, the M-step can use the complete data to find the optimal parameters. The partial derivative with respect to θ_{rj} is found, and equated to zero, give us an exact mathematical solution,

$$\left[\frac{\partial l_c}{\partial \theta_{rj}}\right] = \sum_{i=1}^{n} z_{ir}\left[\frac{y_{ij}}{\theta_{rj}} - \frac{1-y_{ij}}{1-\theta_{rj}}\right] = 0, \ \forall_r, \forall_j \tag{4}$$

Therefore, $\hat{\theta}_{rj} = \frac{\sum_i z_{ir} y_{ij}}{\sum_i z_{ir}}$.

Also, under complete data, the maximum likelihood estimates of π_1 and π_2 are simply $\hat{\pi}_r = \frac{\sum_i z_{ir}}{n}$, i.e., the proportion of the n cases in group r. There are other calculation in the E-step, where we use current parameter estimates, $\hat{\theta}_{rj}$ and $\hat{\pi}_r$ to find the expected value of z_{ir}. For row i the posterior probability of being in group r is,

$$\hat{z}_{ir} = \frac{\pi_r \prod_{j=1}^{p} \theta_{rj}^{y_{ij}} (1-\theta_{rj})^{1-y_{ij}}}{\sum_{r=1}^{R} \pi_r \prod_{j=1}^{p} \theta_{rj}^{y_{ij}} (1-\theta_{rj})^{1-y_{ij}}}. \tag{5}$$

2.3 Particle Swarm Optimisation

PSO is originally attributed to Eberhart and Kennedy in 1995 [11]. It is a population based stochastic optimisation technique which iteratively optimises candidate solutions (which are called particles) regarding to their fitnesses. We can move these particles through multi-dimensional search space according to a simple mathematical formula over the particle's position and velocity. Each particle moves based on its personal best position and the position of its best neighbour. Moving the swarm of particles toward the best solutions will be expected. The position of ith particle is represented as the vector $\boldsymbol{X}_i(t) = (x_{i1}(t), x_{i2}(t), ..., x_{in}(t))$ in an n-dimensional search space at time t. The position of the particle is changed based on its own experience (particle and memory influence) and that of its neighbours (swarm influence). A particle's position, $X_i(t)$ is updated at each iteration of PSO by adding its velocity at time t represented by $\boldsymbol{V}_i(t)$:

$$X_i(t+1) = X_i(t) + V_i(t+1). \tag{6}$$

The velocity is changed according to three components: current motion influence, particle memory influence, and swarm influence:

$$V_{i,j}(t+1) = wV_{i,j}(t) + C_1 r_1 (X_{pbest_{i,j}} - X_{i,j}(t)) + C_2 r_2 (X_{leader,j} - X_{i,j}(t)) \tag{7}$$

where w denotes the inertia weight to control the impact of the previous velocity; C_1 and C_2 are the self and swarm confidence learning factors which control how far a particle will go in a single iteration. In the other word, they represent the attraction of a particle toward its best previous position and the best particle of the population; r_1 and r_2 are uniform random variables between 0 and 1; X_{pbest} denotes the personal best position of ith particle so far and X_{leader} represents the position of a particle to guide and lead other particles toward better regions of the search space.

3 PSO for Optimisation of Parameters of Bernoulli Mixture Model

Since the parameters of the Bernoulli mixture model (BMM) are in Euclidean space, we expect that PSO as a global optimisation method in continuous search spaces would be a good candidate to optimise these parameters. In this section, we first introduce the fitness function used in our PSO algorithm and then describe the particle encoding representing the parameters of BMM.

3.1 Fitness Function for PSO

We first merge two steps of the EM algorithm in order to develop a proper fitness function to be optimised by PSO. In order to get a proper fitness function, we first substitute the posterior probability $\hat{z}_{ir}$ (see Equation 5) in the equation of likelihood under complete knowledge (see Equation 3) as follows:

$$\begin{aligned} l_c(\theta, \pi | y, z) = \sum_{i=1}^{n} \sum_{r=1}^{R} \hat{z}_{ir} \sum_{j=1}^{p} \Big[y_{ij} \log(\theta_{rj}) \\ + (1 - y_{ij}) \log(1 - \theta_{rj}) \Big] + \sum_{i=1}^{n} \sum_{r=1}^{R} \hat{z}_{ir} \log \pi_r \end{aligned} \tag{8}$$

where $\hat{z}_{ir}$ is given by Equation 5. Since the parameter values which optimise the likelihood under complete knowledge (Equation 3) are the same as the parameters which optimise the log likelihood in Equation 2, the maximum log likelihood is computed through substitution of these parameters in Equation 2. As described earlier, the parameter π in the complete likelihood should satisfy the constraint $\sum_{r=1}^{R} \pi_r = 1$. Hence, it is necessary to use a Lagrange multiplier [5] to satisfy this constraint. So, let $Q = l_c + \lambda(\sum_{r=1}^{R} \pi_r - 1)$. The partial derivatives of Q with respect to π_r is found and equated to zero as follows:

$$\frac{\partial Q}{\partial \pi_r} = \frac{\partial l_c}{\partial \pi_r} + \lambda = \sum_{i=1}^{n} z_{ir} \frac{1}{\pi_r} + \lambda = 0. \tag{9}$$

Based on the constraint, the following equation is obtained:

$$1 = \sum_r \pi_r = \sum_r \Big(-\frac{\sum_{i=1}^{n} z_{ir}}{\lambda}\Big) = -\frac{\sum_i \sum_r z_{ir}}{\lambda} = -\frac{\sum_i 1}{\lambda} = \frac{-n}{\lambda}. \tag{10}$$

Therefore, the function Q which should be optimised by PSO is as follows,

$$Q = l_c - n\Big(\sum_{r=1}^{R} \pi_r - 1\Big) \tag{11}$$

3.2 Particle Encoding

Since PSO needs a particle encoding to represent the parameters which should be optimized, we develop a very simple encoding to represent all required parameters in the likelihood for Bernoulli mixture models. The parameters are $\boldsymbol{\theta} = (\theta_{11}, \theta_{12}, ..., \theta_{rp})$ and $\pi = (\pi_1, ..., \pi_r)$ so that $0<\theta_{11}, \theta_{12}, ..., \theta_{rp}<1$ and $0<\pi_1, ..., \pi_r<1$. We should have a vector of parameters like $\boldsymbol{\psi} = (\theta_{11}, \theta_{12}, ..., \theta_{rp}, \pi_1, ..., \pi_r)$ represented by each particle.

3.3 PSO with EM Algorithm

Since the EM algorithm is a well-known approach as a local search, finding a hybrid method which can lead to a global search is worthwhile. Hence, PSO as global search is a good candidate method to mix with the EM algorithm. The Hybrid PSO and EM algorithm can be represented as the follows:

Algorithm 1. Hybrid PSO algorithm as a heuristic method to overcome the drawbacks of the EM algorithm

```
 1: Initialize the position and velocity of particle
 2: repeat
 3:     for all particle $P$ in population do
 4:         Calculate fitness value of $P$
 5:         Update its personal best if fitness value is better than best fitness
            value pbest, and set current position as pbest
 6:     end for
 7:     for all particle $P$ in population do
 8:         Find local best particle from neighbourhood
 9:         Compute particle velocity (equation 7)
10:         Update particle position (equation 6)
11:         Initialize EM algorithm with the position of current particle
12:         Improve the position of current particle through applying EM algorithm
13:     end for
14: until stopping criterion
```

4 Experiment Design

To assess the starting points for the EM algorithm that are found by the Random starts, K-means and Split off method, and compare the performance the proposed method, we consider 3 challenging datasets. The first one is from [13] which is the presence/absence measure of the 25 most abundant plant species ($n = 25$) on 17 plots ($p = 17$) from a grazed meadow in Steneryd Nature Reserve in Sweden. The second data come from the R package “bipartite”which can be download from [10]. It is a plant-pollinator network from Memmott (1999). The rows are insect species ($n = 79$) and the columns are plant species ($p = 25$). A one is recorded if that insect species was recorded pollinating the flowers of that plant species. It started as a count data matrix (i.e. the number of times that type of insect was seen pollinating that type of flower), but this is the binary version. The third one is from the book [16] which is nine rodent species ($p = 9$) in 28 urban canyons ($n = 28$) in Southern California. These are challenging and substantial binary datasets which are representative of a range of matrix sizes.

Akaike information criterion (AIC, [1]) is an information-theoretic criterion for model selection based on the statistical likelihood function. In fact, it provides a method for comparison among models. In the case of several candidate models for a given dataset, they are ranked according to their AIC, and the model with the minimum AIC is the best model for fitting. The value of AIC represents the amount of lost information in the case of choosing a model to estimate the true model. The best model for fitting is the model which loses the minimum amount of information and has the least AIC. AIC is computed as:

$$AIC = 2K - 2\log(L) \tag{12}$$

where K indicates the number of parameters in the model and L is the maximized likelihood of a fitted statistical model [3]. $AICc$ is the second-order correction of AIC for small sample size n with respect to K which is calculated as the following:

$$AICc = -2\log(L) + 2K(n/(n - K - 1)). \tag{13}$$

In this paper, we use AIC and $AICc$ as a measure of the goodness of fitted model and compare different methods for finding initial points for EM algorithm based on the values of AIC and $AICc$.

5 Results and Discussion

We first applied Random starts, K-means, Split off and Hybrid PSO in dataset 1, 2 and 3. Since these methods are indeterministic, we run these algorithms 30 times for each dataset when we have three to six row clusters and then calculate 95% confidence interval for the maximum likelihood, AIC and $AICc$ for each dataset. Biologists are often interested to have three to six row groups. Hence, we compare the hybrid PSO with K-means, random starts and divisive methods when we have three to six clusters. Tables 1, 2 and 3 demonstrate maximum likelihood, P-value based on student's t-test to compare the maximum log likelihood resulting from the proposed method with other methods, the number of parameters in the Bernouli mixture models, AIC, $AICc$ and the number of clusters (column R) for datasets 1, 2 and 3. We arranged a student's t-test to compare the performance of the proposed method with other methods. As can be seen in Table 1, the maximum likelihood (ML), AIC and $AICc$ for the model with three, four and five row clusters, which are found by the Hybrid PSO method, are statistically better than those values found by the EM algorithm utilising other methods for initialising starting points. For the model with six row clusters, our experiments show that the model found by our proposed method is statistically equivalent with the models found by other methods for dataset one.

The results for dataset 2 can be seen in Table 2. In this dataset, the comparison of ML, AIC and $AICc$ for the models with three, four and five row clusters

Table 1. Comparison of Hybrid PSO method with the traditional methods when we have three to six row clusters in dataset 1

Method	Max.ll	P-value	npar	*AIC*	*AICc*	R
Random starts	-182.20 ± 0.00	2.24E-63	53	470.41 ± 0.00	486.46 ± 0.00	3
K-means	-182.66 ± 0.00	1.25E-67	53	471.32 ± 0.00	487.38 ± 0.00	3
Split off	-193.14 ± 10.83	0.03	53	492.28 ± 43.33	508.34 ± 43.33	3
Hybrid PSO	-181.06 ± 0.00	–	53	468.12 ± 0.01	484.18 ± 0.01	3
Random starts	-160.65 ± 0.07	3.1E-10	71	463.31 ± 0.29	493.17 ± 0.29	4
K-means	-162.21 ± 0.00	–	71	466.43 ± 0.00	496.29 ± 0.00	4
Split off	-178.94 ± 10.14	0.0011	71	499.89 ± 40.56	529.75 ± 40.57	4
Hybrid PSO	-160.30 ± 0.00	–	71	462.60 ± 0.00	492.46 ± 0.00	4
Random starts	-146.17 ± 0.10	7.1E-06	89	470.34 ± 0.42	519.38 ± 0.42	5
K-means	-145.90 ± 0.00	1.35E-06	89	469.81 ± 0.03	518.85 ± 0.03	5
Split off	-167.96 ± 9.26	6.27E-05	89	513.93 ± 37.06	562.97 ± 37.06	5
Hybrid PSO	-145.87 ± 0.00	–	89	469.75 ± 0.00	518.79 ± 0.00	5
Random starts	-133.49 ± 0.27	0.90	107	480.09 ± 1.09	555.49 ± 1.09	6
K-means	-133.02 ± 0.11	0.91	107	480.04 ± 0.46	554.55 ± 0.46	6
Split off	-158.98 ± 8.56	0.19	107	531.96 ± 34.25	606.46 ± 34.25	6
Hybrid PSO	-130.93 ± 40.15	–	107	472.00 ± 642.28	544.87 ± 943.06	6

Table 2. Comparison of Hybrid PSO method with the traditional methods when we have three to six row clusters in dataset 2

Method	Max.ll	P-value	npar	*AIC*	*AICc*	R
Random starts	-578.67 ± 2.26	3.16E-13	77	1311.35 ± 9.07	1317.85 ± 9.07	3
K-means	-582.42 ± 0.10	5.12E-50	77	1318.84 ± 59.94	1325.34 ± 59.94	3
Split off	-586.08 ± 0.00	1.1E-56	77	1326.15 ± 0.00	1332.65 ± 0.00	3
Hybrid PSO	-564.21 ± 0.10	–	77	1282.42 ± 0.42	1288.92 ± 0.42	3
Random starts	-564.02 ± 0.45	6E-09	103	1282.04 ± 1.81	1288.54 ± 1.81	4
K-means	-545.84 ± 3.05	0.9739	103	1279.67 ± 12.21	1309.35 ± 12.21	4
Split off	-572.86 ± 0.00	7.35E-13	103	1351.72 ± 0.00	1363.40 ± 0.00	4
Hybrid PSO	-545.75 ± 4.39	–	103	1245.50 ± 17.57	1309.18 ± 17.57	4
Random starts	-564.28 ± 0.21	4.55E-13	129	1386.56 ± 0.85	1289.06 ± 0.85	5
K-means	-528.56 ± 4.03	0.3818	129	1315.12 ± 16.14	1333.59 ± 16.14	5
Split off	-557.15 ± 0.00	5.3E-11	129	1372.30 ± 0.00	1390.77 ± 0.00	5
Hybrid PSO	-525.21 ± 6.20	–	129	1308.42 ± 24.82	1326.89 ± 24.82	5
Random starts	-505.77 ± 0.12	0.9462	155	1321.54 ± 0.49	1289.30 ± 0.49	6
K-means	-505.77 ± 2.80	0.9482	155	1321.53 ± 11.18	1348.48 ± 11.18	6
Split off	-542.40 ± 0.00	6.59E-08	155	1394.81 ± 0.00	1421.75 ± 0.00	6
Hybrid PSO	-505.42 ±10.09	–	155	1320.84 ± 38.40	1347.49 ± 44.18	6

found by the different methods show that the performance of hybrid method is better than the old methods. Although, for six row clusters, the performance of the Hybrid PSO method is better than Split off method, its performance is equivalent with Random starts and *K*-means.

Table 3 shows the results for dataset 3. In this dataset, for three row clusters, the performance of the Hybrid method is better than Random starts and Split

Table 3. Comparison of Hybrid PSO method with the traditional methods when we have three to six row clusters in dataset 3

Method	Max.ll	P-value	npar	*AIC*	*AICc*	R
Random starts	-87.8420 ± 0.00	–	29	233.6840 ± 0.00	242.1000 ± 0.00	3
K-means	-85.0119 ± 0.08	0.9983	29	228.0230 ± 0.35	236.4390 ± 0.35	3
Split off	-87.4620 ± 0.00	–	29	232.9230 ± 0.00	241.3400 ± 0.00	3
Hybrid PSO	-84.9210 ± 0.00	–	29	227.8410 ± 0.00	236.2570 ± 0.00	3
Random starts	-78.5822 ± 0.19	0.7039	39	235.1637 ± 4.81	250.7087 ± 4.81	4
K-means	-79.5077 ± 0.52	0.0010	39	237.0149 ± 2.10	252.5599 ± 2.10	4
Split off	-82.9460 ± 0.00	–	39	243.8930 ± 0.00	2259.4380± 0.00	4
Hybrid PSO	-78.5450 ± 0.00	–	39	235.0890 ± 0.00	250.6340 ± 0.00	4
Random starts	-74.9245 ± 0.06	0.3981	49	247.8491 ± 0.24	273.2227 ± 0.02	5
K-means	-74.9465 ± 1.17	0.4084	49	247.8933 ± 4.71	273.2663 ± 4.71	5
Split off	-79.0220 ± 0.00	0.012	49	256.0440 ± 0.00	281.4170 ± 0.00	5
Hybrid PSO	-72.9772 ± 4.45	–	49	243.9555 ± 1.76	269.3285 ± 1.76	5
Random starts	-71.2060 ± 4.77	0.9185	59	260.4122 ± 1.40	298.7371 ± 0.07	6
K-means	-71.3434 ± 0.01	0.1231	59	260.6861 ± 0.06	299.0111 ± 0.06	6
Split off	-74.7390 ± 0.00	1.21E-28	59	267.4790 ± 0.00	305.8030 ± 0.00	6
Hybrid PSO	-71.4571 ± 0.14	–	59	260.2467 ± 2.01	298.1400 ± 8.02	6

off methods, however it is equivalent to the K-means method. The model found by Hybrid PSO in dataset 3 with four row clusters, has a better ML, *AIC* and *AICc* than K-means and Split off methods and this model is approximately equivalent to Random starts. The performance of Random starts and K-means in dataset 3 with five and six row clusters, are the same as proposed method, but proposed method works better than Split off method.

6 Conclusion

This paper presented a new Hybrid PSO algorithm with a new fitness function and a new encoding scheme for Bernoulli mixture models which makes the EM algorithm independent of the initial points. Different methods (Random starts, K-means, Split off and Hybrid PSO methods) for choosing the best starting points for the EM algorithm were evaluated and a new heuristic algorithm which combines PSO and the EM algorithm was introduced. Since the EM algorithm is applied as a local search and its result is highly dependent on initial points, a Hybrid PSO approach could improve the EM algorithm to be independent of initial points. The results showed that in most cases, the performance of the Hybrid PSO is statistically better than the other methods and in a few cases, it is equivalent to the other methods. Our result showed that the proposed method performs better than other methods in 23 cases out of 36 and in the remaining cases as well as other methods. However the computation time for hybrid PSO is more than the other traditional methods. Since PSO has been used as a clustering method in literature, as a future work, the performance of using PSO as a clustering method can be investigated to choose the initialization points in EM algorithm.

References

1. Akaike, H.: Information theory as an extension of the maximum likelihood principle. Academiai Kiado (1973)
2. AlRashidi, M.R., El-Hawary, M.E.: A survey of particle swarm optimization applications in electric power systems. Transaction on Evolutionary Computation 13(4), 913–918 (2009)
3. Anderson, D.: Model Based Inference in the Life Sciences. Springer (2008)
4. Ari, C., Aksoy, S.: Maximum likelihood estimation of Gaussian mixture models using particle swarm optimization. In: Proceedings of International Conference on Pattern Recognition (ICPR), pp. 746–749 (2010)
5. Bertsekas, D.P., Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific (1999)
6. Bessadok, A., Hansen, P., Rebai, A.: EM algorithm and variable neighborhood search for fitting finite mixture model parameters. In: Proceedings of International Conference on Computer Science and Information Technology, pp. 725–733 (2009)
7. Bithell, J.F.: Computer-assisted analysis of mixtures and applications. meta-analysis, disease mapping and others. Statistics in Medicine 20(19), 2990–2991 (2001)
8. Deb, P.: Finite mixture models. Summer north American stata users' group, Stata Users Group (2008)
9. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society 39(1), 1–38 (1977)
10. Dormann, C.F., Gruber, B., Frund, J.: Introducing the bipartite package: Analysing ecological networks. R News 8(2), 8–11 (2008)
11. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
12. Laird, N.: Nonparametric maximum likelihood estimation of a mixing distribution. Journal of the American Statistical Association 73(364), 805–811 (1978)
13. Manly, B.F.: Multivariate Statistical Methods: a Primer. Boca Raton (2005)
14. Mclachlan, G., Peel, D.: Finite Mixture Models, 1st edn. Wiley Series in Probability and Statistics. Wiley-Interscience (2000)
15. Sara, V.G.N., Rawan, A.: A modified fuzzy K-means clustering using expectation maximization. In: Proceedings of IEEE World Congress on Computational Intelligence (2006)
16. Quinn, G.P., Keough, M.J.: Experimental Design and Data Analysis for Biologists. Cambridge University Press (2002)
17. Saeidi, R., Mohammadi, H., Ganchev, T., Rodman, R.: Particle swarm optimization for sorted adapted Gaussian mixture models. IEEE Transactions on Audio, Speech, and Language Processing 17(2), 344–353 (2009)
18. Seidel, W., Mosler, K., Alker, M.: A cautionary note on likelihood ratio tests in mixture models. Annals of the Institute of Statistical Mathematics 52(3), 481–487 (2000)
19. Woodward, W.A., Parr, W.C., Schucany, W.R., Lindsey, H.: A comparison of minimum distance and maximum likelihood estimation of a mixture proportion. Journal of the American Statistical Association 79(387), 590–598 (1984)

An Agent-Based Model for Simulation of Traffic Network Status

Manh Hung Nguyen[1,2], Tuong Vinh Ho[1], Manh Son Nguyen[1], Thi Hoai Phuong Phan[1], Thi Ha Phan[1], and Van Anh Trinh[1]

[1] Posts and Telecommunication Institute of Technology (PTIT), Hanoi, Vietnam
[2] IRD, UMI 209 UMMISCO, Institut de la Francophonie pour l'Informatique (IFI), Hanoi, Vietnam
{nmhufng,manhsoncntt,phuonghhs,hathiphan,vanh22}@yahoo.com, ho.tuong.vinh@auf.org

Abstract. Recently, almost proposed simulation models for traffic simulation is in one of two main categories: either micro simulation based on agent model, or macro simulation based on flow modelling. Inspire of many advantages of agent-based approach in small scale simulation, it is no longer suitable for large scale simulation because of the huge amount of processing and calculations. In order to avoid this limitation, this paper introduces an agent-based model for a large scale: instead of visualise the circulation of all individual transports, we visualise only the status of traffic network and the simulation of circulation is considered as a background process. This model is applied to the traffic network of Hanoi to analyse the hot or bottle neck points on the transportation network of the city.

Keywords: Multi-Agents System, Simulation model, Traffic network, Intelligent transportation network.

1 Introduction

Recently, there have been many researches interested in the field of transportation network simulation. Therefore, there have been many models and tools proposed. Most of them are agent-based models. In which, intelligent agent and multiagent system seem to be suitable for simulate transportation network at the micro level. Each transport is thus modelled as an intelligent agent. It could observe other transports and obstacles to change its own speed as well as direction to go to its destination as fast as possible. The transportation network therefore could be modelled as a multiagent system whose each agent has a personnel goal (its destination to go) and they have to coordinate and/or interact together in order to prevent accidents from happening.

There are many models proposed in this tendency. For instance, MATSim development team [3] is developing a framework and platform for a transportation network simulation, called MATSim. MATSim provides a toolbox to implement large-scale agent-based transport simulations. The toolbox consists of several modules which can be combined or used stand-alone: toolbox for demand-modeling, agent-based mobility-simulation (traffic flow simulation), re-planning, a controller to iteratively run simulations as well as methods to analyze the output generated by the modules. SUMO

(Simulation of Urban MObility) [2,8] is a highly portable, microscopic road traffic simulation package designed to handle large road networks. It is mainly developed by employees of the Institute of Transportation Systems at the German Aerospace Center. SUMO allows to simulate how a given traffic demand which consists of single vehicles moves through a given road network. The simulation allows to address a large set of traffic management topics. It is purely microscopic: each vehicle is modelled explicitly, has an own route, and moves individually through the network. Al-Dmour [1] developed TarffSim, a Multiagent Traffic Simulation for micro-simulation and macro-simulation of traffic. Gokulan and Srinivasan [5] have been implemented two different types of multi-agent architectures on a simulated complex urban traffic network in Singapore for adaptive intelligent signal control. Piorkowski et al. [11] are developing TraNS, an open-source simulation environment, as a necessary tool for proper evaluation of newly developed protocols for Vehicular Ad Hoc Networks (VANETs). Mengistu et al. [10] provided a framework for development and execution of parallel applications such as multi-agent based simulation (MABS) in large scale. Lotzmann [9] presented an agent-based traffic simulation approach which sees agents as individual traffic participants moving in an artificial environment.

In spite of many advantages in modelling of individual's behaviors in circulation of agent-based models, they have a limitation in the number of agents in a simulation, especially in a large scale as a transportation network. Although there are some models which are able to simulate with a big number of transports, they are not easy to apply to the traffic situation and the circulation culture of Vietnamese. In Vietnam, most of transports are motorbikes. Their drivers do not always respect the circulation laws: they could go to anywhere as long as there is enough place in ahead to go. Moreover, most of streets in Vietnam have no lanes and no one respect the rule of following only one lane on a street.

Our objective thus is to develop a simulation model and tool typically for the traffic situation and the circulation culture of Vietnamese. The model is also based on multiagent system. However, in order to get over the limitation of scale in agent-based models, we do not visualise all the agents in their instant circulations. We show only the instant status of streets to see the level of congestion in streets as well as the global status of the traffic network. All calculation of activities and attributes such as speed, travel path, time, plans of individual agent and the level of congestion on streets, etc. are done in background processes of the model.

This paper is organised as follows: Section 2 presents our agent-based model for simulation of traffic network status. Section 3 presents a case study in which we apply the proposed model to simulate the traffic network status of Hanoi. Finally, section 4 discusses the presented work and draws some perspectives for future work.

2 Propose Model

Our model is depicted in the Fig.1. The main ideas is to separate the visual level and the calculation level. This makes our model more flexible: its easy to change the input data to display. The input data thus could be either those from simulation, or those from realistic on the instant traffic network which are captured from on site cameras, if it is possible. Moreover, this separation also limits the effect of processing speed on the visualisation: we could simulate with a huge number of agents with a bit slow speed, but the results are then display as those of fast speed because the display is now independent from the calculation. This model has six main steps as the followings:

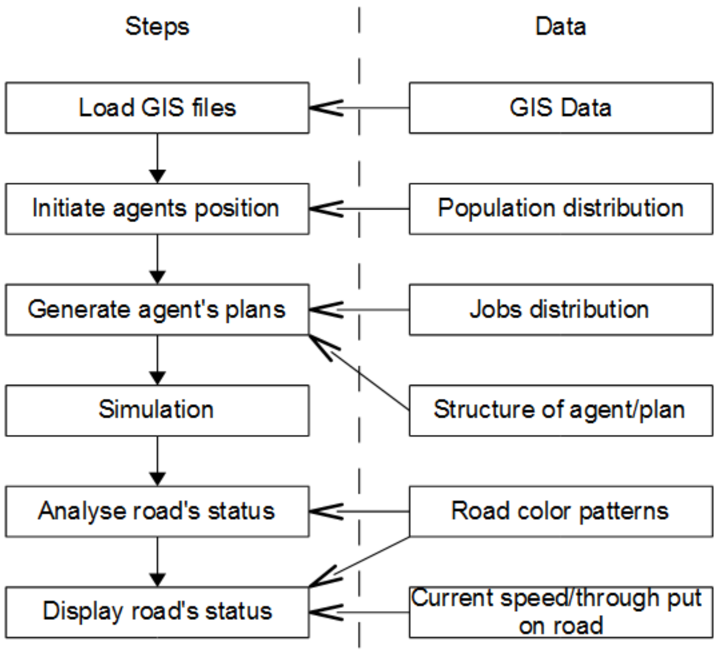


Fig. 1. Steps and processing data in the model

Step 1: Load GIS files.

This step loads the GIS (Geographic Information System) files to create the roads network. The use of GIS files enables us to work with the realistic data from the real transportation network. For each road, the GIS data contains:

– id: road identification.
– name: road name.
– direction: one way or dual ways
– permitted vehicles: kinds of vehicle can circulate on the road
– capacity: the width or through put of road
– lanes: number of lanes

Step 2: Initiate agent's position.

The second step initiates the agent's position. The position of agents is determined by a zone. Thus the number of agent in a specific zone z is determined as following:

$$n(z) = \alpha * d(z) * \frac{s(z)}{S} * N \tag{1}$$

where α is the simulation ratio regarding the real size of population; $d(z)$ is the density of population in the zone z; $s(z)$ is the surface of the zone z; S is the overall surface of simulation zone (entry city); N is the total population of the simulation city.

Step 3: Generate agent's plans.

This step generate plans for each individual agent. A plan contains followings information:

- start time: the start time to circulate
- department: the start position of the circulation. A position is represented by its real (longitude (x), latitude (y)).
- destination: the destination(s) to get to of the individual
- max speed: the maximal speed for the individual. This must not be higher than the maximal permitted speed by law.
- type of vehicle: this attribute is reserved to determine the size of vehicle on the road.

The plans are generated based on the population distribution, the structure of jobs in society, and the distribution of offices, commercial centres, schools, hospitals, etc... in the city. An individual could have many plans with different destinations and time. For instance, a student takes his motorbike to get to his university at 7am, then back home at 12pm. At 2pm he outs again with his motor to go to commercial centre to shop with a friend, and then both of them go to the friend's house at 5pm, and then he backs home at 7pm.

Step 4: Simulation.

Each individual is represented by an agent with its daily plans. At any moment, we can detect the position of agent on the road by considering two factors:

- Its circulation path: this is either statically determined by the Dijkstra's algorithm (Dijkstra [4]), or dynamically found by the improved star increment algorithm (Huang et al. [6], an improvement from Koenig et al. [7]).
- Its speed: the real speed of agent is determined by following rules:
  - Accelerate: when there is neither obstacle nor red light in front of it and its speed is not maximal yet.
  - No change: when there is some obstacles in front of it such that it could not get over, or its speed already maximal.
  - Slow down: when there are many obstacles in front of it and they are decrease their speed too.

Step 5: Analyse road's status.

The actual status of road is determined by the comparison between the current through put of road and the road's capacity, this rate is calculated in percentage:

- Blocked road: the rate is $> 95\%$
- Very slow road: the rate is between 85% and under 95%
- Slow road: the rate is between 75% and under 85%
- Normal road: the rate is under 75%

Another metric for this analysis is the average speed of transports on the road. For instance, the road is blocked if this speed is less than $1km/h$; very slow if the speed is between $1km/h$ and $5km/h$; slow if the speed is between $5km/h$ and 10km/h; normal if the speed is higher than $10km/h$.
Step 6: Display road's status.

The displayed color of road (intersection) depends on the circulation status of the road (intersection resp.):

- Red: the road (intersection) is blocked
- Orange: the road almost full, the vehicles movement is very slow
- Yellow: the road contains many transports, the movement is slow
- Gray: the road has a normal traffic, the movement is normal

3 A Case Study: Simulation of Hanoi Traffic Network Status

3.1 Simulation Setup

Initiation of Agents Population. The initial position of agents is created with the same rate of realistic population distribution of Hanoi, by districts (Table 1[1]). For instance, if we take the simulation rate is $1 : 100$, there will be about 23000 agents live in the 9 central districts. Therefore, the will be about 2200 agents live in Ha Dong district, and 2500 other agents live in Thanh Xuan district.

The destination of agents is determined based on its jobs and family situation. For instance, a student will go to his university. An officer may go directly to his office or pass over his or her son's school.

The data about the distribution of offices, hospitals, schools, universities, tourist sites, commercial centres, manufactures, etc. is stocked in a GIS file (Figure 2). This enables us to capture the realistic position of an individual's destination, and then the real travel distance for each agent on its plans.

Table 1. Population distribution by central districts

District	**Number of quarters**	**Surface (km^2)**	**Population (1000 people)**
Ba Dinh	14	9.22	235.7
Cau Giay	8	12.04	237.0
Dong Da	21	9.96	379.2
Ha Dong	17	47.91	217.7
Hai Ba Trung	20	9.60	325.6
Hoan Kiem	18	5.29	156.6
Hoang Mai	14	41.04	329.0
Tay Ho	8	24	139.2
Thanh Xuan	11	9.11	252.0

[1] Statistical numbers collected from multi sources in 2009.

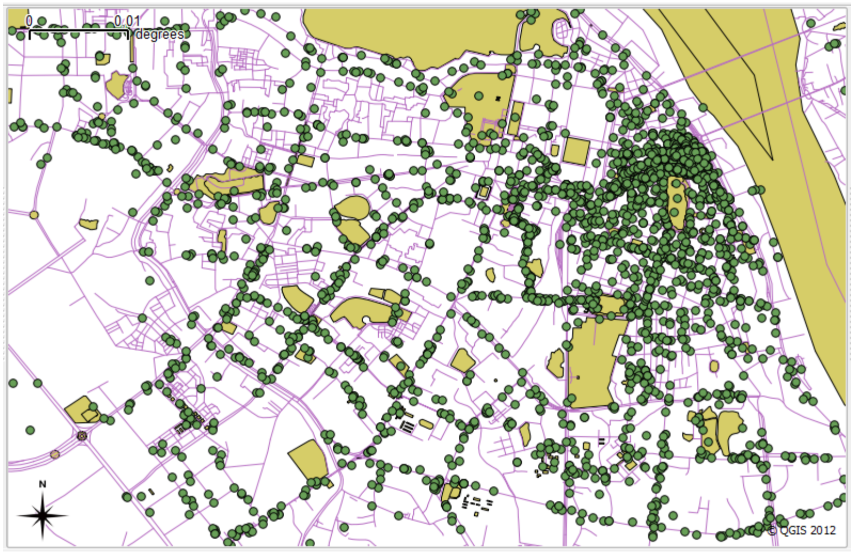


Fig. 2. Distribution of offices, hospitals, schools, tourist sites, etc. of Hanoi

```
<agent id=‘‘agentId’’ />
    <plan startTime = ‘‘06:30:00’’ maxSpeed=‘‘30km/h’’ vehicle=‘‘car’’>
        <department x=‘‘105.7870’’ y=‘‘20.9780’’/>
        <destination x=‘‘105.8449’’ y=‘‘21.0069’’/>
    </plan>
    <plan startTime = ‘‘07:00:00’’ maxSpeed=‘‘30km/h’’ vehicle=‘‘car’’>
        <department x=‘‘105.8449’’ y=‘‘21.0069’’/>
        <destination x=‘‘105.8550’’ y=‘‘21.0280’’/>
    <plan startTime = ‘‘16:30:00’’ maxSpeed=‘‘30km/h’’ vehicle=‘‘car’’>
        <department x=‘‘105.8550’’ y=‘‘21.0280’’/>
        <destination x=‘‘105.8449’’ y=‘‘21.0069’’/>
    </plan>
    <plan startTime = ‘‘17:00:00’’ maxSpeed=‘‘30km/h’’ vehicle=‘‘car’’>
        <department x=‘‘105.8449’’ y=‘‘21.0069’’/>
        <destination x=‘‘105.7870’’ y=‘‘20.9780’’/>
    </plan>
</agent>
```

Fig. 3. Representation of an individual’s plans in XML format

Construction of Agents' Plans

An agent's plan is created based on many information about the agent: its home position, its jobs will determine an office or a school. This will then determine the time and destination to move. For instance, a woman officer lives in Thanh Xuan district. She takes her car to take her son to school before 7am at Kim Lien quarter. Then, she goes to her office in Ba Trieu street before 8am. At the end of day, she leaves her office at 4h30pm and then, gets to her son's school to take him at 5pm and then, the mother and son back home together (Figure.3).

Once all agents' plans are planned for each day, we launch the simulation for the day and calculate the intensity of transportation on each street as the proposed model.

3.2 Results

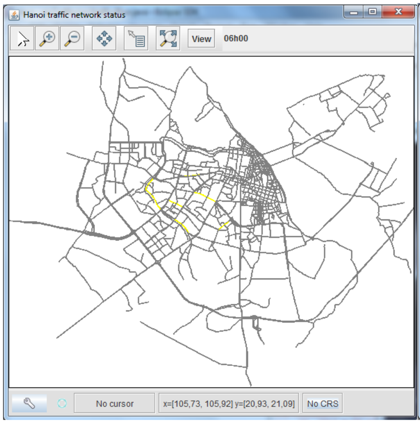
Displaying Level. The results of traffic network status are presented in the Figure 4. At 6am, there is no red or orange street, there are only some streets in yellow because it is not a rush hour yet. At 7:20am, it is really a rush hour: there are at least 5 streets in red, many in orange and yellow. Meanwhile, the number of streets in red at 6pm is lower than those at 7:20am but it is still higher than those at 12pm.

These visual representations of traffic network status enable us to track and to compare the overall status of traffic network at many daily moments. These also show whether a moment is a rush hour or not.

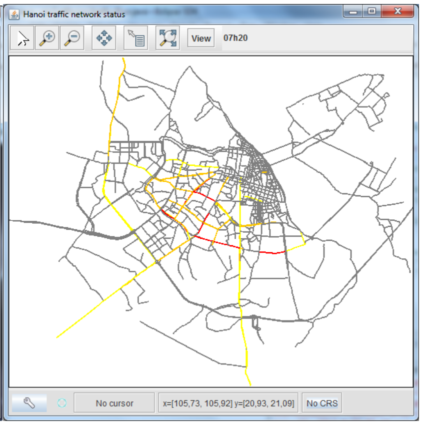
In order to represent more detail on the traffic network, our tool also enables to see in detail on any point on the network by clicking on it, there will be a small window with the detail information appears. For instance, as depicted in the Figure 5, when we click on the crossroads of st. Ton That Tung and st. Truong Chinh at the moment of 7:20am. There is a window appears to show some more detail instant information: the instant throughput of st. Truong Chinh is about 95.5% (this street is in red), while those of st. Ton That Tung is about 92.3% (this street is in orange). Moreover, other information about the involving streets is also displayed: name of the street, type of street (one way or not), etc.

Analysis Level. As data is tracked and save in a .xls file form, it is easy to analyse in any strategy or direction. For instance, we could compare the traffic status in a day between any two streets as depicted in the Figure 6.a for the st. Tran Duy Hung and the st. Pham Ngoc Thach. Therefore, there in no difference between their throughput in low traffic hours. Inversely, in rush hours, the level of congestion on the st. Pham Ngoc Thach is generally higher than those on the st. Tran Duy Hung, especially in 7-8am and 5-7pm.

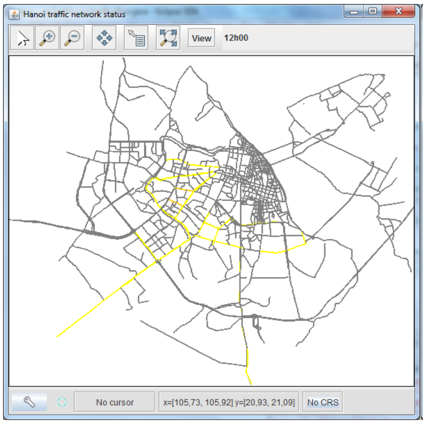
Another analysis we could take is to do on overall network, in particular, the statistic on the levels of street congestion (Figure 6.b). In this analysis, we count the number of streets in the same level of congestion and compare them together. The results indicate that the most rush hours are 7-8am and 5-6pm. Other time, there is not many street in congestion.


Hanoi traffic network status
View
06h00
No cursor
x=[105,73, 105,92] y=[20,93, 21,09]
No CRS

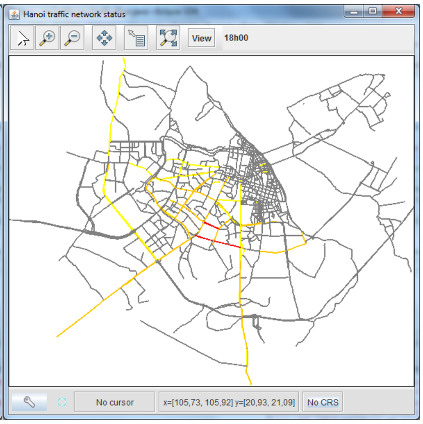
(a) 6:00 A.M


Hanoi traffic network status
View
07h20
No cursor
x=[105,73, 105,92] y=[20,93, 21,09]
No CRS

(b) 7:20 A.M

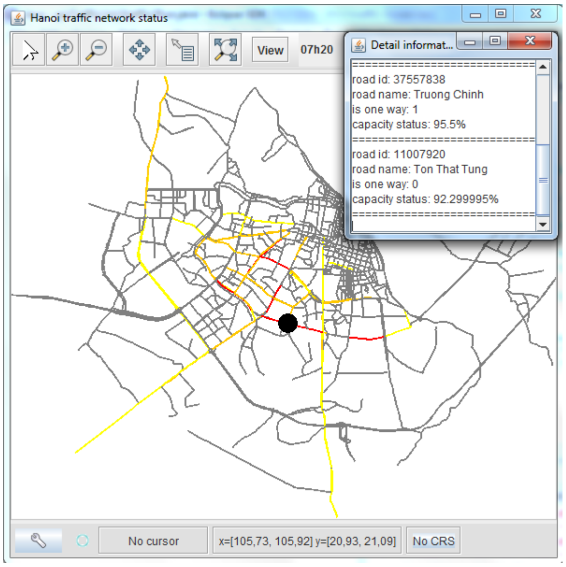

Hanoi traffic network status
View
12h00
No cursor
x=[105,73, 105,92] y=[20,93, 21,09]
No CRS

(c) 12:00 P.M


Hanoi traffic network status
View
18h00
No cursor
x=[105,73, 105,92] y=[20,93, 21,09]
No CRS

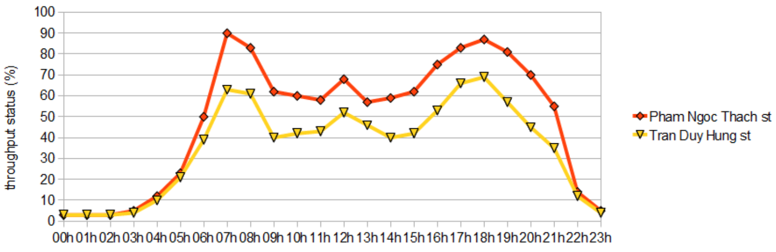
(d) 06:00 P.M

Fig. 4. Traffic status in different daily moments

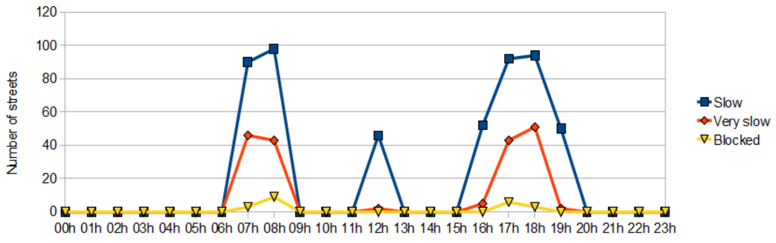

Hanoi traffic network status
View
07h20
Detail informat...
No cursor
x=[105,73, 105,92] y=[20,93, 21,09]
No CRS

```
===========================
road id: 37557838
road name: Truong Chinh
is one way: 1
capacity status: 95.5%
===========================
road id: 11007920
road name: Ton That Tung
is one way: 0
capacity status: 92.299995%
===========================
```

Fig. 5. Detail view on a point



(a) Comparing the throughput between streets



(b) Statistic on levels of street congestion

Fig. 6. Some analysing instances

4 Conclusion

This paper proposed an agent-based model for simulation the traffic network status. The model is then applied to simulate the traffic network status of Hanoi. This model enables to show the instant traffic network status at any daily moment. This also helps us to analyse the statistic on some particular street as well as those of all streets in the network. Another advantage of this model is that it could simulate the traffic network status for any city as long as we have data about the real traffic network, the population distribution, and the jobs/age distribution of the city.

Testing some real scenarios in the traffic network to find out the best scenario, simulating to optimise some routing strategies, or evaluate some new propose policies on the urban circulation are our works in the near future.

References

1. Al-Dmour, N.A.: TarffSim: Multiagent traffic simulation. European Journal of Scientific Research 53(4), 570–575 (2011)
2. Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO - simulation of urban mobility: An overview. In: The Third International Conference on Advances in System Simulation, SIMUL 2011, Barcelona, Spain, pp. 63–68 (October 2011)
3. MATSim development team (ed.). MATSIM-T: Aims,approach and implementation. Technical report, IVT, ETH Zürich, Zürich (2007)
4. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
5. Gokulan, B.P., Srinivasan, D.: Multi-agent system in urban traffic signal control. IEEE Comp. Int. Mag. 5(4), 43–51 (2010)
6. Huang, B., Wu, Q., Zhan, F.B.: A shortest path algorithm with novel heuristics for dynamic transportation networks. International Journal of Geographical Information Science 21(6), 625–644 (2007)
7. Koenig, S., Likhachev, M., Furcy, D.: Lifelong planning a*. Artif. Intell. 155(1-2), 93–146 (2004)
8. Krajzewicz, D.: Traffic Simulation with SUMO – Simulation of Urban Mobility Fundamentals of Traffic Simulation. International Series in Operations Research & Management Science, vol. 145, ch. 7, pp. 269–293. Springer, New York (2010)
9. Lotzmann, U.: TRASS: A Multi-Purpose Agent-Based Simulation Framework for Complex Traffic Simulation Applications, pp. 79–107 (2009)
10. Mengistu, D., Tröger, P., Lundberg, L., Davidsson, P.: Scalability in distributed multi-agent based simulations: The jade case. In: Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking Symposia, FGCNS 2008, vol. 5, pp. 93–99. IEEE Computer Society, Washington, DC (2008)
11. Piórkowski, M., Raya, M., Lezama Lugo, A., Papadimitratos, P., Grossglauser, M., Hubaux, J.-P.: TraNS: realistic joint traffic and network simulator for VANETs. Sigmobile Mob. Comput. Commun. Rev. 12(1), 31–33 (2008)

Self-Adaptive Particle Swarm Optimization

Adiel Ismail[1,2] and Andries P. Engelbrecht[2]

[1] Department of Computer Science, University of the Western Cape, South Africa
aismail@uwc.ac.za

[2] Department of Computer Science, University of Pretoria, South Africa
engel@cs.up.ac.za

Abstract. Particle swarm optimization (PSO) has been used to solve a wide variety of optimization problems. The basic PSO algorithm contains a number of control parameters, including the inertia weight, w, and the acceleration coefficients, c_1 and c_2. The PSO, as an optimization algorithm, is ideally suited to optimize its own parameters. This paper proposes that the control parameters of PSO be optimized in a secondary swarm where each position vector component of each particle contains a prospective PSO control parameter (i.e. w, c_1 and c_2) of the main swarm. This approach relieves the user from specifying appropriate parameters when using PSO. Application of the self-adaptive particle swarm optimizer (SAPSO) to 12 well known test functions shows that SAPSO managed to reach pre-specified values quicker than an adaptive PSO using fitness rank to update the inertia weight.

1 Introduction

PSO is a population based stochastic optimization algorithm which was originally developed by Kennedy and Eberhart [3] to model the social behavior of birds. PSO has increased in popularity amongst researchers and practitioners because of its simplicity and the fact that the derivative of the objective function is not required [7].

Although the PSO is easy to implement, the success of PSO largely depends on selecting appropriate values for its control parameters such as the inertia weight, w, and acceleration coefficients, c_1 and c_2. Parameter values which are incorrectly initialized may lead to suboptimal solutions, stagnation of the algorithm, premature convergence, slower convergence, or even divergent behaviour of the swarm [2].

Optimal control parameters are also problem dependent. A set of optimal parameters may produce good results on some problems, but poorer results on others.

In the basic PSO each particle is equipped with the same set of control parameters that are constant or static for the entire duration of the PSO process. Numerous empirical studies have determined good static values for the PSO parameters [1], [10]. Determining optimal static control parameters for the PSO is time consuming and often impractical. Preferably, a different set of control parameters should be used at different stages of the PSO search to accurately mirror the dynamic process of PSO.

Parameter values that vary during the execution of PSO were subsequently introduced by Shi and Eberhart [10]. Numerous empirical studies have reported good values for control parameters that vary during the search and which lead to improved performance of the PSO [4] and [9].

A few studies focused on optimizing the control parameters alongside the optimization of the decision variables [5],[8].

The PSO is well suited to optimize difficult non-linear problems where little or no domain knowledge is available [7]. Hence, the PSO, apart for optimizing the objective function, could also be used to optimize the control parameters.

This paper proposes the self-adaptive PSO (SAPSO), which selects a parameter based on a probability model proposed by Wang *et al* [12]. Two swarms are maintained, one comprising candidate solutions to the objective function, while the second swarm contains particles with candidate control parameters embedded in the components of its position vectors. During the search process, a set of control parameters is selected for each particle in the main swarm from the secondary swarm using Wang *et al*'s approach. The SAPSO algorithm self-adapts the PSO control parameters.

The rest of the paper is organized as follows: Section 2 provides an overview of PSO. The probability model for selecting a strategy used by Wang *et al*'s self-adaptive learning based particle swarm optimization (SLPSO) is presented in section 3. The SAPSO is presented in section 4. The experiments and their results are presented and discussed in section 5. The paper is concluded in section 6.

2 Particle Swarm Optimization

The population or swarm, in PSO terminology, consists of a number of particles, each a potential solution to the problem. Each particle has a position and a velocity that are manipulated over time. The PSO searches for an optimum solution by drawing each particle stochastically toward its personal best position found so far and toward the best position of its neighbourhood [3]. The position and velocity of the ith particle are denoted as $\mathbf{x}_i$ and $\mathbf{v}_i$, respectively. The best position visited by the ith particle is represented as $\mathbf{xpbest}_i$, and the best position in the entire swarm is referred to as $\mathbf{xGbest}$. The velocity and position of each particle in the swarm are respectively adjusted according to the following two equations,

$$\begin{aligned} v_i^d(t+1) = w \cdot v_i^d(t) + c_1 \cdot r_1^d(t) \cdot (xpbest_i^d(t) - x_i^d(t)) \\ + c_2 \cdot r_2^d(t) \cdot (xGbest^d(t) - x_i^d(t)) \end{aligned} \quad (1)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (2)$$

where $d = 1, 2, ..., D$, $i = 1, 2, ..., N$; N is the size of the swarm; D is the dimension of the solution vector and search space, w is the inertia weight that determines the influence of the previous velocity on the current velocity, with $0 \leq w < 1$; c_1 and c_2 are positive constants called the acceleration coefficients; $\mathbf{r}_1$ and $\mathbf{r}_2$ are two vectors containing random numbers, each random number uniformly distributed in $(0, 1)$; and t indicates the iteration number.

3 Adaptive Learning Particle Swarm Optimization

To select the most appropriate strategy from a set of four velocity/position update strategies Wang *et al*'s self-adaptive learning based particle swarm optimization (SLPSO) uses a probability model which self-adapts based on the past performance of a strategy. Not knowing which strategy to use initially, each strategy is assigned an equal probability which allows it an equal chance of being selected to update a particle's velocity. The basic idea is to increase the probability if a strategy leads to an improvement in performance. To achieve the latter N^{S1} decreasing weights, all totalling 1 are created, i.e. $w_i = \frac{log(N^{S1}-i+1)}{log(1)+log(2)+...+log(N^{S1})}$, $i = 1, 2, ..., N^{S1}$ where $S1$ denotes the swarm and N^{S1} the number of particles in swarm $S1$. The selection pressure of an approach based on this formula is much lower than one based on linearly increased weights (i.e. $w_i = \frac{i}{1+2+...+N^{S1}}$). Each update strategy is assigned a variable that accumulates weights associated with the particle which selected the update strategy. The particles are sorted in decreasing fitness and a weight is assigned to each particle with the largest weight assigned to the particle with the best fitness and the smallest weight assigned to the least fit particle in iteration t. Each particle's weight is then added to the accumulator associated with the update strategy which was most recently selected by the particle under consideration. For more detail about SLPSO refer to [12].

4 Self-Adaptive Particle Swarm Optimization

The approach outlined in section 3 is incorporated in SAPSO with its two PSO swarms, $S1$ and $S2$, where swarm $S1$ contains candidate solutions of the objective function which is optimized, while swarm $S2$ contains 3-dimensional particles with PSO control parameters, w, c_1 and c_2 embedded in the 1st, 2nd and 3rd position vector component, respectively. Swarm $S2$ contains parameter particles which are prospective candidates for updating swarm $S1$. To select the most appropriate set of parameters from swarm $S2$, Wang *et al*'s probability model of selecting an appropriate strategy as described in section 3 is used. The self-adaptive PSO (SAPSO) algorithm appears in Algorithm 1.

The equations in lines 34 and 38 of Algorithm 1 are used to update the probabilities of a parameter particle, where α is the learning coefficient which controls the updating proportion.

Essentially, each parameter particle in $S2$ is assigned a probability; initially, equally set to $1/N^{S2}$. An accumulator of probabilities is assigned to each parameter particle. During each iteration the particles in $S1$ are sorted in descending fitness. A weight, w_i, is calculated as proposed by Wang *et al* and assigned to a particle based on its fitness rank. For each particle i in $S1$, a weight, w_i, is then added to the accumulator of the parameter particle which was used most recently to update particle i. The better the resulting fitness of a particle in $S1$, the larger the increase in probability of the corresponding parameter particle in $S2$. During each iteration a parameter particle in $S2$ is selected for each particle in swarm $S1$ using roulette wheel selection.

In the problem coined by Van den Bergh and Engelbrecht [11] as 'Two steps forward, one step back', a particle with improved fitness replaces a particle with poorer fitness where some vector components of the latter particle are closer to the solution vector's components compared to the vector components of the improved particle, effectively resulting in the particle to move away from the optimal position with respect to these vector components. The procedure *GbestUpdate*() in line 14 of the SAPSO algorithm prevents the 'Two steps forward, one step back' problem by inserting each vector component of each particle's **xpbest** sequentially in **xGbest** and subsequently evaluating it. Any substituted vector component of **xpbest** that leads to an improvement in *Gbest* is retained in **xGbest**.

Swarm $S2$ is updated as follows. First calculate the fitness of each particle in swarm $S2$ as described below. Next, update the personal best of each particle and the global best of swarm $S2$. The personal best values and global best is subsequently used to update the velocity and position of particles in swarm $S2$. The fitness of parameter particle j in swarm $S2$ is calculated as follows. First, temporarily update the velocity and position of all particles in swarm $S1$ using the set of values for w, c_1 and c_2 extracted from particle j in swarm $S2$. The particle in swarm $S1$ whose current position yields the lowest function value, is returned as the fitness of parameter particle j. Note, the temporary velocities and positions of all particles in swarm $S1$ are discarded once the fitness has been calculated for a parameter particle.

5 Experiments and Results

The optimization test functions used in this paper and its parameters are described in this section.

5.1 Experimental Procedure

The principle goal of this paper is to investigate the performance of the proposed SAPSO which self-adapts the control parameters of the PSO. Performance is based on (a) the average best value reached by the swarm over 30 simulations and (b) the number of iterations required to reach an acceptable function value as specified in Table 1 within a maximum of 2×10^5 iterations. The performance of the SAPSO was compared with the adaptive PSO (APSO) of Panigrahi *et al* [6]. In APSO the inertia weight is updated using a particle's fitness rank, $Rank_i$, as $w_i = w_{min} + \frac{(w_{max} - w_{min}) \times Rank_i}{N^S}$. For all the experiments, the main swarm

$S1$ and swarm $S2$ consisted of 20 particles. All experiments were executed for 2×10^5 function evaluations, bearing in mind that 20 function evaluations are required per iteration by the APSO. Additional function evaluations are required by SAPSO during execution of the *GbestUpdate*() routine. All experiments were repeated 30 times. The velocity and the position of particles in swarm $S2$ were updated every 40th iteration. For APSO, parameters w_{min}, w_{max}, c_1 and c_2 were initialized to 0.4, 0.9, 2 and 2, respectively. The optimization functions are defined in Table 1. The parameters for the test functions used in the experiments are reflected in Table 1. The domain refers to the space in which the optimum will be searched for while the threshold refers to an acceptable function value to be reached within the maximum specified number of iterations.

Algorithm 1. The SAPSO algorithm

```
1:  Initialize swarm $S1$ to random positions and set velocities to zero.
2:  Initialize swarm $S2$ to random positions in ranges [0,1], [0,2] and [0,2]
    for dimensions 1,2 and 3, resp. of each particle, set velocities to zero.
3:  Initialize parameter selection probabilities to equal values,
     i.e. $probPARM_i = \frac{1}{N^{S2}}$, $i = 1, 2, ..., N^{S2}$, where $N^{S2}$ denotes swarm size
4:  Initialize SAPSO parameters,
     i.e. the learning rate, $\alpha = \frac{1}{6}$ and $Generations = 10$.
5:  Initialize weights associated with particles,
        $w_i = \frac{log(N^{S1}-i+1)}{log(1)+log(2)+...+log(N^{S1})}$,   $i = 1, 2, ..., N^{S1}$, $N^{S1}$ = swarm size
6:  Set iteration counter, $t = 0$.
7:  while stopping criteria is not satisfied do
8:    for each particle $i$ in $S1$ do
9:         Select a parameter particle, $k$, using roulette wheel selection
10:        based on its probability and store $k$ in array $particleSelected[i] = k$
           extract $w$, $c_1$ and $c_2$ and update posn. and vel. of particle $i$ in $S1$.
11:      if $f(\mathbf{x}_i^{S1}) < f(\mathbf{xpbest}_i^{S1})$ then
12:            $\mathbf{xpbest}_i^{S1} = \mathbf{x}_i^{S1}$
13:            $pbest_i^{S1} = f(\mathbf{x}_i^{S1})$
14:            Perform GbestUpdate() for particle $Gbest^{S1}$.
15:        end if
16:        if $f(\mathbf{x}_i^{S1}) < f(\mathbf{xGbest}^{S1})$ then
17:            $Gbest = f(\mathbf{x}_i^{S1})$
18:            $\mathbf{xGbest} = \mathbf{x}_i^{S1}$
19:        endif
20:    end for
21:    Sort the particles in ascending $fitness$
22     for each particle $i$ in $S1$ do
23:        Retrieve parameter particle $j$ (indexed by $particleSelected[i]$)
           that produced $fit_i$
24:        Add weight $i$ to the $j$'s accumulator, i.e. $S_j = S_j + w_i$
25:    end for
26:    if $t$ % $(4 \times Generations) == 0$
27:        Update swarm $S2$
28:        for each parameter particle $j$ in $S2$ do
29:            $probPARM_i = \frac{1}{N^{S2}}$, i.e. initialize probability of parm$_i$
30:            $S_j = 0$
31:        end for
32:    else if $(t$ % $Generations) == 0$
33:        for each parameter particle $j$ in $S2$ do
34:            $probPARM'_j = (1 - \alpha) \cdot probPARM_j + \alpha \cdot \frac{S_j}{Generations}$
35:            $S_j = 0$
36:        end for
37:        for each parameter particle $j$ in $S2$ do
38:          $probPARM_j = \frac{probPARM'_j}{probPARM'_1+probPARM'_2+...+probPARM'_{N^{S2}}}$
39:        end for
40:    end if
41:    $t = t + 1$
42. end while
```

Table 1. Definitions and parameters of test functions

Function (where $D = 30$)	Domain	Threshold	Name
$f_1(\mathbf{x}) = \sum_{i=1}^{\frac{D}{2}} 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2$	$[-10, 10]^D$	100	Rosenbrock
$f_2(\mathbf{x}) = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^D$	100	Quadric
$f_3(\mathbf{x}) = -20 \cdot \exp(-0.2 \cdot \sqrt{\frac{1}{D}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{D} \cos(2\pi x_i))$	$[-32, 32]^D$	0.01	Ackley
$f_4(\mathbf{x}) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^D$	50	Rastrigin
$f_5(\mathbf{x}) = \sum_{i=1}(\lvert x_i + 0.5 \rvert)^2$	$[-100, 100]^D$	0.0	Step
$f_6(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0.01	Sphere
$f_7(\mathbf{x}) = -\sum_{i=1}^{D} x_i \sin(\lvert x_i \rvert^{\frac{1}{2}})$	$[-500, 500]^D$	-1000	Schwefel
$f_8(\mathbf{x}) = \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i) + 10)$ and $y_i = \begin{cases} x_i & \text{if } \lvert x_i \rvert < \frac{1}{2} \\ \frac{round(2x_i)}{2} & \text{if } \lvert x_i \rvert \geq \frac{1}{2} \end{cases}$	$[-5.12, 5.12]^D$	50	Non-continuous Rastrigin
$f_9(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	0.01	Schwefel's P2.22
$f_{10}(\mathbf{x}) = \sum_{i=1}^{D} i \cdot x_i + random[0, 1)$	$[-1.28, 1.28]^D$	0.01	Quadric Noise
$f_{11}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]^D$	0.01	Griewank
$f_{12}(\mathbf{x}) = \frac{\pi}{D}(10\sin^2(\pi \cdot y_i) + \sum_{i=1}^{D-1}(y_i - 1)^2 \cdot (1 + 10\sin^2(\pi \cdot y_{i+1})) + (y_D - 1)^2) + \sum_{i=1}^{D} u(x, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x, a, k, m) = \begin{cases} k(x_i - a)^m, & \text{if } x_i > a \\ 0, & \text{if } -a \leq x_i \leq a \\ k(-x_i - a)^m, & \text{if } x_i < -a \end{cases}$	$[-50, 50]^D$	0.01	Generalized Penalized function

All the global minima are zero, excepting the Schwefel f_7 function, which has a global minimum $f_{\min} = -12569.5$.

5.2 Experimental Results

Table 2 contains the results for the APSO and the SAPSO where the threshold as reflected in Table 1 was a terminating condition subject to 2×10^5 as the maximum number of iterations allowed. Columns below header 'Success rate' indicate the percentage of simulations out of 30 that managed to reach the threshold. The average number of iterations required to reach the thresholds are also reflected in Table 2. Results indicate that SAPSO required far fewer iterations to reach the pre-specified values than APSO in 11 of the 12 functions. SAPSO could not reach the threshold of the f_{10} function in any one of the 30 simulations.

The global best values over 30 simulations are tabulated in Table 4 for both APSO and SAPSO for each of the 12 test functions. SAPSO achieved better average global best values than APSO on all functions except for function f_{10}. SAPSO showed marginal improvement in fitness over APSO in the case of the Griewank function (f_{11}).

The logarithm of the average global best value over 30 simulations for the various test functions for both APSO and SAPSO are plotted in subfigures (a) to (l) of figures 1 and 2. Plots in figures 1 and 2 indicate that SAPSO converged quicker to an optimum than the APSO for 11 of the 12 test functions. APSO converged prematurely in the case of functions f_1, f_4 and f_{12}, while SAPSO continued to reach superior solutions.

Table 2. Success rate in reaching threshold in 30 simulations for the APSO and SAPSO

Function	Success rate		Average number of function evaluations to reach threshold	
	APSO	SAPSO	APSO	SAPSO
f_1	100%	100%	21517	3396
f_2	100%	100%	62384	19380
f_3	93%	100%	79229	6336
f_4	63%	100%	54391	664
f_5	0%	100%	–	42879
f_6	100%	100%	16179	5297
f_7	67%	93%	73969	1324
f_8	100%	100%	40641	576
f_9	100%	100%	18585	6586
f_{10}	50%	0%	150060	–
f_{11}	47%	17%	16527	7420
f_{12}	73%	100%	40424	2543

Table 3. Results of Mann-Whitney U-Test

Function	Approach	N	Mean Rank	U	Z	Better Approach?
f_1	APSO	30	45	15	−6.431	
	SAPSO	30	16	885		SAPSO
f_2	APSO	30	45.5	0	−6.652	
	SAPSO	30	15.5	900		SAPSO
f_3	APSO	30	37.1	252	−2.927	
	SAPSO	30	23.9	648		SAPSO
f_4	APSO	30	45.5	0	−6.652	
	SAPSO	30	15.5	900		SAPSO
f_5	APSO	30	45.5	0	−6.652	
	SAPSO	30	15.5	900		SAPSO
f_6	APSO	30	45.5	0	−6.652	
	SAPSO	30	15.5	900		SAPSO
f_7	APSO	30	39.9	168.5	−4.161	
	SAPSO	30	21.1	731.5		SAPSO
f_8	APSO	30	37.1	251.5	−2.934	
	SAPSO	30	23.9	648.5		SAPSO
f_9	APSO	30	45.5	0	−6.652	
	SAPSO	30	15.5	900		SAPSO
f_{10}	APSO	30	15.5	900	−6.652	
	SAPSO	30	45.5	0		APSO
f_{11}	APSO	30	35.9	288	−2.395	
	SAPSO	30	25.1	612		SAPSO
f_{12}	APSO	30	45.5	0	−6.652	
	SAPSO	30	15.5	900		SAPSO

Table 4. Average global best of 30 simulations for the APSO and SAPSO

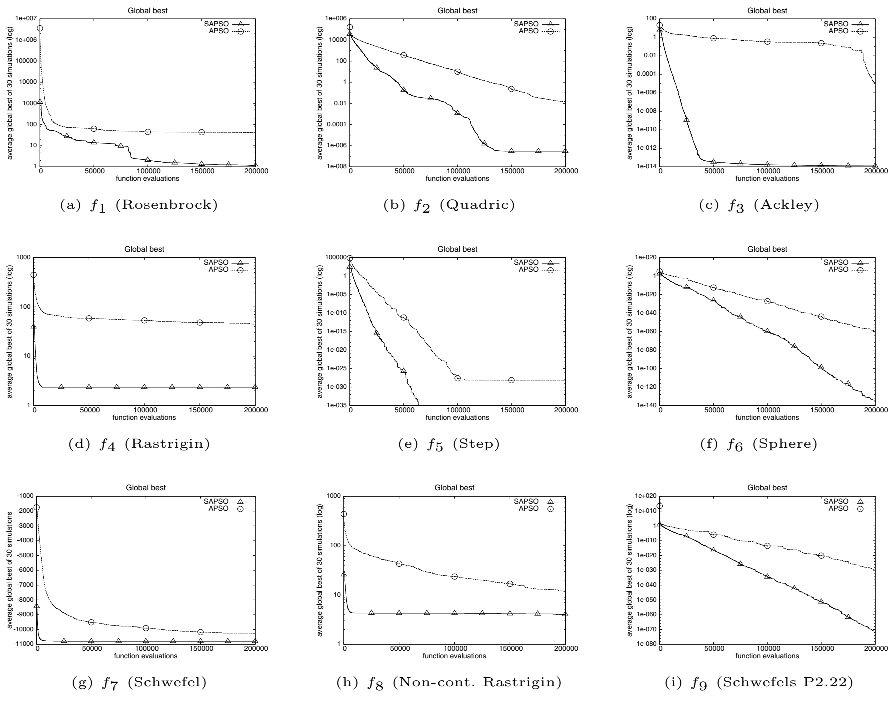
Function	APSO	SAPSO
f_1	$4.15E+01 \pm 3.14E+01$	$1.14E+00 \pm 3.11E+00$
f_2	$1.42E-02 \pm 5.89E-02$	$3.02E-07 \pm 1.65E-06$
f_3	$1.15E-05 \pm 6.21E-05$	$1.28E-14 \pm 1.14E-14$
f_4	$4.48E+01 \pm 1.46E+01$	$2.35E+00 \pm 1.42E+00$
f_5	$7.34E-29 \pm 3.30E-28$	$0.00E+00 \pm 0.00E+00$
f_6	$2.77E-60 \pm 1.49E-59$	$5.44E-137 \pm 2.98E-136$
f_7	$-1.03E+04 \pm 4.67E+02$	$-1.08E+04 \pm 4.33E+02$
f_8	$1.18E+01 \pm 1.09E+01$	$4.03E+00 \pm 1.85E+00$
f_9	$3.82E-30 \pm 1.56E-29$	$6.86E-73 \pm 3.76E-72$
f_{10}	$1.13E-02 \pm 4.33E-03$	$8.82E-02 \pm 5.01E-02$
f_{11}	$5.64E-02 \pm 4.36E-02$	$3.17E-02 \pm 3.93E-02$
f_{12}	$6.91E-02 \pm 1.31E-01$	$1.57E-32 \pm 5.57E-48$

Results of the Mann-Whitney U-Test appear in Table 3. The SAPSO algorithm performed significantly better than APSO on 11 of the 12 test functions. SAPSO also performed better than APSO on the Griewank function (f_{11}) despite the small difference in the average of the final global best values over 30 simulations as reflected in Table 4. In this case SAPSO had more simulations with lower global best values than APSO, while its average global best is fairly similar to that of APSO.

Figure 2 reflects the average values of the inertia weight, w, and acceleration coefficients, c_1 and c_2, at each iteration of PSO for all 30 simulations. For all functions a rather low average inertia weight is maintained throughout the optimization process. For all the functions the value of c_2 is greater than c_1; a trend usually exhibited by unimodal functions. The average inertia weight of all particles was quite low, oscillating between 0.2 and 0.5 in most of the test functions, while the values of the acceleration coefficients generally varied between 0.9 to 1.4. In the case of multimodal functions one would have expected a larger value for c_1 than c_2 during the initial stages of optimization. This trend is not depicted in any of the graphs. Parameter plots for functions $f_3, f_4, f_5, f_6, f_8, f_9, f_{11}$

and f_{12} exhibit similar trends; an initial large value for w which reduces quickly while c_1 and c_2 are initially equivalent with c_2 increasing quickly as PSO progresses while c_1 decreased quickly. The acceleration coefficients of functions f_1, f_2, f_7 and f_{10} show large adjustments of c_1 and c_2 compared to the remaining functions. For all other cases the acceleration coefficients appear to converge within 25% of the total number of function evaluations, which may lead to stagnation and subsequent poor performance of the SAPSO. This can be prevented by ensuring that the parameter particles in swarm 2 are re-initialized as soon as stagnation is detected. The rather low inertia weight also indicates that the acceleration coefficients were mainly tasked with exploring the search space.

It must be noted that the PSO solves a D-dimensional problem while the SAPSO solves a $D+3$ dimensional problem. The search space of SAPSO is thus larger than the search space of the APSO. The results reported in this paper are quite good bearing in mind that the same number of function evaluations are used for both search spaces. A fairer comparison between the two algorithms would have been to exclude the function evaluations which are required to calculate the fitness of a parameter particle.

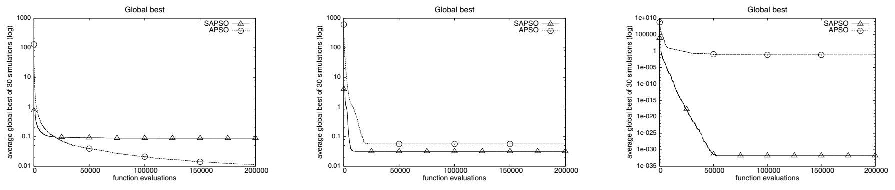


(a) f_1 (Rosenbrock) (b) f_2 (Quadric) (c) f_3 (Ackley)

(d) f_4 (Rastrigin) (e) f_5 (Step) (f) f_6 (Sphere)

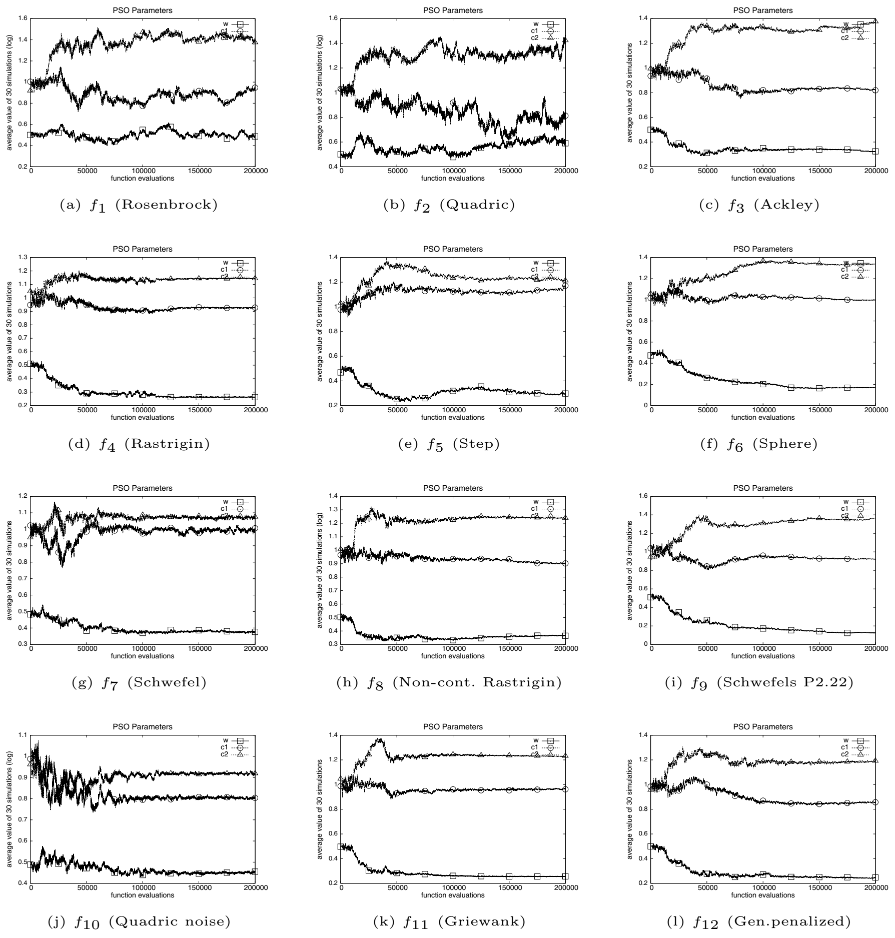
(g) f_7 (Schwefel) (h) f_8 (Non-cont. Rastrigin) (i) f_9 (Schwefels P2.22)

Fig. 1. Plots of average gbest for functions f_1 to f_9



(j) f_{10} (Quadric noise) (k) f_{11} (Griewank) (l) f_{12} (Gen.penalized)

Fig. 2. Plots of average gbest for functions f_{10} to f_{12}



(a) f_1 (Rosenbrock) (b) f_2 (Quadric) (c) f_3 (Ackley)

(d) f_4 (Rastrigin) (e) f_5 (Step) (f) f_6 (Sphere)

(g) f_7 (Schwefel) (h) f_8 (Non-cont. Rastrigin) (i) f_9 (Schwefels P2.22)

(j) f_{10} (Quadric noise) (k) f_{11} (Griewank) (l) f_{12} (Gen.penalized)

Fig. 3. Plots of average parameter values for functions f_1 to f_{12}

6 Conclusion

Determining optimal static control parameters for the PSO is not an easy task and determining it exhaustively is time consuming and often impracticable. This paper presented the self-adapting PSO (SAPSO) that automatically adapts the inertia weight and the acceleration coefficients of the PSO, while simultaneously optimizing the objective function in the main swarm. Each particle in the main swarm is equipped with its own set of control parameters which are selected from the secondary swarm using a probability model proposed by Wang *et al* [12]. SAPSO succeeded in reaching pre-specified function values much faster than the APSO in 11 of the 12 test functions.

References

1. Carlisle, A., Dozier, G.: An off-the shelf PSO. In: Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, USA (2001)
2. Clerc, M., Kennedy, J.: The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
3. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
4. Malik, R.F., Abdul Rahman, T., Mohd. Hashim, S.Z., Ngah, R.: New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight. International Journal of Computer Science and Security 1(2), 43 (2007)
5. Meissner, M., Schmuker, M., Schneider, G.: Optimized Particle Swarm Optimization (OPSO) and its application to artificial neural network training. BMC Bioinformatics 7, 125 (2006)
6. Panigrahi, B.K., Pandi, V.R., Das, S.: Adaptive particle swarm optimization approach for static and dynamic economic load dispatch. International Journal of Energy Conversion and Management 49, 1407–1415 (2008)
7. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through Particle Swarm Optimization. Natural Computing 1(2-3) (2002)
8. Parsopoulos, K.E., Vrahatis, M.N.: Parameter selection and adaptation in Unified Particle Swarm Optimization. Mathematical and Computer Modelling 46, 198–213 (2007)
9. Ratnaweera, A., Watson, H.C., Halgamuge, S.K.: Particle Swarm Optimiser with Time Varying Acceleration Coefficients. In: International Conference on Soft Computing and Intelligent Systems, pp. 240–255 (2002)
10. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, New York, pp. 591–600 (1998)
11. Van den Bergh, F., Engelbrecht, A.P.: A Cooperative Approach to Particle Swarm Optimization. IEEE Transactions on Evolutionary Computation 8(3) (June 2004)
12. Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., Tian, Q.: Self-adaptive learning based particle swarm optimization. Information Sciences 181, 4515–4538 (2011)

Evaporation Mechanisms for Particle Swarm Optimization

Juan Rada-Vilela, Mengjie Zhang, and Winston Seah

School of Computer Science and Engineering
Victoria University of Wellington
PO Box 600, Wellington New Zealand
{juan.rada-vilela,mengjie.zhang,winston.seah}@ecs.vuw.ac.nz

Abstract. This paper presents a novel approach to dealing with sample noise in Particle Swarm Optimization (PSO) by introducing a heterogeneous swarm whose particles have different evaporation factors. So far, previous works have considered only homogeneous swarms in which the evaporation factor is the same across particles. However, choosing a proper factor largely depends on the severity of noise in the optimization problem. If the level of noise cannot be determined *a priori*, arbitrarily choosing the evaporation factor can lead to rather poor results. This paper shows that heterogeneous swarms are generally better than homogeneous ones in low to medium levels of noise, and also in its absence.

1 Introduction

The benefits of incorporating an evaporation mechanism into Particle Swarm Optimization (PSO) deal with sample noise in optimization problems has been explored in previous articles [1,2,3]. Clearly, the advantages of such a mechanism are its cheap computational cost, its simple integration into particles, and its ability to mitigate the effect of sample noise.

Thus far, most articles in the topic have experimented with homogeneous swarms, that is, swarms in which all particles have the same evaporation factor. However, the results from [1] clearly show that the proper amount of evaporation depends on the severity of noise in the environment. Specifically, low evaporation factors are favorable for low levels of noise while high evaporation factors are better suited for high levels of noise. This is just fine when the severity of noise can be determined *a priori*, but when such is not the case, arbitrarily choosing an evaporation factor can deteriorate significantly the performance of the PSO algorithm. Thus, facing this scenario, we want to investigate the performance of a heterogeneous swarm in which particles have different evaporation factors in order to mitigate the disruptive effect of sample noise regardless its severity.

The overall objective of this paper is to determine whether heterogeneous swarms yield better results than homogeneous ones on large-scale optimization problems with sample noise. The PSO variant chosen for this study is the Random Asynchronous PSO (RA-PSO) [4] since it has shown to outperform other PSO variants specially when dealing with different levels of noise [1]. Specifically, we will focus on:

1. Comparing the performance of a heterogeneous swarm against homogeneous ones on large-scale optimization problems under different levels of multiplicative sample noise from a Gaussian distribution.
2. Performing statistical significance tests to identify the best-performing swarm.
3. Analyzing the difference in performance between the swarms.

The rest of this paper is structured as follows. Section 2 presents the fundamentals of PSO and RA-PSO along with sample noise in optimization and evaporation mechanisms to deal with it. Section 3 describes the experimental design to achieve the objectives proposed. Section 4 presents the results with discussions. Finally, Sections 5 and 6 present our conclusions and suggestions for further research.

2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was invented by Eberhart and Kennedy in 1995 [5] with inspiration on social models (e.g. bird flocking, fish schooling) and swarming theory. It is a population-based algorithm in which its individuals (known as particles) encode potential solutions to n-dimensional optimization problems. These particles explore the search space through cooperation with other particles by communicating the best solutions found so far and moving towards them.

Each particle has a position vector $\mathbf{x}(t)$ that encodes a potential solution to the problem, and a velocity vector $\mathbf{v}(t)$ that balances the trade-off between exploration and exploitation of the search: high velocities lead to large changes in the positions (exploration), while low velocities produce small changes (exploitation). Equations 1 and 2 determine the change in position and velocity (respectively) for particle i in dimension j at iteration $t+1$:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{1}$$

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_2(t)[\hat{y}_{ij}(t) - x_{ij}(t)] \tag{2}$$

where w is the inertia of the particle [6], c_1 and c_2 are positive acceleration coefficients that weigh the importance of their own and external experience, $r_1(t)$ and $r_2(t)$ are random values sampled from independent uniform distributions, $y_{ij}(t)$ and $\hat{y}_{ij}(t)$ are the best positions found by particle i and its neighborhood (respectively) in dimension j.

The communication between particles is based on passing messages within their neighborhoods informing about their positions and respective quality. Every time a particle receives a message with a better position than that of previously received messages, the position will be stored as its external best. Also, whenever a particle finds a better position, it will be stored as its own best position and communicated to its neighbors.

The neighborhoods are defined by the topology of the swarm which establishes all the communication links between the particles. Even when several topologies have been proposed in the literature, the one relevant to this paper is the **star** topology which makes the swarm fully connected.

2.1 Random Asynchronous PSO

The RA-PSO algorithm was proposed in [4] to model paralellism in PSO such that results are reproducible. This variant uses asynchronous communications which allows particles to consider the latest findings by the swarm before updating their positions. Moreover, particles are randomly selected to perform their operations (evaluate, communicate, and update). Thus, particles might be obsolete for not being selected at all across some iterations and others might even be selected more than once within the same iteration. These characteristics provide fast convergence due to asynchronous communications and particles being selected more than once in an iteration. More importantly, once obsolete particles are selected to operate, their new findings will provide diversity to the swarm given that these will likely be distant to those from more active particles. This variant is described in Algorithm 1.

```
while not stopping condition do
    for $i \leftarrow 1$ to $|\mathbb{S}|$ do
        Particle p ← random($\mathbb{S}$);
        p.evaluate();
        p.communicate();
        p.update();
    end
end
```

Algorithm 1. Random Asynchronous PSO (RA-PSO)

2.2 Noise in Optimization

There are three types of dynamic optimization problems [7]: **Type I** where the shape of the search space changes but not the location of the optimum, **Type II** where only the location of the optimum changes, and **Type III** where the shape of the search space changes as well as the location of the optimum. For example, consider the search space given by $f(x, t) = x^2$ at time t, then the different types of dynamic optimization problems are:

- **Type I**: if $f(x, t+1) = x^4$, then the shape changes but the global minimum remains at $x = 0$.
- **Type II**: if $f(x, t+1) = x^2 + 10$, then the shape remains the same, but the location of the global optimum is shifted by 10 units.
- **Type III**: if $f(x, t+1) = x^4 + 10$, then the shape changes as well as the location of the global minimum.

According to this classification, multiplicative sample noise belongs to type III environments because it is modeled as $f(x, t) = f(x)(1 + N(0, \sigma))$ where $N(0, \sigma)$ is a random value (in this case) sampled from a Gaussian distribution of mean zero and standard deviation σ. Notice that the severity of noise depends on the standard deviation σ, where higher values produce a more dispersed distribution which directly translates into a more severe sample noise.

2.3 Evaporation in PSO

The idea of having evaporation mechanisms in Swarm Intelligence was first proposed for Ant Colony Optimization to encourage the exploration of new solutions in the search space [8]. This idea was later exploited for PSO as a mechanism to encourage particles to track optimal solutions in dynamic optimization problems [2,3].

The evaporation in particles makes them progressively worsen the objective value of the best position found by themselves and their neighbors. Thus, as iterations pass and no better positions have been found, particles lower their expectations and hence are able to accept positions with even worse objective values than their once personal and neighborhood bests. Consequently, particles become tolerant to noise to a certain extent. The multiplicative evaporation mechanism for minimization problems is defined in Equation 3

$$f_i^*(\mathbf{y}_i(t)) = \begin{cases} f(\mathbf{y}_i(t)), & \text{if } f(\mathbf{y}_i(t)) < f_i^*(\mathbf{y}_i(t-1)), \\ f_i^*(\mathbf{y}_i(t-1)) \times (1+\rho), & \text{otherwise} \end{cases} \tag{3}$$

where $\mathbf{y}_i(t)$ and $f_i^*(\mathbf{y}_i(t))$ are the best (personal or neighborhood) position of particle i found until iteration t and its respective objective value, and $\rho \in (0,1)$ is a constant multiplicative amount which determine the evaporation rate of the objective function values of their best positions. Thus, for minimization problems, the higher the evaporation factor, the faster the particles vanish the objective value of their best positions.

2.4 Related Work

The evaporation mechanism in PSO was introduced by Cui *et al.* [3] for tracking optimal solutions in dynamic optimization problems. Their experimentation was performed on a spherical function with different levels of additive noise sampled from a Gaussian distribution, and from their results they claim that the evaporation allows the swarms to quickly converge and track optimal solutions.

Fernandez-Marquez and Arcos [2] presented a subtractive evaporation mechanism and contrasted it with the multiplicative approach proposed by Cui *et al.* [3]. They incorporated both models into a variant of PSO named multi-Quantum Swarm Optimization, and experimented on the Moving Peaks Benchmark function. They concluded that there is no significant difference between the results using either of the evaporation models.

Rada-Vilela *et al.* [1] compared the performance of both RA-PSO and PSO using homogeneous swarms and different evaporation factors on large-scale optimization functions under different levels of multiplicative noise sampled from a Gaussian distribution. Their results showed that, while the evaporation mechanism effectively mitigates the disruptive effects of noise, higher evaporation factors are needed to produce better results as the severity of noise increases. They concluded that RA-PSO is significantly better than PSO, more tolerant to noise, and better suited for the evaporation mechanism.

3 Experimental Design

3.1 Benchmarks

The benchmark functions chosen to assess the performance of the swarms are those from the CEC'2010 Special Session and Competition on Large-Scale Global Optimization [9]. The challenges of these functions to optimization techniques are its large-scale nature and the different degrees of separability. It comprises 20 minimization functions which separability ranges from separable to fully non-separable, and their objective function values are within a positive range with a global minimum at $f(\mathbf{x}) = 0$. Five sets of functions are defined in this suite as:

– $[F_{01-03}]$ separable, where each dimension can be independently optimized from the others (Set $\mathcal{A}$);
– $[F_{19-20}]$ fully non-separable, where any two dimensions cannot be optimized independently (Set $\mathcal{B}$);
– $[F_{04-08}]$ partially separable with only a single group of m dimensions that is non-separable (Set $\mathcal{C}$);
– $[F_{09-13}]$ partially separable with $\frac{d}{2m}$ groups of m dimensions that are non-separable (Set $\mathcal{D}$); and
– $[F_{14-18}]$ partially separable with $\frac{d}{m}$ groups of m dimensions that are non-separable (Set $\mathcal{E}$),

where the parameters d and m refer to the number of dimensions and the size of the groups (respectively). The default values of these parameters are $d = 1\,000$ and $m = 50$, which are also the ones used in this paper. For further details about these functions, please refer to [9].

3.2 Experimental Setup

We are interested in measuring the performance of a heterogeneous swarm whose particles have different evaporation factors equally distributed between $\rho = \{0.01, 0.05, 0.1, 0.25, 0.5\}$. These factors were arbitrarily chosen to balance the trade-off between exploitation and exploration by having from low to high evaporation factors. Its performance is assessed upon the set of large-scale optimization benchmarks previously described, each including sample noise from Gaussian distributions and respective standard deviations $\sigma = \{0.0, 0.11, 0.22, 0.33\}$, forcing the totality of the samples to lie within 3σ by resampling if needed. Thus, for $\sigma = 0.33$, it is ensured that noise will be at most 1.0 ± 0.99. Furthermore, its performance is compared with that of two homogeneous swarms with low and high evaporation factors ($\rho = 0.1$ and $\rho = 0.5$, respectively) in order to determine which approach is better on noisy optimization problems.

The swarms are made up by 50 particles with $1\,000$ dimensions each. The communication is defined by the star topology, and the acceleration and inertia coefficients are chosen according to the guidelines presented in [10]. In general, we use a simple configuration just to focus on the effects of evaporation and noise between the swarms. The parameters used are presented in Table 1.

Table 1. Parameter values

Parameter	Value
Independent runs	50×300 iterations
Number of particles	50 in $\mathbb{R}^{1000}$ with star topology
Acceleration	Static with $c_1 = c_2 = 1.49618$
Inertia	Static with $w = 0.729844$
Maximum velocity	$0.25 \cdot \lvert x_{\max} - x_{\min} \rvert$
Velocity clamping	hyperbolic tangent
Standard deviation of noise	$\sigma = \{0.00, 0.11, 0.22, 0.33\}$
Heterogeneous evaporation	$\rho_b = \{0.01, 0.05, 0.1, 0.25, 0.5\}$
Homogeneous evaporation	$\rho_a = 0.1, \rho_c = 0.5$

Experiments are performed as follows. For each σ, a set of three swarms (one heterogeneous and two homogeneous) perform 50 independent runs of 300 iterations on each benchmark function. In each run, the three swarms have the exact same initial conditions. That is, a) particles are distributed exactly the same in the search space; b) the seeds for the pseudo-random number generators r_1 and r_2 are different between particles, but the set of seeds is the same across swarms; c) the seed for the pseudo-random number generator that selects the next particle is the same across swarms; and d) particles have different seeds for the Gaussian noise generators, but the set of seeds are the same across swarms. Once all the iterations have been performed, the objective value of the best personal position of each particle is computed without noise, and the one with the best solution in the swarm is recorded. Thus, after 50 independent runs, the objective values of the 50 best solutions found are recorded and used for analysis.

In this way, the results from the swarms can be directly compared on a one-to-one basis according to the same level of noise in each independent run. More importantly, the statistical significance tests on the results can be performed using paired samples, which provides a greater confidence than using unpaired ones. The statistical significance of the results is assessed by the pairwise Wilcoxon rank sum test with Bonferroni correction at a significance level of $\alpha = 0.05$. This test is preferred because it does not assume the normality of the samples and it has shown to be helpful in analyzing the behavior of meta-heuristics [11].

4 Results and Discussions

The results from the experimentation are presented in Figure 1, and the results from the statistical significance tests on the differences in performance between the swarms are presented in Figure 2. These results show that the heterogeneous swarm (b) is significantly better than the homogeneous swarms (ac) in most of the cases. On the one hand, it can be seen that swarm (b) never produces worse results than swarm (a) regardless of the level of noise. Moreover, it produces significantly better results at least in 40% of the cases, while similar results in the remaining ones. On the other hand, swarm (b) produces significantly better results than swarm (c) in low levels of noise ($\sigma = \{0.00, 0.11\}$), but its

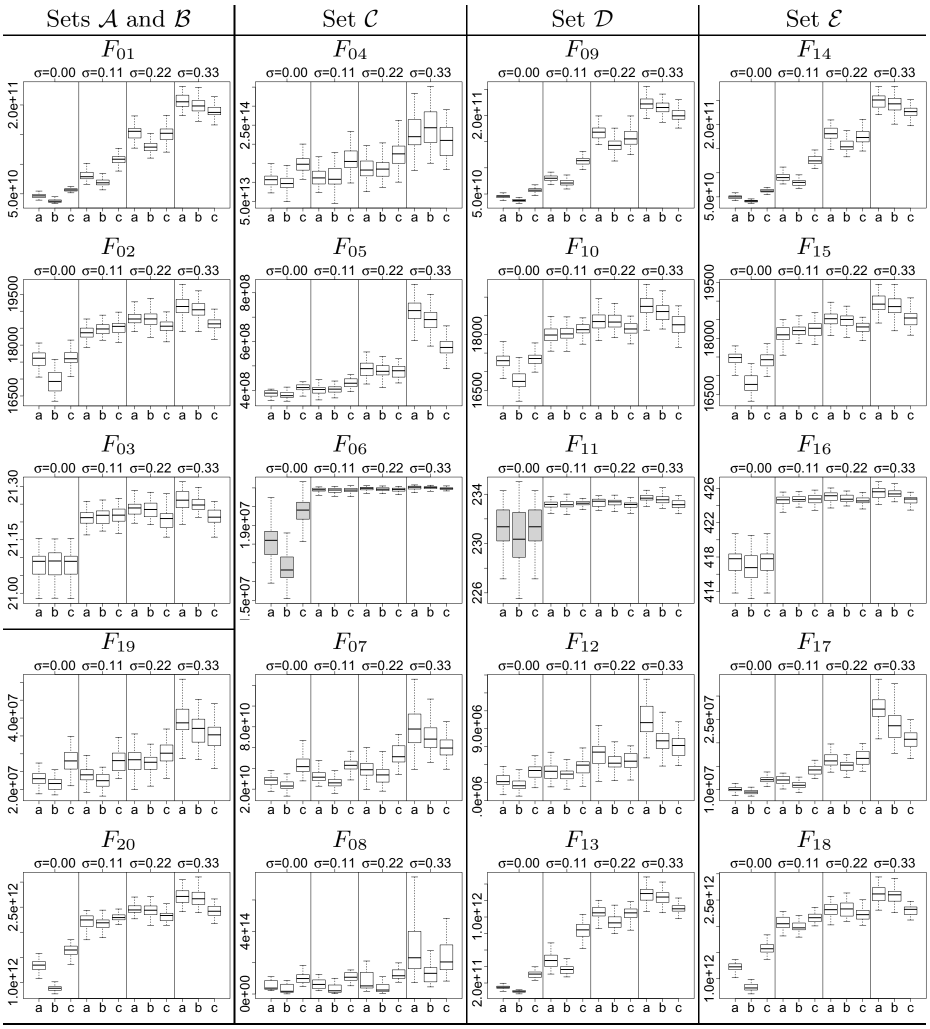


Fig. 1. The results are grouped by benchmark function and shown as boxplots representing the distribution of the objective function values of the best solutions found on each benchmark function. Notice that each subfigure is divided into sections according to the severity of noise, which is identified across the top axis in terms of the standard deviation σ. The bottom axis indicates the swarm to which the results belong: (a) homogeneous swarm $\rho = 0.1$, (b) heterogeneous swarm $\rho = \{0.01, 0.05, 0.1, 0.25, 0.5\}$, and (c) homogeneous swarm $\rho = 0.5$; while the vertical axis shows the objective function values. Also, for better visualization, the boxplots in gray indicate that the results have been scaled using the following factors: $[F_{06}, \sigma = 0.00] : \dot{a} = \dot{b} = \dot{c} = 1.7$ and $[F_{07}, \sigma = 0.00] : \dot{a} = \dot{b} = \dot{c} = 1.04$.

F01	ab	bc	ca	F02	ab	bc	ca	F03	ab	bc	ca
00	+	−	+	00	+	−	=	00	=	=	=
11	+	−	+	11	=	=	+	11	=	=	=
22	+	−	=	22	=	+	−	22	=	+	−
33	+	+	−	33	=	+	−	33	=	+	−

$\mathcal{A}$: Separable functions

F19	ab	bc	ca	F20	ab	bc	ca
00	+	−	+	00	+	−	+
11	+	−	+	11	=	−	+
22	=	−	=	22	=	+	−
33	=	=	−	33	=	+	−

$\mathcal{B}$: Non-separable functions

F04	ab	bc	ca	F05	ab	bc	ca	F06	ab	bc	ca	F07	ab	bc	ca	F08	ab	bc	ca
00	=	−	+	00	=	−	+	00	+	−	+	00	+	−	+	00	=	−	+
11	=	−	+	11	=	−	+	11	=	=	=	11	+	−	+	11	=	−	+
22	=	−	+	22	=	=	=	22	=	=	=	22	=	−	+	22	=	−	=
33	=	+	=	33	+	+	−	33	=	+	=	33	=	+	−	33	+	−	=

$\mathcal{C}$: Single-group of 50 non-separable functions

F09	ab	bc	ca	F10	ab	bc	ca	F11	ab	bc	ca	F12	ab	bc	ca	F13	ab	bc	ca
00	+	−	+	00	+	−	=	00	=	=	=	00	=	−	+	00	+	−	+
11	+	−	+	11	=	=	+	11	=	=	=	11	=	−	+	11	+	−	+
22	+	−	−	22	=	+	−	22	=	+	−	22	+	=	=	22	+	−	=
33	+	+	−	33	+	+	−	33	=	+	−	33	+	=	−	33	=	+	−

$\mathcal{D}$: 10 groups of 50 non-separable functions

F14	ab	bc	ca	F15	ab	bc	ca	F16	ab	bc	ca	F17	ab	bc	ca	F18	ab	bc	ca
00	+	−	+	00	+	−	=	00	=	=	=	00	+	−	+	00	+	−	+
11	+	−	+	11	=	=	+	11	=	=	=	11	+	−	+	11	=	−	+
22	+	−	=	22	=	+	−	22	=	=	−	22	+	−	=	22	=	+	−
33	=	+	−	33	=	+	−	33	=	+	−	33	+	+	−	33	=	+	−

$\mathcal{E}$: 20 groups of 50 non-separable functions

Summary

b	a			c		
	−	=	+	−	=	+
00	13	7	0	17	3	0
11	7	13	0	13	7	0
22	6	14	0	9	4	7
33	7	13	0	1	2	17
Total:	33	47	0	40	16	24

a	c		
	−	=	+
00	14	6	0
11	16	4	0
22	2	9	9
33	0	3	17
Total:	32	22	27

Fig. 2. This figure presents the results from the statistical significance tests from comparing the differences in performance between the swarms. Each row represents a different level of noise, the columns represent the comparison of one swarm with respect to another, and the cells in the respective intersections contain the statistical significance between the two swarms under the same level of noise. Since the benchmarks are minimization problems, the symbols '−' and '+' indicate that the performance of a swarm is significantly better or worse (respectively) than that of another, and '=' indicates that their performance is similar. Moreover, a summary of these results is also presented to compare the global performance of the heterogeneous swarm (b) with respect to its counterparts (ac) as well as that between the homogeneous swarms. Thus, the summary provides a guideline in terms of the overall performance of the swarms according to the level of noise.

performance detriments as noise becomes more severe. Nonetheless, swarm (b) still manages to produce better results in the majority of the cases except for when the level of noise is high ($\sigma = 0.33$).

The advantage of heterogeneous swarms is that the balance between exploration and exploitation is distributed across particles. That is, particles with low evaporation are able to provide exploitation since the best positions are considered for more iterations until a better one is found. Conversely, particles with high evaporation provide exploration since just in a few iterations the quality of their best solutions is diminished by more than half and any other solution can be potentially considered as their bests. Thus, such a diversity in evaporation factors allows the swarm to cope well and adapt faster in the presence of low to medium levels of noise. However, when the objective values are subject to high levels of noise, particles with low to medium evaporation factors will retain them for longer causing the swarm to be driven towards deceptive areas for more iterations.

Comparing the homogeneous swarms (ac), the performance was favorable for swarm (a) in low levels of noise ($\sigma = \{0.00, 0.11\}$) but progressively detrimented as the levels of noise became higher ($\sigma = \{0.22, 0.33\}$). In the presence of medium to high levels of noise, the swarm with higher evaporation factors (c) significantly outperformed that with lower evaporation (a). Such a performance coincides with the findings in [1], and is expected given that low evaporation when noise is high provides exploitation of solutions whose objetive value is far from the real one, whereas high evaporation when noise is low provides too much exploration without exploiting much the best solutions found.

5 Conclusions

The evaporation mechanism in PSO further controls the trade-off between exploration and exploitation. On the one hand, particles with low evaporation factors favor exploitation as their best (personal and neighborhood) positions are retained for more iterations. On the other hand, particles with high evaporation factors favor exploration since their best positions are replaced with new ones in just a few iterations. Thus, the former particles are better suited when noise is low since exploitation is more accurate, whereas the latter ones are better when noise is high since severely affected solutions are discarded faster.

This paper has presented a novel approach to implementing the evaporation mechanism into PSO by assigning different evaporation factors to the particles in the swarm. Such a heterogeneous swarm is able to outperform homogeneous ones in the presence of low to medium levels of noise without requiring to adjust the evaporation factors to match the severity of noise. Contrarily, in homogeneous swarms, their performance depends on selecting the appropriate level of evaporation according to the level of noise, thus requiring some knowledge about the severity of noise in advance. Therefore, if noise is known *a priori* to be high, a homogeneous swarm with high evaporation factors is better suited. Otherwise, the performance of the heterogeneous swarm is generally better.

6 Future Work

This research can be further extended by:

– Experimenting with higher evaporation factors in the heterogeneous swarm in order to improve its performance in the presence of high levels of noise.
– Assessing the performance in dynamic problems such as the Moving Peaks Benchmarks [12] or the Generalized Dynamic Benchmark Generator [13].
– Compare the performance of heterogeneous swarms against homogeneous ones with dynamic evaporation mechanisms.

References

1. Rada-Vilela, J., Zhang, M., Seah, W.: A performance study on the effects of noise and evaporation in particle swarm optimization. In: IEEE Congress on Evolutionary Computation, pp. 873–880 (2012)
2. Fernandez-Marquez, J.L., Arcos, J.L.: An evaporation mechanism for dynamic and noisy multimodal optimization. In: Genetic and Evolutionary Computation Conference, pp. 17–24 (2009)
3. Cui, X., Hardin, C.T., Ragade, R.K., Potok, T.E., Elmaghraby, A.S.: Tracking non-stationary optimal solution by particle swarm optimizer. In: 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 133–138 (2005)
4. Rada-Vilela, J., Zhang, M., Seah, W.: Random Asynchronous PSO. In: 5th International Conference on Automation, Robotics and Applications, pp. 220–225 (2011)
5. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
6. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: IEEE World Congress on Computational Intelligence, pp. 69–73 (1998)
7. Eberhart, R., Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. In: IEEE Congress on Evolutionary Computation, vol. 1, pp. 94–100 (2001)
8. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
9. Tang, K., Li, X., Suganthan, P.N., Yang, Z., Weise, T.: Benchmark Functions for the CEC 2010 Special Session and Competition on Large-Scale Global Optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China (2009)
10. van den Bergh, F.: An analysis of particle swarm optimizers. PhD thesis, University of Pretoria, South Africa (2002)
11. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. Journal of Heuristics 15(6), 617–644 (2009)
12. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: IEEE Congress on Evolutionary Computation, vol. 3, pp. 1875–1882 (1999)
13. Li, C., Yang, S.: A Generalized Approach to Construct Benchmark Problems for Dynamic Optimization. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 391–400. Springer, Heidelberg (2008)

The Performance and Sensitivity of the Parameters Setting on the Best-so-far ABC

Anan Banharnsakun, Booncharoen Sirinaovakul, and Tiranee Achalakul

Department of Computer Engineering
King Mongkut's University of Technology Thonburi
Bangkok, Thailand
ananb@ieee.org, boon@kmutt.ac.th, tiranee@cpe.kmutt.ac.th

Abstract. Artificial Bee Colony (ABC) is a metaheuristic technique in which a colony of artificial bees cooperates in finding good solutions in optimal search space. The algorithm is one of the Swarm Intelligence algorithms explored in recent literature. However, ABC can sometimes be a slow technique to converge. In order to improve its performance the modified version of ABC called Best-so-far ABC were proposed. The results demonstrated that the Best-so-far ABC can produce higher quality solutions with faster convergence than either the original ABC or the current state-of-the-art ABC-based algorithm. In this work, we aim to extend the performance analysis of the Best-so-far ABC algorithm by investigating the effect of each proposed modification to the overall performance as well as to present the sensitivity of the parameters setting on the algorithm.

Keywords: Best-so-far Artificial Bee Colony (Best-so-far ABC), Swarm Intelligence, Numerical Optimization, Sensitivity of the Parameters Setting.

1 Introduction

An optimization problem is a problem of finding the best solution from all feasible solutions. Most of such optimization problems are considered NP-hard; it is strongly believed that they cannot be solved to optimality within polynomial computation time. Therefore, In order to solve these problems, previous research tends to employ an approximation that finds a near-optimal solution in a reasonable amount of time rather than a method that is guaranteed to find the optimal solution in an exponential time.

Generally, to achieve the expected optimization results, a design of the optimization model should be considered on two search strategies including exploration and exploitation. Exploration and exploitation are the important mechanisms in a robust search process. While exploration process is related on the independent search for an optimal solution, exploitation uses existing knowledge to bias the search.

In recent years, many algorithms mimicking the food foraging behavior of swarms of honey bees have been developed to balance both exploration and exploitation processes. Examples in these algorithms are the Marriage in Honey Bees Optimization (MBO) [1], the Bee Algorithm (BA) [2], the Bee Colony Optimization [3], the

Virtual Bee Algorithm (VBA) [4], the Elite Bee Method [5], the Artificial Bee Colony (ABC) [6], and the Bee Swarm Optimization (BSO) [7,8]. These variant of algorithms show the effectiveness for solving many science and engineering problem domains [9-11].

Artificial bee colony (ABC) is the one which has been most widely studied on and applied to solve the real world problems [11]. Although the activities of exploitation and exploration are well balanced and help to mitigate both stagnation and premature convergence in the ordinary ABC algorithm, the convergence speed is still an issue in some situations.

To enhance the exploitation and exploration processes, The Best-so-far ABC has been proposed by Banharnsakun et al. [12]. The experimental results have demonstrated that the Best-so-far ABC is able to produce higher quality solutions with faster convergence than the original ABC and other state-of-the-art heuristic-based algorithms [12,13].

In this work, we aim to present the performance analysis of the Best-so-far ABC algorithm by investigating the effect of each modification to the overall performance as well as to present the sensitivity of the parameters setting on the algorithm.

This paper is organized as follows: Section 2 presents a brief overview of the best-so-far ABC Algorithm. Section 3 proposes the performance analysis on the Best-so-far ABC Algorithm. Section 4 present the sensitivity of the parameters setting on the Best-so-far ABC. Section 5 draws a conclusion.

2 The Ideas of the Best-so-far ABC Algorithm

To better understand the Best-so-far ABC, a brief description of three modifications of the Best-so-far ABC including the Best-so-far ABC method (BSF), the Adjustable Search Radius (ASR), and Objective-value-based Comparison Method (OBC) is presented.

2.1 The Best-so-far Method (BSF)

In the original ABC algorithm [1], each onlooker bee selects a food source based on a probability that varies according to the fitness function explored by a single employed bee. Then the new candidate solutions are generated by updating the onlooker solutions as shown in Eq. 1.

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}) \tag{1}$$

In the Eq. 1, v_{ij} is a new feasible solution that is modified from its previous solution value (x_{ij}) based on a comparison with the randomly selected position from its neighboring solution (x_{kj}). $\emptyset_{ij}$ is a random number between [-1,1] which is used to adjust the old solution to become a new solution in the next iteration. $k \in \{1,2,3..,SN\} \wedge k \neq i$ and $j \in \{1,2,3..,D\}$ are randomly chosen indexes. The difference between x_{ij} and x_{kj} is a difference of position in a particular dimension. However, changing only one dimension of the solution x_i in the original ABC results in a slow convergence rate.

In the best-so-far method, all onlooker bees use existing information from all employed bees to make a decision on a new candidate food source. Thus, the onlookers can compare information from all candidate sources and are able to select the best-so-far position. The new method used to calculate a candidate food source is shown in Eq. 2.

$$v_{id} = x_{ij} + \Phi f_b(x_{ij} - x_{bj}) \tag{2}$$

where			
	v_{id}	=	The new candidate food source for onlooker bee position i dimension *d, d=1,2,3,...D*
	x_{ij}	=	The selected food source position i in a selected dimension j
	Φ	=	A random number between -1 to 1
	f_b	=	The fitness value of the best food source so far
	x_{bj}	=	The best so far food source in selected dimension j

2.2 The Adjustable Search Radius (ASR)

Although the best-so-far method can increase the local search ability compared to the original ABC algorithm, the solution is easily entrapped in a local optimum. In order to resolve this issue, the improvement on both exploitation and exploration based on a global search ability of the scout bee has been introduced.

In the Best-so-far ABC, the scout bee will randomly generate a new food source by using Eq. 3 whenever the solution stagnates in the local optimum.

$$v_{ij} = x_{ij} + \emptyset_{ij}\left[\omega_{max} - \frac{iteration}{MCN}(\omega_{max} - \omega_{min})\right]x_{ij} \tag{3}$$

Where v_{ij} is a new feasible solution of a scout bee that is modified from the current position of an abandoned food source (x_{ij}) and $\emptyset_{ij}$ is a random number between [-1,1]. The value of ω_{max} and ω_{min} represent the maximum and minimum percentage of the position adjustment for the scout bee. The value of ω_{max} and ω_{min} are fixed to 1 and 0.2 respectively. These parameters were chosen by the experimenter. With these selected values, the adjustment of scout bee's position based on its current position will linearly decrease from 100 percent to 20 percent in each experiment round, i.e. a scout bee will utilize the exploration process in the early step and will employ the exploitation process by using existing information of solution in the later steps.

2.3 The Objective-Value-Based Comparison Method (OBC)

Basically, the comparison of the new solution and the old solution is done by the fitness value. If the fitness of the new solution is better than the fitness of the old solution, we select the new one and ignore the old solution. The fitness value can be obtained from the following Eq. 4.

$$Fitness(f(x)) = \begin{cases} \dfrac{1}{1+f(x)} & \text{if } f(x) \geq 0 \\ 1 + |f(x)| & \text{if } f(x) < 0 \end{cases} \tag{4}$$

Based on Eq. 4, we can see that when $f(x)$ is larger than the zero but has a very small value, e.g. 1E-20, the fitness value of equation $\frac{1}{1+1E-20}$ is rounded up to be 1 (1E-20 is ignored). This will lead the fitness of all solutions to become equal to 1 in the later iterations. In other words, there is no difference between the fitness values that is equal to $\frac{1}{1+1E-20}$ and $\frac{1}{1+1E-120}$. Thus, a new solution that gives a better fitness value than the old solution will be ignored and the solution will stagnate at the old solution. In order to solve this issue, the objective value of function is directly used to compare and to select between the old solution and the new solution in each iteration.

3 The Performance Analysis on the Best-so-far ABC Algorithm

To investigate the performance on three methods of the Best-so-far ABC, numerical benchmark functions as shown in Table 1 were used in this experiment.

Table 1. Numerical benchmark functions

Function Name	Function	Ranges
Sphere	$f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2$	$-100 \leq x_i \leq 100$
Griewank	$f_2(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{D}(x_i^2)\right) - \left(\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$	$-600 \leq x_i \leq 600$
Rastrigin	$f_3(\vec{x}) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$-5.12 \leq x_i \leq 5.12$
Rosenbrock	$f_4(\vec{x}) = \sum_{i=1}^{D} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2$	$-30 \leq x_i \leq 30$
Ackley	$f_5(\vec{x}) = 20 + e - 20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}$	$-30 \leq x_i \leq 30$
Schaffer	$f_6(\vec{x}) = 0.5 + \frac{(\sin\sqrt{x_1^2 + x_2^2})^2 - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$	$-100 \leq x_i \leq 100$

The objective of the search process is to find the solutions that can produce the minimum output value from these benchmark functions. The number of employed and onlooker bees were set to 50. The value of limit and the maximum numbers of iterations were set to 50 and 1000 respectively. Each of the experiments was repeated 20 times with different random seeds. All the experiments in this paper were run on the same hardware (Intel Core 2 Quad with 2.4 GHz CPU and 4 GB memory).

3.1 The Effect of the Best-so-far Method

Fig. 1 illustrates the effect of our Best-so-far method (BSF) on convergence speed. The plot shows that the solution obtained from ABC with Best-so-far method can quickly converge to the best solution found so far (in each iteration). Onlooker bees exploit the best-so-far solution provided by employed bees to bias their search direction. Consequently, when the number of iterations is increased, the solution quality of ABC with Best-so-far method is improved quickly.

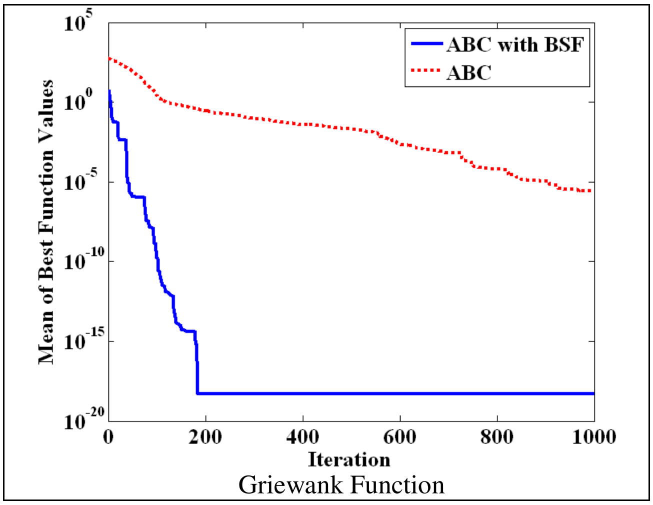


Fig. 1. Iterations to convergence for ABC and ABC with BSF

3.2 The Effect of the Adjustable Search Radius

The effect of adjustable search radius (ASR) is illustrated in Fig. 2. The result shows that the exploitation process introduced to a scout bee can improve solution quality of the ABC algorithm. The scout bee with adjustable search radius ability can also use existing information to derive a better solution when compared to a scout bee in original ABC that has only exploration ability.

3.3 The Effect of Objective-Value-Based Comparison Method

From Fig. 3, it can be seen that our objective-value-based comparison method (OBC) can help ABC to avoid an issue of solution stagnation. In original ABC, the solution can sometimes be hard to improve because the fitness values of the old and the new solution are too similar. With the objective-value-based comparison method, the new solution that gives the same fitness value but provides a better objective value will be selected and thus improve the overall solution quality.

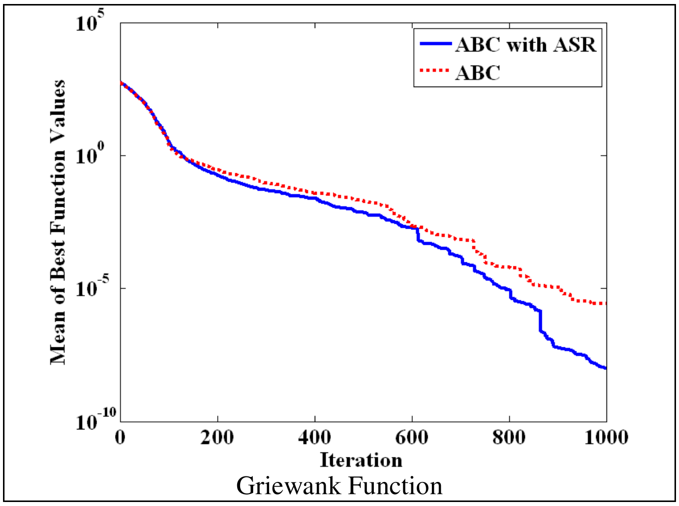


Fig. 2. Iterations to convergence for ABC and ABC with ASR

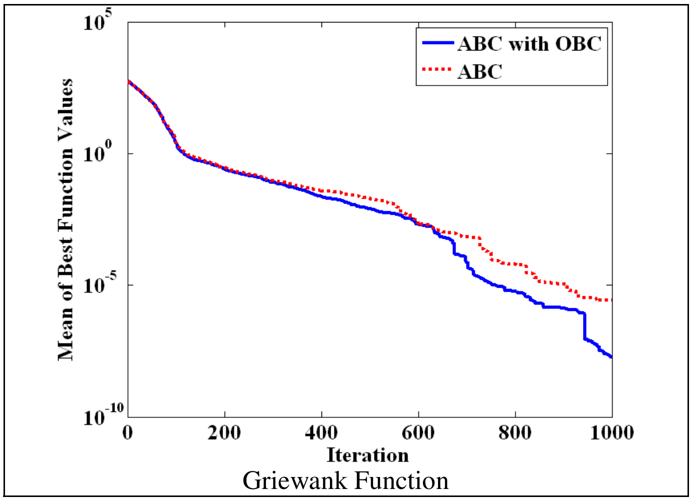


Fig. 3. Iterations to convergence for ABC and ABC with OBC

The summary of mean and standard deviation of the output values of benchmark functions obtained from these 3 modifications were recorded and shown in Table 2.

The results from these experiments illustrate that the Best-so-far method (BSF) helps onlooker bees to bias their search direction to the best solution found so far in each iteration of every test case. The improvement is thus obvious in most experiments. On the other hand, the benefit of adjustable search radius (ASR) and objective-based comparison method (OBC) can only be observed when the solution is stagnated in a local optimum. Consequently, we can conclude that the Best-so-far method (BSF) most affects the performance and solution quality of the algorithm.

Table 2. Mean and S.D. of ouput values from ABC with different modification methods

Fn	D	ABC		ABC with BSF		ABC with ASR		ABC with OBC	
		Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
f_1	30	3.32E-9	2.45E-9	2.33E-11	6.49E-11	4.04E-11	1.66E-10	2.76E-9	1.83E-9
f_2	30	2.63E-6	1.10E-5	5.17E-19	6.00E-19	9.90E-9	2.03E-8	1.79E-8	3.77E-8
f_3	30	0.04977	0.22247	3.55E-17	2.58E-17	1.39E-6	6.17E-7	0.00014	0.00053
f_4	30	4.43710	3.20420	0.01698	0.01801	1.69539	1.36531	2.71635	1.93521
f_5	30	8.28E-6	2.95E-6	0	0	6.89E-6	1.76E-6	6.11E-6	2.06E-6
f_6	2	1.44E-5	3.92E-5	5.26E-17	3.91E-17	4.76E-17	3.26E-17	1.19E-6	1.76E-6

***Fn**, Function; **D**, dimension of problems*

4 Sensitivity of the Parameters Setting on the Best-so-far ABC

In order to analyze the sensitivity of the parameters setting of Best-so-far ABC, we also use the same methodology proposed by [14]. In this methodology, we experiment with different population sizes and limit values. The population size refers to the colony size. A limit value is a parameter used to control scout production frequency. If a limit value is small, the scout production frequency is high.

4.1 Variation in Colony Sizes

In Table 3, the mean of the best function values with the colony sizes of 20, 40, and 100 are presented. The experimental results show the evidence that as the colony size increases, the Best-so-far ABC algorithm produces better results. The reason for the improvement is the fact that the diversity of the feasible solutions in the search space is increased with the population size. However, the improvement is not observed indefinitely (shown in Ackley function). In other words, there is a balanced point of population size for each problem configuration and an increment beyond that point can no longer improve the performance.

Table 3. The Mean of the best function values obtained with 1000 iterations using different colony sizes

Fn	D	Colony Size					
		20 (n_e=n_o=10)		40 (n_e=n_o=20)		100 (n_e=n_o=50)	
		Mean	S.D.	Mean	S.D.	Mean	S.D.
f_1	50	1.28E-40	5.74E-40	1.17 E-90	4.24 E-90	3.69E-221	2.76E-220
f_2	50	4.46E-25	1.99E-24	1.44E-120	6.40E-120	4.65E-275	3.54E-274
f_3	50	1.53E-28	6.84E-28	3.67E-93	1.64E-92	8.65E-272	2.05E-271
f_4	50	1.16514	2.24618	0.418848	1.07602	0.0757589	0.160013
f_5	50	2.98E-7	1.33E-6	0	0	0	0
f_6	2	2.00E-179	4.23E-178	0	0	0	0

***Fn**, Function; **D,** number of dimensions; $\boldsymbol{n_e}$, number of employed bees; $\boldsymbol{n_o}$, number of onlooker bees*

4.2 Variation in Limit Values

The “limit” values of the scout production can be defined with two variables; n_e is the number of employed bees, and D is the number of problem dimension. We analyze the effect of the scout production process on the performance of Best-so-far ABC with different “limit” values: 0.1 x n_e x D, 0.5 x n_e x D, and 1.0 x n_e x D. We also investigate these limit values on different colony sizes (20 and 100).

Table 4. The Mean of the best function values obtained with using different limit sizes

Fn	D		Colony Size					
			20 (n_e=n_o=10)			100 (n_e=n_o=50)		
			Limit = 0.1xn_exD	Limit = 0.5xn_exD	Limit = 1.0xn_exD	Limit = 0.1xn_exD	Limit = 0.5xn_exD	Limit = 1.0xn_exD
f_1	50	**Mean**	2.10E-217	1.19E-180	1.07E-57	0	0	0
		S.D	3.12E-216	2.47E-179	4.77E-57	0	0	0
f_2	50	**Mean**	0	3.30E-266	1.66E-224	0	0	0
		S.D	0	4.26E-265	2.12E-223	0	0	0
f_3	50	**Mean**	0	4.88E-121	2.43E-99	0	0	0
		S.D	0	2.18E-120	1.09E-98	0	0	0
f_4	50	**Mean**	0.79444	0.01888	0.00048	5.32E-6	1.22 E-29	0
		S.D	1.68993	0.03532	0.00130	1.47E-5	5.46E-29	0
f_5	50	**Mean**	0	0	0	0	0	0
		S.D	0	0	0	0	0	0
f_6	2	**Mean**	0	0	0	0	0	0
		S.D	0	0	0	0	0	0

Fn, Function; D, number of dimensions; n_e, number of employed bees; n_o, number of onlooker bees

Note that in this experiment, for all benchmark functions with exceptions of Schaffer function, the Maximum Cycle Number (MCN) was 10000. For Schaffer, the MCN was set to 2000.

Table 4 shows that the scout production limit affects the performance of Best-so-far ABC for small colony sizes. The solution quality and convergence speed for all benchmark functions except the Rosenbrock function (f_4) are improved when the scout production limit is high. However, the effect of the “limit” becomes much smaller when a large colony size is used. In other words, higher production limit only gives slightly better solutions in a case of a large colony size. It can be concluded that the diversity of the feasible solutions is sufficiently provided by the large population size. Thus, the diversity controlled by the scout production will give smaller effect to the algorithm in this case.

5 Conclusions

In this paper, the performance analysis of the Best-so-far ABC algorithm by investigating the effect of three modifications including best-so-far method (BSF), the

Adjustable Search Radius (ASR), and the objective-value-based comparison method (OBC) to the overall performance and the sensitivity of the parameters setting on the algorithm were proposed. The results show that the improvement by using the best-so-far-method (BSF) is thus obvious in most experiments, while the benefit of adjustable search radius (ASR) and objective-based comparison method (OBC) can only be observed when the solution is stagnated in a local optimum. The results also show that two parameters including colony sizes and limit values have the effect on the performance of the algorithm. Both values are used to control the diversity of the feasible solutions in the search space. A balanced point of colony sizes and limit values for each problem configuration will give the optimal result. In the future work, the use of nonparametric tests based on a statistical analysis will be considered and the more test cases will be addressed in order to give the confidence on the results provided.

Acknowledgements. This work is supported by the Thailand Research Fund and King Mongkut's University of Technology Thonburi through the Royal Golden Jubilee Ph.D. Program under Grant No. PHD/0038/2552.

References

1. Abbass, H.A.: Marriage in honey-bee optimization (MBO): a haplometrosis polygynous swarming approach. In: The Congress on Evolutionary Computation (CEC 2001), pp. 207–214 (2001)
2. 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 2nd International Conference on Intelligent Production Machines and Systems (IPROMS 2006), pp. 454–459. Elsevier, Oxford (2006)
3. Teodorovíc, D., Dell'Orco, M.: Bee colony optimization - a cooperative learning approach to complex transportation problems. In: Proceedings of the 10th Meeting of the EURO Working Group on Transportation, pp. 51–60 (2005)
4. 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)
5. Sundareswaran, K., Sreedevi, V.T.: Development of Novel Optimization Procedure Based on Honey Bee Foraging Behavior. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pp. 1220–1225 (2008)
6. Karaboga, D.: An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey (2005)
7. Drias, H., Sadeg, S., Yahi, S.: Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 318–325. Springer, Heidelberg (2005)
8. Akbari, R., Mohammadi, A., Ziarati, K.: A Novel Bee Swarm Optimization Algorithm for Numerical Function Optimization. Communications in Nonlinear Science and Number Simulation 15, 3142–3155 (2010)
9. Karaboga, D., Akay, B.: A survey: algorithms simulating bee swarm intelligence. Artificial Intelligence Review 31, 61–85 (2009)

10. Ziarati, K., Akbari, R., Zeighami, V.: On the performance of bee algorithms for resource-constrained project scheduling problem. Applied Soft Computing 11, 3720–3733 (2011)
11. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.: A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial Intelligence Review, doi: 10.1007/s10462-012-9328-0
12. Banharnsakun, A., Achalakul, T., Sirinaovakul, B.: The Best-so-far Selection in Artificial Bee Colony Algorithm. Applied Soft Computing 11, 2888–2901 (2011)
13. Banharnsakun, A., Sirinaovakul, B., Achalakul, T.: Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence 25, 583–593 (2012)
14. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (ABC) algorithm. Applied Soft Computing 8, 687–697 (2008)

FAME, Soft Flock Formation Control for Collective Behavior Studies and Rapid Games Development

Choon Sing Ho, Yew-Soon Ong, Xianshun Chen, and Ah-Hwee Tan

School of Computer Engineering, Nanyang Technological University,
50 Nanyang Avenue, Singapore 639798
{csho,asysong,chen0469,asahtan}@ntu.edu.sg

Abstract. We present FAME, a comprehensive C# software library package providing soft formation control for large flocks of agents. While many existing available libraries provide means to create flocks of agent equipped with simple steering behavior, none so far, to the best of our knowledge, provides an easy and hassle free approach to control the formation of the flock. Here, besides the basic flocking mechanisms, FAME provides an extensive range of advanced features that gives enhanced soft formation control over multiple flocks. These soft formation features include defining flocks in any user-defined formation, automated self-organizing agent within formation, manipulating formation shape at real-time and bending the formation shape naturally along the curvature of the path. FAME thus not only supports the research studies of collective intelligence and behaviors, it is useful for rapid development of digital games. Particularly, the development cost and time pertaining to the creation of multi-agent group formation can be significantly reduced.

Keywords: soft formation, flock, behavioral animation, collective behavior, games.

1 Introduction

Advancements in the field of digital games have evolved significantly in the recent decades and becoming ever more complex in game content. In early generation of digital games, due to the limitation of the computer systems, game contents and mechanics are kept relatively simple. As computer systems are packed with more memory and computational power, games development studios are trending towards creating new generation of digital games with substantially rich digital contents, while some bearing close resemblances to the real world. Citing the recent popular Grand Thief Auto IV (GTA) digital game, for example, which showcases an open world adventure game taking place in a fictional city with over 2000 buildings that is designed based on modern day New York. Thus, modeling of natural and social phenomena including human crowd behaviors and the flocking behaviors within groups of animals is fundamental to instill the traits of vibrancy and realism in the virtual city.

While several libraries and source codes pertaining to flock behavioral modeling are readily available on the web, it is noted that most provide only means to achieve simple steering behavior. Although the steering mechanisms provide a good underlying building block to achieve natural and realistic motion, these libraries however, do not give high level control over the formation of the group. In the real world, we can easily observe that some animals tend to flock together in some shape or formation, for a common good – to survive. Hence, a software library equipped with comprehensive set of tools to allow easy creation of group behavioral animation with flexible controls over the formation shape of the flock would be helpful for collective behavior studies and rapid development of digital games. We envisioned that this formation constraint should be a soft one, s.t.:

1. *Soft formation constraint*: the agent should be able to temporarily deviate from their formation when encountering anomalies, such as obstacles, steep terrains, and other agents, and return to its formation when the path is clear.
2. *Self-organizing agents*: the agent should not be fixed to its assigned position in the formation, but able to self-organize to fill up the formation quickly and naturally.
3. *Real-time formation control*: the formation should be able to change or molded according to users preference during runtime.
4. *Flexible flock formation*: when the group is navigating in the environment, the formation should bend naturally along with the curvature of the path.

In this paper, we present Flocking Animation Modeling Environment denoted in short as FAME, a soft flock formation library which provides extensive controls over the flock formation. One significant advantage is that development cost and time pertaining to creation of group behavioral animations can be significantly reduced. In what follows, we will give an overview of some related work. Next we will show an overview of the system architecture deployed in FAME as well as some of the key implementation details. Subsequently, we present a successful game implemented using FAME, followed by brief concluding statements.

2 Related Works

The study on collective intelligence and cooperative behaviours among agents is an ongoing and interesting area of research [1,2]. A popular approach to simulate large crowds of animated characters is through the use of behavioral models. A behavioral model involves receiving stimuli from the environment and provides an appropriate response [3]. In the literature, three core classes of behavioral models can be identified to date to create group movement animation. They are the particle system, the flocking system, and the behavioral system.

The particle system is a fast, physics based and computationally cheap approach to simulate the movement of large number of agents. Agents controlled by the particle system approach respond to a global force that defines the direction of movement while avoiding collisions. Some studies based on the particle system are listed as follows. Hughes introduces a continuum model which bears

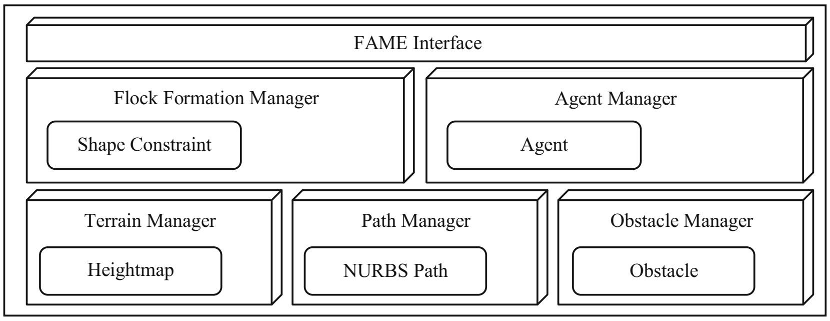


Fig. 1. FAME system architecture

close similarities to classical fluid dynamics to study on the flow of a crowd of pedestrians [4]. Chenney introduced "flow tiles" [5], a tiled based method that defines the motion of crowd for each segment that makes up the entire map.

Flocking system, on the other hand, is composed of several simple heuristics rules to simulate the natural and well-coordinated group movement behaviors such as flock of birds, a school of fish and a horde of animals in the virtual computer environment. First published by Craig Reynolds in 1987, Reynolds demonstrated a "Boids" computer simulation system [6] which simulates flock behavior using three simple rules, namely cohesion: to move towards the averaged position of all nearby agents, alignment: to steer and move in the same direction with nearby agents, and separation: to move away from nearby agent that are too close in proximity to avoid collision. This simple computer simulation has thereafter spin-off a vast array of useful real world applications including artificial life simulation, crowd simulation, military simulation, robotics, movie productions and computer games.

Behavioral system controls the agents using the rule based approach. As such, agents using the behavioral system approach are often seen as being smarter than the flocking system. An interesting research in this area is the simulation of artificial fishes by Tu et al. [7]. The research group created a virtual marine world simulation, and realistically emulated the appearance, movement and behaviour of the fishes. The behavior of the fish is driven based on the states of the fish and its intention.

Recent research interest in this field of study has been seeking for more efficient ways to achieve more realistic rendering through improved rendering techniques [8]; refining navigation algorithms to create more natural movement of autonomous agents [9,10,11,12]; increasing the number of agents that the system can handle in real time through the use of dedicated hardware, specifically the GPUs [13,14,15]; and by reducing computational cost through optimization of the data structure and efficient management of system resources [16,17].

With the maturing of work in the field, there is an increasing trend to develop toolkits to speed up development process. Recently, Shawn et al. developed a set of tools, library and test cases to ease development effort as well as to measure steering performance of individual agents [18,19]. Similarly, Erra et al. [14] developed BehaveRT, a GPU-based library to visualize large scales of autonomous agents. However, to the best of our knowledge, none of these libraries focus on formation control in groups of autonomous agents.

3 FAME API and Architecture

In this section, the system architecture of FAME is depicted in Figure 1 and composes the following core modules: *FAMEInterface*, *FlockFormationManager*, *AgentManager*, *ObstacleManager*, *PathManager* and *TerrainManager*.

FAMEInterface: This is the point of communication between the game application and FAME. It facilitates the creation and removal of flock agents. Upon initialization, FAME computes the new positions and orientations of each the agent in the flock, for given frame rate of interest, which is a user-defined parameter.

FlockFormationManager: The flock formation manager manages the creation and removal of flock groups with the desired formations. Here in FAME, each group of agents is bounded by a polygonal shape constraint. Each shape constraint is identified by a unique index upon creation and stored within a hash. This is to facilitate fast retrieval of flock object when a query is performed.

AgentManager: The agent manager handles agents at the individual level. Similar to the flock formation manager, the agent manager assigns a unique index to each agent and is stored in a hash table upon creation to facilitate fast retrieval of agent object upon query. Each agent is assigned to a flock group at one time. Within each flock group, the agent has a uniquely assigned destination location in the map, which accords to the desired formation defined.

ObstacleManager: The obstacle manager handles the list of obstacles in the scene at real time. Obstacles in FAME can be defined using a circular object or polygonal representation. Upon creation of the obstacles, the obstacle object is stored using a grid data-structure so as to reduce the real time requirements in computational effort when performing queries on them.

PathManager: The path manager handles the list of paths planned within the game environment. In FAME, a path is defined by a set of control points modeled using Non Uniform Rational B-Spline (NURBS).

TerrainManager: The terrain manager handles the terrain related properties in a game, such as the terrain size and surface elevation data. These information is used to the control movement of the agents, such as the maximum gradient that the agent is allowed to climb, and to adjust the movement speed accordingly when the agent travels uphill or downslope.

4 Soft Formation Features

In this section, we present several soft formation mechanisms provided in FAME for simulating large groups of agent flocking in a natural and believable manner, while maintaining the desired formation. These mechanisms provided includes:

1)*defining flock of irregular formation*, 2)*self-organized agent within formation*, 3)*formation shape morphing*, 4)*flexible formation path following* and 5)*seamless space partitioning mechanism*. Implementation details are as follows:

4.1 Defining Flock of Irregular Formation

FAME supports creation of flock formations that are formulated in the form of soft polygonal shape constraints. Thus agents belonging to a particular group shall stay within the defined polygon when possible. The shape constraint is populated with agents by first performing a uniform sampling, to generate the same number of sample points on the polygon. The sample points are then assigned as destination position P^{Dest} to each members of the flock. After which, the guiding steering force will attract the agents towards their individually assigned position.

Some properties of this shape constraint include a set of control points defining the shape of the flock, a list of sample points defining the agents' position in the formation, the flock centroid and orientation. This facilitates a fast and easy transformation control (translation, rotation, scale) over the flock formation in the virtual environment.

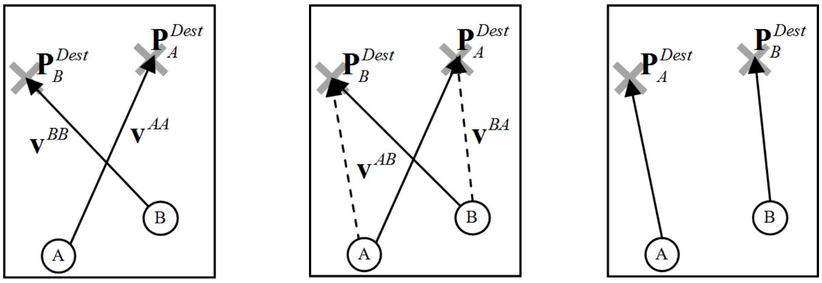
- *Translation*: The group can move around by redefining the flock centroid position.
- *Rotation*: The group can reorientate to face a particular direction by rotating the control points as well as the sample points.
- *Scale*: The group can expand or shrink the size of the formation by scaling the control points as well as the sample points.

4.2 Self-organized Agents Within Formation

It is natural that while the agents are forming the formation, those who arrived first should move inwards to fill up the front spaces while those whom arrive later will fill up the rear. Here, FAME provides such feature to automatically reorganize the agents in the formation during runtime. As the agents navigate in the virtual environment, some might encounter anomalies, such as obstacles, steep terrain and other agents of different flock groups. Thus, these agents would have to make a detour and take a longer time to move back to the formation. Other agents that manages to arrive at the formation first would move inwards, allowing agents that arrive later to fill up the formation from the rear. Implementation detail is described as follows: Given an agent A and its randomly picked neighbour agent B, let $\mathbf{P_A}$ and $\mathbf{P_B}$ be the initial position of agent A and B, respectively, while $\mathbf{P_A^{Dest}}$ and $\mathbf{P_B^{Dest}}$ denote the final destinations of agent A and B, respectively. First the following direction vectors to destination are calculated,

$$\mathbf{v^{AA}} = \mathbf{P_A^{Dest}} - \mathbf{P_A} \tag{1}$$

$$\mathbf{v^{BB}} = \mathbf{P_B^{Dest}} - \mathbf{P_B} \tag{2}$$



a) Calculate V^{AA} and V^{BB} b) Calculate V^{AB} and V^{BA} c) Swap if criterion is met

Fig. 2. Criterion for two agent to swap destination

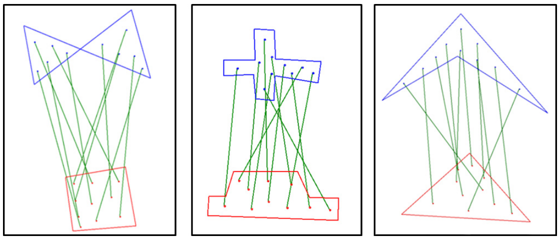


Fig. 3. Examples of agent-destination mapping from the original positions(bottom) to the destination positions in the new formation(top)

$$\mathbf{v^{AB}} = \mathbf{P_B^{Dest}} - \mathbf{P_A} \tag{3}$$

$$\mathbf{v^{BA}} = \mathbf{P_A^{Dest}} - \mathbf{P_B} \tag{4}$$

where $\mathbf{v^{AA}}$ is the directional vector from agent A's position to agent A's destination while $\mathbf{v^{BB}}$ is the directional vector from agent B's position to agent B's destination. Similarly,$\mathbf{v^{AB}}$ is the directional vector from agent A's position to agent B's destination and $\mathbf{v^{BA}}$ is the directional vector from agent B's position to agent A's destination. The agents will thus swap their destination position if the following criterion is met (see Figure 2).

$$||\mathbf{v^{AA}}|| + ||\mathbf{v^{BB}}|| > ||\mathbf{v^{AB}}|| + ||\mathbf{v^{BA}}|| \tag{5}$$

which can be computed faster by just comparing the square distance of each vector, such that

$$\mathbf{v^{AA}} \cdot \mathbf{v^{AA}} + \mathbf{v^{BB}} \cdot \mathbf{v^{BB}} > \mathbf{v^{AB}} \cdot \mathbf{v^{AB}} + \mathbf{v^{BA}} \cdot \mathbf{v^{BA}} \tag{6}$$

It is noteworthy that the above reorganization process does not need to be calculated for every time frame as the agents position is highly unlikely to change drastically. Rather it is sufficient to calculate periodically after every few seconds to reduce the computational requirements. The algorithm leverage on the neighborhood calculation process when choosing a random neighboring agent to swap their destinations, which is already an incurred computation cost needed to compute the steering forces, and hence the additional computation cost incurred is minimal.

4.3 Formation Shape Morphing

FAME allows user to change/morph the formation shape easily during runtime. Shape morphing refers to the process where a given shape transforms into another shape. In the context of shape constraint flocking, agents housed in a shape formation constraint morph or transform into another formation shape, while filling up the space inside the new formation strategically and naturally. Here, we showcase two formation morphing scenarios: 1) morphing of formation shape from a given initial formation shape to a new user-defined formation shape of interest and 2) a shape deformation approach to changing of formation shape.

Morphing Approach. In the first scenario, we consider how the flock of agent change its current formation to take up a new formation shape. First, a new formation has to be defined. Next, a uniform sampling is performed on the new formation to generate a set of n evenly distributed points inside the shape, where n refers to the number of agents in the initial formation. These points define the new position of that the agents should move to in order to fill up the new formation shape. Finally, a one-to-one mapping process is performed to assign every agent to its new destination position in the new formation shape. The desired mapping should be such that the total Euclidean distance from each agent's current position to its new destination is small. This is to allow the agent to form the new formation within the shortest possible time. The depictions of the mappings result considered in FAME are depicted in Figure 3.

Shape Deformation Approach. The second scenario showcases how agents adjust their positions within the formation when it undergoes some shape deformation. The method proposed in previous scenario works well in cases when the formation only requires to undergo one shape deformation. However it is computationally expensive when the changes to the formation shape is minor

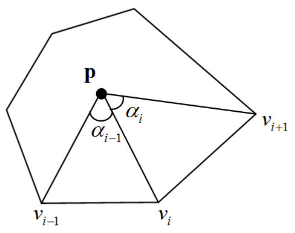


Fig. 4. Formation Shape Constraint

and continuous over time. Hence, this method is able to complement and make up for the drawbacks in the previous method.

Here, FAME uses the mean-value coordinate to calculate the Barycentric weight of each control point on the position of the agents inside a formation of any arbiturary shape. When the initial shape formation constraint is created and populated with n agents, the Barycentric coordinates is calculated for each of the initial position $P^{init} = \mathbf{P_0}, \mathbf{P_1}..., \mathbf{P_{n-1}}$ such that

$$\mathbf{p} = \sum_i B_i(\mathbf{p}) v_i,\ s.t. \sum_i B_i(\mathbf{p}) = 1 \tag{7}$$

where $\mathbf{p} \in P^{init}$ is the coordinate of the i control points defining the formation (see Figure 4). The mean value coordinate $B_i(\mathbf{p})$ is calculated as follows,

$$\sum_i B_i(\mathbf{p}) = \frac{w_i}{\sum_{j=1}^{k} w_j},\ and\ w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{||\mathbf{v_i} - \mathbf{p}||} \tag{8}$$

where $\alpha_i, 0 < \alpha_i < \pi$ is the angle formed in the triangle $[p, v_i, v_{i+1}]$.

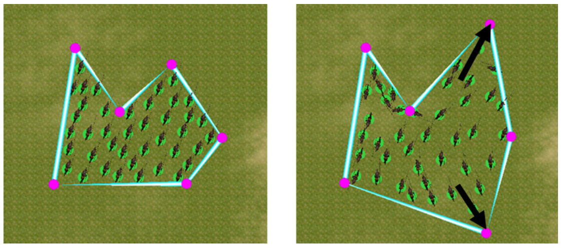


Fig. 5. Shape deformation approach to change formation from original formation (left) to the new formation (right)

With the computed Barycentric coordinate, the new position of the agent $\mathbf{p^{new}}$ can be easily determined when adjustments are made to the control points. A screen-shot of the shape morphing result using the deformation approach is depicted in Figure 5.

4.4 Flexible Formation Path following Mechanism

Path finding and navigation is one of the essential elements in digital games that allows game agent to navigate around in a complex environment in a natural and realistic manner. In addition, game environments are often non-static as such moving obstacles are commonly present in the scene and could also be introduced during runtime. Hence, it is natural for a flock of agents to be able to react and steer away from obstacles encountered in the scene autonomously, while at the same time the group formation shape should bend naturally along the path's curvature while negotiating the path. Here, FAME provides the functionality to define a path in the form of a NURBS. Each flock group can be tasked to perform path following along the specified path and while doing so, FAME automatically bends the shape constraint along the path's curvature, resulting in a more natural and realistic movement (see Figure 6).

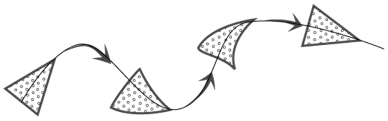


Fig. 6. Flexible formation - Ability to bend the formation along the path's curvature

Implementation details can be found in our previous work on Autonomous Multi-agents in Flexible Flock Formation [12].

4.5 Seamless Space Partitioning of Agent Object

FAME provides a seamless approach to partitioning the game environment, requiring no explicit efforts from the user or game developer, to support creation of a larger number of agents running in real time. One well known pitfall of the classical flock model is that each agent's steering decision is made based on the positions of the neighboring agent that lies within its visible radius. A brute force neighborhood query process would require the distance between every pair of agents to be computed, which would incur a computation complexity of $O(n^2)$. Such an approach does not scale well with large number of agents.

The underlying mechanism utilizes a quad tree data structure to subdivide the entire map into multiple quadrants. After game agents are populated, FAME performs a recursive search to place the agent in the respective leaf node of the quad tree and updates the node whenever the agent travels across quadrants. This speeds up the neighborhood query process as agents lying in quadrants that does not intersects in the region of interest are rapidly eliminated. This divide and conquer approach brings about a reduction of computational complexity from $O(n^2)$ to $O(n \log n)$.

5 FAME in Successful Commercial Game Launch

FAME library package was used successfully in collaboration Singapore-MIT Gambit Game Lab in the development of a commercial game, Dark Dot. Dark Dot is an action shooter games released on the Apple iPad platform, where the player controls a group of minions known as Darklets in a quest to defeat the enemy (see Figure 7). Here in Dark Dot, the advanced flock formation mechanics provided by FAME is used to control the locomotion of the Darklets.

Shortly after it was released in Oct 2011, Dark Dot became the top action game in 48 countries and clinched the number one position for several months

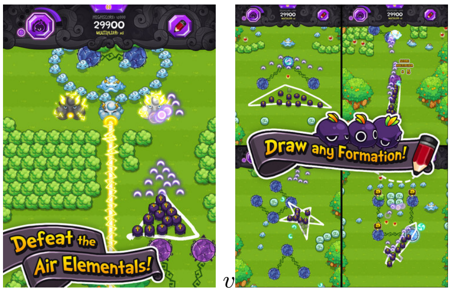


Fig. 7. Dark Dot created by Singapore-MIT Gambit Game Lab (Screenshots taken from Dark Dot official website)

in the iTunes apps store. It received notable comments from the press, citing Dark Dot as *"not only fun and creative and lovely to look at, but it's got some stunningly original game-play mechanics."*

Interested reader can visit the the Dark Dot official website for more information about the game: `http://gambit.mit.edu/loadgame/darkdot.php`. Similarly, more information about FAME can found in the following website: `http://c2inet.sce.ntu.edu.sg/FAME/`

6 Conclusion

In this paper, we have described FAME, a comprehensive C# library designed for easy creations of soft flock formation controls. In this soft formation control, the agents are able to temporary deviate from their formation when encountering anomalies and return to its formation when the path is clear, as well as being able to self-organize within the formation to reduce the overall time required to fill up the formation. Similarly, the formation is able to change or molded according to users preference during runtime and is able to bend naturally along with the curvature while negotiating the path. FAME also provides a seamless way to partition the scene so that the overall computation requirement is reduced. Using FAME technologies, an iPad game – Dark Dot, was launched on Apple iTunes app store and became the top action game in 48 countries. As future work, we hope to leverage from the emerging field of memetic computing technologies [20,21] towards more intelligent behaviors, decision making, social interaction and cultural evolution among virtual agents in games. Also, we shall consider the use of delicated hardwares [22] for real-time performance when intensive AI technologies are deployed.

Acknowledgment. We would like to thank Singapore-MIT GAMBIT Game Lab for the funding and support provided for this project.

References

1. Winfield, A., Erbas, M.: On embodied memetic evolution and the emergence of behavioural traditions in robots. Memetic Computing 3, 261–270 (2011)
2. Satizábal, H., Upegui, A., Perez-Uribe, A., Rétornaz, P., Mondada, F.: A social approach for target localization: simulation and implementation in the marxbot robot. Memetic Computing 3, 245–259 (2011)
3. Durupinar, F.: From audience to mobs: crowd simulation with psychological factors. Ph.D. dissertation, Bilkent University (2010)
4. Hughes, R.: The flow of human crowds. Annual Review of Fluid Mechanics 35, 169–182 (2003)
5. Chenney, S.: Flow tiles. In: Symposium on Computer Animation. SCA (2004)
6. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: Computer Graphics and Interactive Techniques. SIGGRAPH (1987)
7. Tu, X., Terzopoulos, D.: Artificial fishes: physics, locomotion, perception, behavior. In: Computer Graphics and Interactive Techniques. SIGGRAPH (1994)
8. Aubel, A., Boulic, R., Thalmann, D.: Real-time display of virtual humans: levels of details and impostors. IEEE Transactions on Circuits and Systems for Video Technology 10(2), 207–217, Comput. Graphics Lab., Swiss Fed. Inst. of Technol., Lausanne, Switzerland
9. van den Berg, J., Lin, M.C., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: IEEE International Conference on Robotics and Automation, pp. 1928–1935 (2008)
10. Sean, C., Jamie, S., Dinesh, M.: Way portals: Efficient multi-agent naviation with line-segment goals. In: Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games - I3D (2012)
11. Lin, M.C., Sud, A., Van den Berg, J., Gayle, R., Curtis, S., Yeh, H., Guy, S., Andersen, E., Patil, S., Sewall, J., Manocha, D.: Real-Time Path Planning and Navigation for Multi-agent and Crowd Simulations. In: Egges, A., Kamphuis, A., Overmars, M. (eds.) MIG 2008. LNCS, vol. 5277, pp. 23–32. Springer, Heidelberg (2008)
12. Ho, C.S., Nguyen, Q.H., Ong, Y.-S., Chen, X.: Autonomous Multi-agents in Flexible Flock Formation. In: Boulic, R., Chrysanthou, Y., Komura, T. (eds.) MIG 2010. LNCS, vol. 6459, pp. 375–385. Springer, Heidelberg (2010)
13. Erra, U., De Chiara, R., Scarano, V., Tatafiore, M.: Massive Simulation using GPU of a distributed behavioral model of a flock with obstacle avoidance. In: Vision, Modeling and Visualization, VMV (2004)
14. Erra, U., Frola, B., Scarano, V.: BehaveRT: A GPU-Based Library for Autonomous Characters. In: Boulic, R., Chrysanthou, Y., Komura, T. (eds.) MIG 2010. LNCS, vol. 6459, pp. 194–205. Springer, Heidelberg (2010)
15. Silva, A.R.D., Lages, W.S., Chaimowicz, L.: Boids that see: Using self-occlusion for simulating large groups on gpus. Comput. Entertain. 7, 51:1–51:20 (2010)
16. Reynolds, C.: Interaction with groups of autonomous characters. In: Game Developers Conference, pp. 449–460 (2000)
17. Passos, E.B., Joselli, M., Zamith, M., Clua, E.W.G., Montenegro, A., Conci, A., Feijo, B.: A bidimensional data structure and spatial optimization for supermassive crowd simulation on gpu. Comput. Entertain. 7, 60:1–60:15 (2010)
18. Singh, S., Naik, M., Kapadia, M., Faloutsos, P., Reinman, G.: Watch Out! A Framework for Evaluating Steering Behaviors. In: Egges, A., Kamphuis, A., Overmars, M. (eds.) MIG 2008. LNCS, vol. 5277, pp. 200–209. Springer, Heidelberg (2008)

19. Singh, S., Kapadia, M., Faloutsos, P., Reinman, G.: An Open Framework for Developing, Evaluating, and Sharing Steering Algorithms. In: Egges, A., Geraerts, R., Overmars, M. (eds.) MIG 2009. LNCS, vol. 5884, pp. 158–169. Springer, Heidelberg (2009)
20. Chen, X., Ong, Y.-S., Lim, M.-H., Tan, K.C.: A multi-facet survey on memetic computation. IEEE Trans. Evolutionary Computation, 591–607 (2011)
21. Nguyen, Q.H., Ong, Y.S., Lim, M.H., Krasnogor, N.: Adaptive Cellular Memetic Algorithms. Evolutionary Computation 17, 231–256 (2009)
22. Cao, Q., Lim, M.H., Li, J.H., Ong, Y.S., Ng, W.L.: A context switchable fuzzy inference chip. IEEE Transactions on Fuzzy Systems 14(4), 552–567 (2006)

Incremental Spatial Clustering in Data Mining Using Genetic Algorithm and R-Tree

Nam Nguyen Vinh[1] and Bac Le[2]

[1] Vietnam Informatics and Mapping Corporation
nguyenvinhnam@vietbando.vn
[2] Faculty of Information Technology – University of Science / National University of Ho Chi Minh City, Vietnam
lhbac@fit.hcmus.edu.vn

Abstract. In this article, we present an algorithm based on genetic algorithm (GA) and R-tree structure to solve a clustering task in spatial data mining. The algorithm is applied to find a cluster for a new spatial object. Spatial objects that represent for each cluster computed dynamically and quickly according to a clustering object in the clustering process. This improves the speed and accuracy of the algorithm. The experimental results show that our algorithm yields the same result as any other algorithm and is accommodated to the clustering task in spatial data warehouses.

Keywords: Spatial data mining, Clustering, Genetic Algorithm.

1 Introduction

Due to advanced data collection techniques such as remote sensing, census data acquiring, weather and climate monitoring etc. contemporary geographical datasets contain an enormous amount of data of various types and attributes. Analyzing this data is challenging for traditional data analysis methods which are mainly based on extensive statistical operations. Since classical data mining methods enable us to detect valuable information from extensive relational databases, spatial data mining (SDM) can be an appropriate technique for detecting possible interesting patterns in geographical datasets. SDM is a knowledge discovery process of extracting implicit interesting knowledge, spatial relations, or other patterns not explicitly stored in databases [13, 14].

Clustering is one of the tasks of spatial data mining. It is a typical unsupervised learning technique for grouping similar data points. A clustering algorithm assigns a large number of data points to a smaller number of groups such that data points in the same group share the same properties while, in different groups, they are dissimilar. Clustering has many applications, including part family formation for group technology, image segmentation, information retrieval, web pages grouping, market segmentation, and scientific and engineering analysis [3].

Genetic algorithms belong to a class of directed search methods that are be used for both solving optimization problems and modeling the core of evolutionary systems.

They use a heuristic rather than analytical approach, and thus their solutions are not always exact and their ability to find a solution often depends on a proper and sometimes fragile specification of the problem representation and the parameters that drive the genetic algorithm [1]. There have been proposals and approaches for the application of genetic algorithms for the clustering problems. These algorithms often encode chromosomes based on one or more cluster centers. The centers were chosen at random or determined by K-mean when initializing population [3, 6, 7, 8, 9]. Number of clusters is one of the input parameters, and a method used to determine these cluster centers affect to accuracy and speed of these algorithms.

In this paper, we propose an efficient incremental spatial clustering algorithm, which determines arbitrary shaped clusters. The algorithm based on Genetic Algorithms and R-tree structure. The proposed algorithm can work on very large database and yield the same result as any other algorithm.

2 Preliminaries

2.1 Spatial Clustering

Spatial clustering groups spatial objects such that objects in the same group are similar and objects in different groups are unlike each other. This generates a small set of implicit classes that describe the data. Clustering can be based on combinations of non-spatial attributes, spatial attributes (e.g., shape), and proximity of the objects or events in space, time, and space–time.

In general, the major clustering methods can be classified into the following categories [2]:

- **Partitioning methods:** Given a database of *n* objects or data tuples, a partitioning method constructs *k* ($\leq$ n) partitions of the data, where each partition represents a cluster. That is, it classifies the data into *k* groups, which together satisfy the following requirements: (1) each group must contain at least one object, and (2) each object must belong to exactly one group. Representative algorithms include k-means, k-medoids, CLARANS, and the EM algorithm.
- **Hierarchical methods:** A hierarchical method creates a hierarchical decomposition of a given set of data objects. Representative algorithms include BIRCH and Chameleon.
- **Density-based methods:** Most partitioning methods cluster objects based on the distance between objects. Their general idea is to continue growing a given cluster as long as the density (the number of objects or data points) in the "neighborhood" exceeds a threshold. Representative algorithms include DBSCAN, OPTICS, and DENCLUE.
- **Grid-based methods:** Grid-based methods quantize the object space into a finite number of cells that form a grid structure. Representative algorithms include STING, WaveCluster, and CLIQUE.

2.2 Genetic Algorithm

Genetic Algorithms (GAs) was invented by John Holland and developed this idea in his book "Adaptation in natural and artificial systems" in the year 1975. Holland proposed GA as a heuristic method based on "Survival of the fittest". GA was discovered as a useful tool for search and optimization problems.

The basic genetic algorithm is as follows [4]:

a. [Start] Genetic random population of n chromosomes (suitable solutions for the problem)
b. [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
c. [New population] Create a new population by repeating following steps until the New population is complete
   - [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).
   - [Crossover] With a crossover probability, cross over the parents to form new offspring. If no crossover was performed, offspring is the exact copy of parents.
   - [Mutation] With a mutation probability, mutate new offspring at each position in chromosome
   - [Accepting] Place new offspring in the new population.
d. [Replace] Use new generated population for a further sum of the algorithm.
e. [Test] If the end condition is satisfied, stop, and return the best solution in current population.
f. [Loop] Go to step b for fitness evaluation.

2.3 R-tree

An R-tree is a height-balanced tree similar to a B-tree with index records in its leaf nodes containing pointers to data objects. Nodes correspond to disk pages if the index is the disk-resident, and the structure is designed so that a spatial search requires visiting only a small number of nodes. The index is completely dynamic: inserts and deletes can be inter-mixed with searches, and no periodic reorganization is required [5].

3 Algorithm

Our clustering method is density-based. In our experiments, Euclidian distance has been applied. The proposed algorithm for clustering objects in the 2D plane is depicted as following:

```
**Input**: A set of spatial objects $P$ and the configure $K$
**Output**: A set of clusters $C$
$C \leftarrow \emptyset$
$foreach\ p\ in\ P$
    $Cluster\_GA(p, K, C)$
```

Where, $\boldsymbol{P}$ is a set of clustering objects. $\boldsymbol{K}$ is a configuration for the algorithm that includes a threshold for clustering and algorithm settings for GA – The GA operators selection, crossover, crossover probability, mutation and mutation probability. The function $\boldsymbol{Cluster\_GA}$ finds a cluster in $\boldsymbol{C}$or create a new cluster for object $\boldsymbol{p}$. We implement the algorithm by defining the steps of a genetic algorithm.

3.1 Encoding

A chromosome represents a cluster, and is encoded as shown in Fig. 1.

Valid_Flag	**ClusterId**	**SObjects**	**Reps**

Fig. 1. Encoding a cluster

Valid_Flag indicates whether this cluster is valid. A cluster is valid if it contains at least one spatial object. $ClusterId$ is a numeric value indicating cluster identity. $SObjects$ contains objects that belong to this cluster. The R-tree structure is used to store these objects in $SObjects$ to accelerate computing the fitness. $Reps$ is a subset of $SObjects$ that represents this cluster, and is determined after applying the fitness function defined in section 3.2. The number of objects in $Reps$ changes correspondingly to the position of a clustering object. An example demonstrating this encoding way shown in Fig. 2.

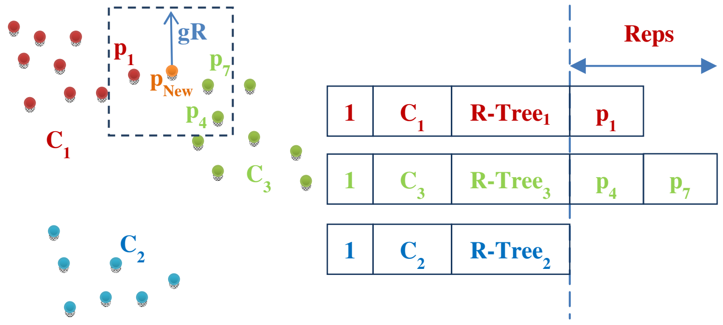


Fig. 2. An example of the cluster encoding

3.2 Fitness Function

The fitness function is applied for each valid cluster in C. It takes the R-tree structure of a cluster to find spatial objects to represent this cluster and to store them in the $Reps$ part. The fitness function acts in the following way:

- **Find candidates represent the cluster**
  We compute the minimum bounding rectangle MBR^p for clustering object p and expand this MBR^p with threshold gR through the following expressions:

$$\begin{aligned} MBR^{query}.left &= MBR^{p}.left - gR \\ MBR^{query}.top &= MBR^{p}.top - gR \\ MBR^{query}.right &= MBR^{p}.right + gR \\ MBR^{query}.bottom &= MBR^{p}.bottom + gR \end{aligned}$$

This MBR^{query} is used to search in the $SObject$ field of each cluster. The selected objects are the searching result. As shown in Fig. 2, $p_1$is a representative object for cluster C_1 and $p_4$and p_7 for cluster C_3. Cluster C_2 has none.

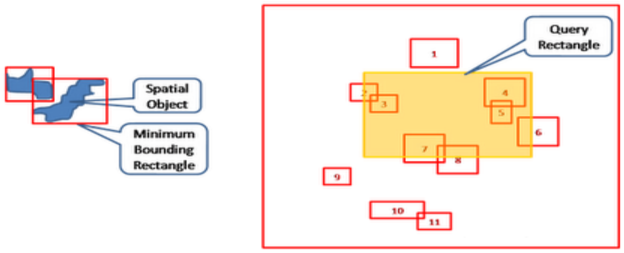


Fig. 3. Query spatial objects by R-tree

- **Returned value**

  The searching result determines the returned value of this function. The result returned by the function is 1 if the number of the representative objects $\neq \emptyset$, otherwise ∞.

Clusters having fitness value ∞ can not be applied genetic operators to accelerate algorithm performance.

3.3 Crossover

The algorithm randomly selects a position in the $Reps$ part of each parent cluster to test merging condition. If expression (1) below is satisfied, two parent clusters are merged into one single cluster.

$$Distance(Reps_i^m, p) \leq gR \ \&\& \ Distance(Reps_j^n, p) \leq gR \qquad (1)$$

$Distance$ is the Euclidian distance between two spatial objects. $Reps_i^m$and $Reps_j^n$ are two objects in $Reps$ of clusters m and n respectively. The example in Fig. 4 indicates the merging case (clusters C_1 and C_3 are going to be merged).

3.4 Mutation

This operator is used to insert clustering object p into a mutated cluster if the following condition (2) is satisfied:

$$Distance(Reps_k^m, p) \leq gR \qquad (2)$$

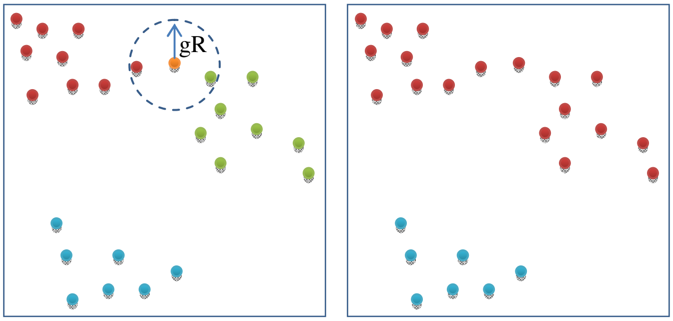


Fig. 4. Merging two clusters

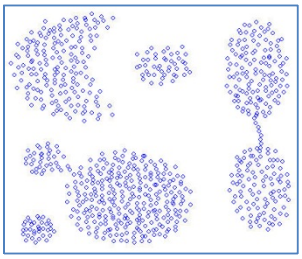
3.5 Algorithm Cluster_GA

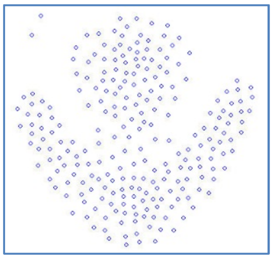
```
        $void\ Cluster\_GA(p, K, C)$
        {
[1]         $P \leftarrow \emptyset$
            $foreach\ cluster\ in\ C$
            {
                $[f, Reps] \leftarrow Fitness(p, cluster)$
                $if\,(f \neq \infty)$
                    $P.push(cluster)$
            }
            $if\,(P \neq \emptyset)$
            {
                $Crossover(p, P, K)$
                $Mutation(p, P, K)$
            }
            $else$
            {
[2]             $c_{new} \leftarrow CreateNewCluster(p)$
                $C.insert(c_{new})$
                $return$
            }
            $if\,(no.of\ generations < K.MAX\_GENERATIONS)$
                $goto$ [1]
            $else$
                $if\ p\ not\ belongs\ to\ any\ c\ in\ C$
                    $goto$ [2]
            $return$
        }
```

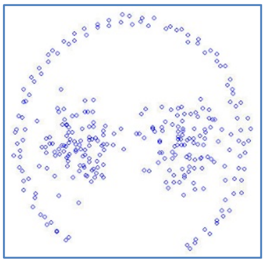
4 Experiments

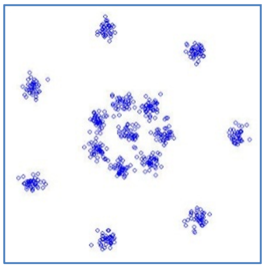
The experiments were done on a 2.40 GHz Intel® Core™ i5 machine with 2GB main memory. The program was compiled by the Visual Studio C++ 2010 compiler using level 3 optimization.

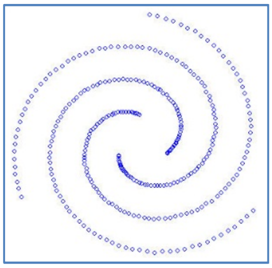
To evaluate our system, we use *Shape* datasets available from [12]. These are relatively small but quite diverse in shape, so they are suitable for checking the correctness of the algorithm for different distributions. Their shapes are shown in Fig. 5. The clustering performance of the proposed algorithm is demonstrated on two different datasets. Dataset *HitPosition_10631* includes 10.631 points and *clus100000* has 100.000 points. These datasets are illustrated in Fig. 6a, b.


(a) Aggregation


(b) Flame


(c) Pathbased


(d) R15


(e) Spiral

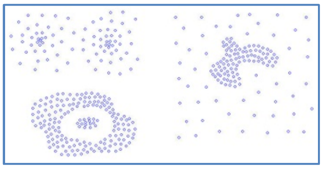

(f) Compound

Fig. 5. Shape Datasets

Clustering radius gR is dynamically computed for each dataset by expression (3) below. With *Shape* datasets, we adjusted the gR parameter by multiplying with a coefficient corresponding to each dataset to obtain desired results. These coefficients are listed in Table 1. The parameters settings for GA are shown in Table 2.

$$gR = \min(x_{max} - x_{min}, y_{max} - y_{min}) \,/\, 30 \tag{3}$$

With visualizing clustering results, we can easily check the algorithm's correctness on the *Shape* datasets. These results are shown in Fig. 7. It can be seen that the proposed algorithm could discover clusters of arbitrary shape.

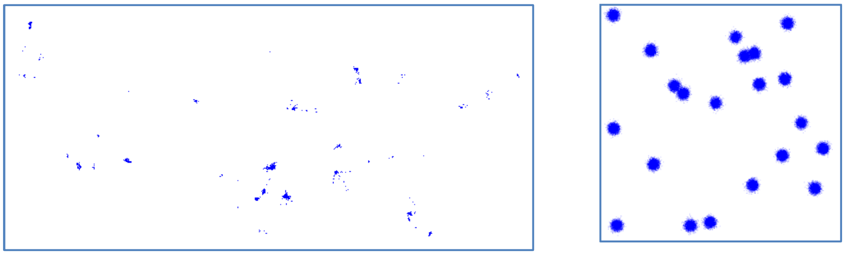

(a) Dataset HitPosition_10631 (b) Dataset clus100000

Fig. 6. Datasets for testing the scalability of our algorithm

Table 1. Adjusting coefficients for parameters gR

Dataset	Adjusting coefficient
Aggregation	2.00
Flame	3.00
Pathbased	2.05
R15	1.50
Spiral	1.50
Compound	2.50

Table 2. The parameters settings for GA

Settings Type	Value
Population size	250
Selection	Elitist
Crossover	One point
Mutation	Uniform
Crossover Probability	0.6
Mutation Probability	0.01
Maximum Generations	50

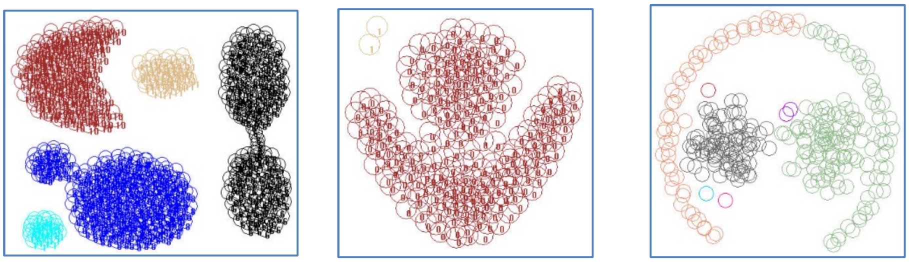
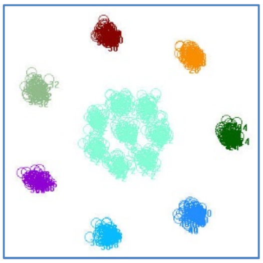
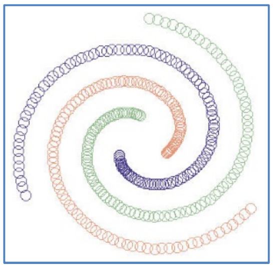

(a) Aggregation (b) Flame (c) Pathbased

Fig. 7. Clustering results of the Shape datasets


(d) R15


(e) Spiral

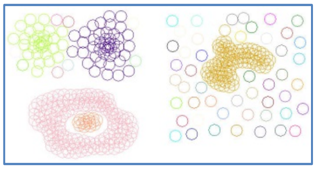

(f) Compound

Fig. 7. *(Continued)*

We used sweep-line algorithm [10] to verify the accuracy of our algorithm on larger datasets because it is easy to implement and gives more accurate results. The sweep-line algorithm with the same parameter *gR* of our algorithm found 36 clusters for *HitPosition_10631*and 17 clusters for *clus100000.* Our clustering results are shown in Fig. 8a, b and Table 3. It can be seen that the proposed algorithm could yield the same result as any other algorithm.

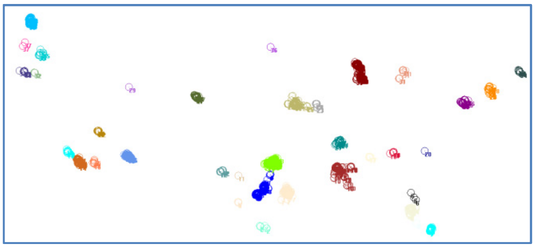


Fig. 8a. HitPosition_10631 clustered

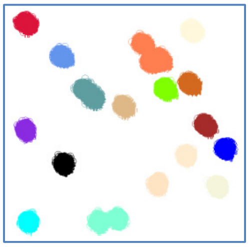


Fig. 8b. Clus100000 clustered

Table 3. The result of the experiment

No.	HitPosition_10631		Clus100000	
	Clusters	Time (ms)	Clusters	Time (ms)
1	36	343	17	15.865
2	36	328	17	15.865
3	36	343	17	15.538
4	36	343	17	15.928
5	36	327	17	15.600
6	36	358	17	15.568
7	36	327	17	15.600
8	36	328	17	15.538
9	36	328	17	15.881
10	36	328	17	15.600

R-tree structure is built based on the smallest rectangles that contain geometry objects without regard to their type. Moreover, the distance between two objects can be determined by the mathematical formula. Therefore, our algorithm can be applied for clustering different kinds of objects such as points, lines and polygons.

5 Conclusions

This paper introduces a new spatial clustering algorithm based on GA and R-tree structure. The proposed algorithm can detect clusters of arbitrary shape and yield the same result as any other algorithms. Furthermore, our algorithm can be applied to cluster any spatial object type, such as point, line and polygon. Spatial databases generally support the R-tree structure, so the algorithm is highly applicable.

References

1. Data Mining – Know It All. Morgan Kaufmann Publishers (2009)
2. Geographic Data Mining and Knowledge Discovery, 2nd edn. CRC Press (2009)
3. Pham, D.T., Afify, A.A.: Clustering techniques and their applications in engineering. Submitted to Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science (2006)
4. Introduction to Genetic Algorithms. Springer (2008)
5. Guttman, A.: R-tree: A dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, vol. 14(2) (June 1984)
6. Lu, Y., Lu, S., Fotouhi, F., Deng, Y., Brown, S.: FGKA: A Fast Genetic K-Means Clustering Algorithm. In: ACM Symposium on Applied Computing (2004)
7. Lu, Y., Lu, S., Fotouhi, F., Deng, Y., Brown, S.: Incremental Genetic K-Means Algorithm and its Application in Gene Expression Data Analysis. BMC Bioinformatics (2004)
8. Ding, Q., Gasvoda, J.: A Genetic Algorithm for Clustering on Image Data. International Journal of Information and Mathematical Sciences (2005)
9. Al-Shboul, B., Myaeng, S.-H.: Initializing K-Means using Genetic Algorithms. World Academy of Science, Engineering and Technology (2009)
10. Zălik, K.R., Zălik, B.: A sweep-line algorithm for spatial clustering. Journal of Advances in Engineering Software 40(6) (2009)
11. Lin, H.-J., Yang, F.-W., Kao, Y.-T.: An Efficient GA-based Clustering Technique. Tamkang Journal of Science and Engineering 8(2), 113–122 (2005)
12. `http://cs.joensuu.fi/sipu/datasets/`
13. Koperski, K., Adhikary, J., Han, J.: Knowledge discovery in spatial databases: Progress and Challenges. In: Proceedings of the SIGMID Workshop on Research Issue in Data Mining and Knowledge Discovery, Technical report 96-08. University of British Columbia, Vancouver, Canada (1996)
14. Koperski, K., Han, J.: Discovery of Spatial Association Rules in Geographic Information Databases. In: Proc. 4th Int. Symp. on Large Spatial Databases, pp. 47–66. Springer, Berlin (1995)

Personalized Email Recommender System Based on User Actions

Quang Minh Ha, Quang Anh Tran, and Thu Trang Luyen

Faculty of Information Technology
Hanoi University
Hanoi, Vietnam
{minhhq_fit,anhtq,tranglt_fit}@hanu.edu.vn

Abstract. Email is one of the most successful computer applications in the Internet, and email-spam is also the biggest problem for users, preventing them to quickly process the important emails in a shortest time. In this paper, we propose an email recommender system using user actions and statistical methods. Instead of a two-class classification with Spam and Ham, we treat the problem as a multi-class classification in which each class is a recommendation action from user to an email. The most common actions are: reply, read and delete. An experiment is also conducted to test the framework, using Naïve Bayesian classifier and different threshold to evaluate the relations between number of features and the performance. The experiment shows a promising result with good prediction accuracy.

Keywords: email filtering, text classifications, naïve bayesian classifier.

1 Introduction

Email is one of the most successful computer applications in the Internet with millions of user world-wide [1]. It is a primary interface for one's workplace [2], supporting the daily communications, exchanging information asynchronously. With the increasingly usability of email, the number of unsolicited bulk email, generally referred as email-spam [3] is also increased. Spam emails prevent users from quickly process wanted emails and reduce the usability effectiveness.

Many approaches have been proposed to address and solve this problem [4,5,6,7]. They can be grouped into two main categories: (1) based on email content, (2) based on email header. The first approach has higher prediction accuracy but low performance and depends on different languages. The second approach has higher performance but also higher error rate. In the first category, SpamAssassin and Bayesian Classification are the two main methods that are commonly used to filter spam emails.

The Bayes Classification was introduced by Paul Graham in 1998 [8]. Using Bayesian Classification, he filters email as Spam or Ham by evaluating the keywords' scores in the email content. This method depends mostly on the quality of the training dataset (collection of spam and ham emails). Another research of Androutsopoulos also addressed this method [9]. Following this method, contents of training emails will be extracted into tokens and store in a database. Then the Bayes theorem is used to evaluate the value of each token by taking the following criteria's:

- The frequency of the token that appears in the collection of SPAM emails
- The frequency of the token that appears in the collection of HAM emails
- The number of SPAM emails in the training dataset
- The number of HAM emails in the training dataset

When an email is coming, its content is also extracted into tokens and their scores in the database are used computed to have the probability of that email occurring in SPAM and HAM dataset. Based on the resulted probability, we can filter the email as SPAM or HAM. The pros and cons of this method are described below:

- Pros:
  - It is a simple probabilistic model in classification problem.
  - Can be easily adapt with the changes of SPAM email contents by extract new features from them and update to the database.
  - Easy to be personalized by user preferences
- Cons:
  - The filter has low performance with the new unknown emails. In order to improve this performance, it has to take time to learn as many features and SPAM emails as possible.

Besides Bayes, other text classification methods are also used to filter email-spam such as Neural Network, Support Vector Machine (SVM) [10-11]…

In overall, the two-class text classification methods can effectively filters spam emails as SPAM or HAM. It reduces the number of email sending to users. However, non-spam emails also have different level of prioritization. For example, emails that require users to reply are more important than need-to-read-only emails such as news feed emails, allowed advertisements or coupons, mailing list discussions… These not-important emails also reduce the time to process email quickly and correctly. Hence, it is author belief that an email filtering using multi-class classification will effectively help user to prioritize and process emails faster.

In addition, there is some other approaches share the same idea with this paper such as Microsoft SNARF, which sort emails based on Social Network and Relationship Finder [17]. Google, on the other hand, rely on more richful statistical data to sort user emails based on Content, Social, Thread and Label features [18].

In this paper, we propose a new framework of email recommender system using user actions and statistical methods. Instead of labeling emails as SPAM or HAM, we label emails with the personalized importance ranking based on user actions preferences. The possible number of user's actions may vary but mainly falls into the following general categories: (1) reply, (2) read, (3) delete or mark as spam. By using this approach, we can not only filter spam, but also suggest the action for users and prioritize emails by different actions based on user preferences. This method remarkably improves the time to process new emails. Summarizing, the major contributions of this paper are:

- We propose a new framework of email recommender system to filter emails by learning the user actions and classify emails based on these actions.
- We adapt the Naïve Bayesian Classifier and add a small modification during the feature selection process to improve the performance
- We prove that the new framework has low error rate and high performance

This paper is organized as follows: Section 2 introduces the preliminaries of Text classification problem, its algorithm and the selected algorithm in this paper which is Naïve Bayesian Classifier algorithm. A description of the dataset is also included. Section 3 discusses the proposed framework for filtering emails and suggests user actions, a method for high performance and good feature extraction is also described. Section 4 presents the experiment results and discussions. Section 5 gives the conclusion and discusses the future work of this research.

2 Preliminaries

2.1 Naïve Bayesian Classifier

Naïve Bayesian Classifier is a statistical text classification approach used to solve the problem of classifying documents. It is a particularly simple and effective classification method [14]. In details, each class in documents will have a score and the best class in Naïve Bayes is the most likely or maximum a posterior class CMAP:

$$C_{MAP} = \arg\max_{c \in C} \prod_{1 \le k \le n_d} P(t_k \mid c) \tag{1}$$

As can be seen, the conditional probabilities are multiplied with each other leading to the problem of floating underflow [13]. Hence, it is better to change the computation from multiplication of probabilities to sum of logarithm of probabilities because log(xy) = log(x) + log(y). Then the highest log probability still is the class of choice.

Therefore, the actual computation of Naïve Bayesian Classifier is:

$$C_{MAP} = \arg\max_{c \in C} [\log P(c) + \sum_{1 \le k \le n_d} \log P(t_k \mid c)] \tag{2}$$

where C_{MAP} is the class of maximum a posterior probability, P(c) is the prior probability of the document in class c, $P(t_k \mid c)$ is the probability showing how much evidence of a token t_k contributes to the class c to be the correct class.

2.2 Dataset

In this paper, a personal mailbox was used to conduct the experiments. We use a blank mailbox and subscribe to many mailing lists, arranging into two main categories: (1)

security discussions, (2) chemistry discussions. The mailing lists are available at http://www.securityfocus.com (for security emails) and https://listserv.indiana.edu/cgi-bin/wa-iub.exe?A0=CHMINF-L (for chemistry emails).

The emails then were being fetched via IMAP into a MySQL Database to store the information about sender, receiver, subject and body. Then we start to train the emails to mark them as one of the three classes: reply = 0, read = 1, delete or spam = 2. This training activity is based on the following criteria's:

- Emails about web application security and security basics are being replied
- Other emails in security focus mailing lists are being read
- Emails from chemistry discussions are being deleted

In total, there are 950 emails in the dataset and the number of email in each class is 245, 480, 225 respectively for reply, read and delete. As can be seen, the number of emails in class 1 (read) is 2 times larger than other two classes. This based on the actual use of common email users which often have many read mails but not many reply and delete ones.

After having the trained emails with specific actions, we run the feature selection program to extract tokens and select the best features.

3 Theoretical Framework

3.1 Descriptions of the Framework

The framework of our email recommender system is described in Figure 1. The process can be summarized as follows:

- When an email is received, a tokens extraction process will start to extract all the possible tokens in the email subject and body and produces a vector for each email
- The vector then being used as the input for the Classifier to classify the email action (or class) using the existing features in the database. The Classifier may vary and depends on the selection of the researcher. Statistical classification algorithms such as Naïve Bayes, Neural networks, Support vector machines (SVM), Decision trees, Hidden Markov model can all be used as the Classifier in this framework.
- After the Classifier finishes its job, it will filter the emails into appropriate categories of actions (for example: reply, read or delete)
- During the filtering process, user preferences are taken into account. For example: if user thinks the classifier filters a wrong email, he or she would correct the filtered result by changing the class. This action would help to re-train the email content by running a feature extraction which will be described in the section 3.4 of this paper.

3.2 Classifier

In general, this Classifier can be any statistical classification algorithms, including Naïve Bayes, Neural Networks, Decision Trees, Support Vector Machines… In this

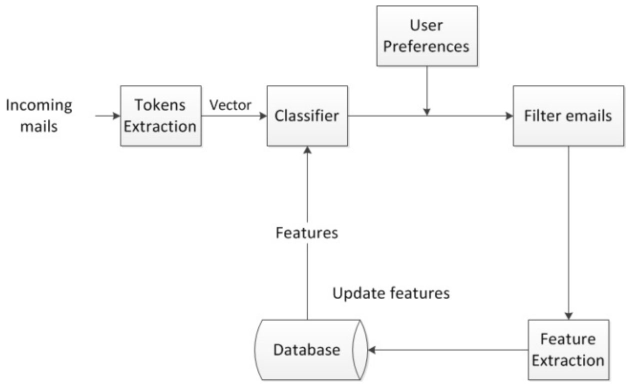


Fig. 1. Email recommender system based on user actions preferences

paper, we use Naïve Bayes because of its simplicity. Therefore, we can test the framework quickly.

Using Naïve Bayesian Classifier, given the scores of features in the Database, we can calculate the score of each incoming email in different classes. Then the maximum score is the target class. The algorithm can be displayed as follow:

```
Procedure classifyNaiveBayesWord
  output: the class v that email is classified into
  input:
  words = a collection of tokens from a input email content
  global:
    vocabulary = Get all extracted tokens in the training dataset
    positions = number of time words found in vocabulary
    examples = number of training dataset
    docs_count = number of emails in each class in the training dataset
    result = list of result
V = list of classes
   for each class v in V:
      p_v = docs_count[of class v] / examples
      vnb = log(p_v)
      for each i in positions:
            vnb += log(score of term i in class v in the features database)
      if (vnb != p_v):
          result ← Insert vnb of class v to result
   sorted_results = sort results from max to min
   if   no element in sorted result:
      return 2 {spam}
   return class v
```

3.3 Tokens Extraction

There are many tokenizing methods available such as N-Grams (for character and for word), Word tokenizing [16]. However, to keep the performance of this process – as email content may contains many tokens, we use a simple regular expression to extracts words and some special characters. The regular expression is described below:

```
'[\w@\-]+'
```

Where *[…]* means group of matching patterns, \w means match every words, @ means match the @ letter, \\- means match the – letter, + means many times.
As the result, we can extract 205061 tokens from 950 emails in the training dataset.

3.4 Feature Extraction

Figure 2 shows the diagram of feature selection process. The components of the diagram are described as follows:

- *Training emails*: the database containing email content, including from, to, subject, body with a specific action label.
- *Combine subject and body into string:* we combine the subject and body of each email into a string, separate with a blank space
- *Get tokens:* given the string of subject and body, we use the regular expression above to extract tokens and produce a vector
- *Get thresholds:* each class (or action) has a threshold to ensure the large differences between score of each token.
- *Choose tokens:* we evaluate tokens' scores using *m-estimate of probability* [12].

$$P[t_k \mid v_j] = \frac{n_k + 1}{n + |Vocabulary|}$$

Where:

n is the total number of tokens in all training emails which belongs to class v_j, n_k is the number of times token t_k found in n tokens, and |Vocabulary| is the total number of tokens in the training emails.

Each token has a list of scores in different classes, calculated using the above *m-estimate of probability*. The token will be selected as a feature if and only if it satisfies the following condition:

$$S_1 - S_2 > S_1 * T_1 \qquad (3)$$

where: S_1 and S_2 is the maximum and second maximum score of a token, T_1 is the threshold of the class which has the maximum score S_1.

This constraint helps to ensure the appropriate distance between the score of selected class and other scores.

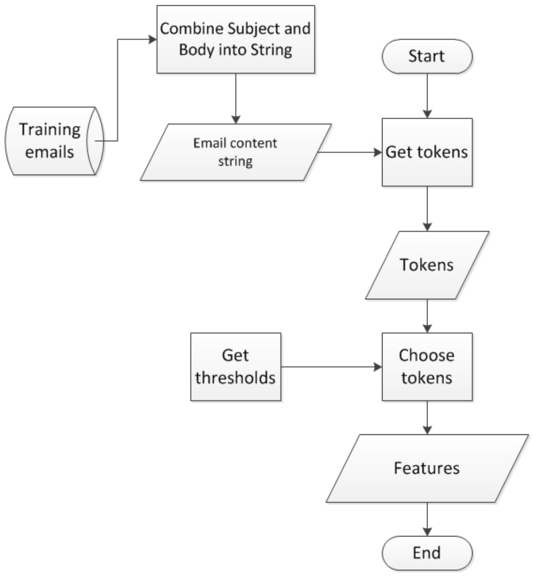


Fig. 2. Feature selection

4 Experiments

4.1 Experiment Settings

There are three common actions of user relate to an email: [0] = Reply, [1] = Read and [2] = Delete or mark as spam. In this paper, we will conduct the experiment to classify emails into three above classes using Naïve Bayesian Classifier. By trying different thresholds set, we will have different set of tokens to evaluate the performance based on number of features selected. The experiment settings are shown in the Table 1:

Table 1. Experiment settings

Settings	Value	Justifications
Number of classes	3 classes	Three most common user actions to an email
Number of tokens from training emails	205061 tokens	N/A
Thresholds [Reply, Read, Delete]	Set 1: 0.5, 0.5, 0.5	Base thresholds
	Set 2: 0.75, 0.75, 0.75	High thresholds
	Set 3: 0.8, 0.8, 0.8	Very high thresholds
	Set 4:0.65, 0.6, 0.75	Different threshold elements

4.2 Naïve Bayesian Classifier Performance

With each threshold set, we will calculate the following parameters: the time to train all data, features selected, average time to classify an email and prediction accuracy of each class. Table 2 shows the experiment results:

Table 2. Naive Bayesian Classifier performance with different threshold set

	Set 1	Set 2	Set 3	Set 4
Training time(s)	8977	3053	2332	4405
Features selected	7895	1880	1226	3016
Avg. running time	13.5	4.3	3.65	5.9
Prediction accuracy:[0]	91.84%	79.59%	56.73%	85.31%
[1]	73.54%	69.58%	76.67%	97.78%
[2]	98.22%	97.78%	97.78%	75.21%
Overall	87.88%	82.31%	77.06%	86.1%
Avg. rate	**83.3%**			

As can be seen, the larger number of features, the better prediction accuracy we have. There is also a fact that the prediction accuracy of class 1 which is the action "Read" always below 80% that is because there are many similarities between emails mark as reply and read. They are both emails about security and IT bugs. The average prediction accuracy keeps at 83 to 87%. Furthermore, we can also notice the relation between number of features, average classifying time and prediction accuracy. These relations are shown in Figure 3:

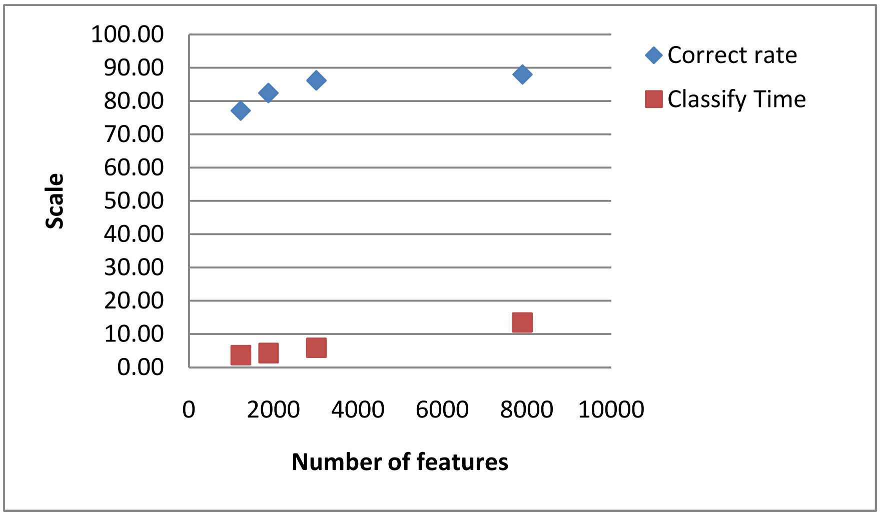


Fig. 3. Relation between number of features, prediction accuracy and classifying time

As can be seen from Figure 3, there's a big change in the running time when the number of features are more than about 3000 features. It increases from 6 to nearly 14 seconds (more than 2 times). Hence, the number of features should not be too high.

Figure 3 also shows that the ideal number of features should be in the range of 2500 to 4000 features. Above this range, the average prediction accuracys only increase 2%.

4.3 Discussions

Table 2 shows that the Set 1 with Thresholds is 0.5, 0.5, and 0.5 have the best prediction accuracy at 87.88% in average. This average rate is not as high as a two-class classifier [15]. However, when looking at the prediction accuracy where it can filters email as "Reply" and "Delete / Spam" are remarkably high at 91.84% and 98.22% respectively. The lowest result comes from the class of "Read" emails. There are two reasons for this lower-than-expected result. Firstly, the number of "Read" emails in the training dataset is two times larger than the number of reply and delete emails. Secondly, "Read" and "Reply" emails share the same topic which is IT Security. Hence, many emails from "Read" class contain the features which belong to "Reply" class.

5 Conclusion

In this paper, we have proposed a personalized email recommender system framework to redefine the way we filter emails based on content using statistical classification methods. Instead of treating email filtering as a two-class classification problem (spam and ham), we treat it as a multi-class classification problem, each class represents an action of user to an email. Using Naïve Bayesian Classifier as the sample classifier, the experiment shows a promising result with good prediction accuracy. In the future work, we will continue to optimize the feature selection algorithms to get better features in shorter times. Also, we will change the Classifier to some other statistical classification methods such as support vector machine, neural networks to evaluate the performance between different methods.

References

1. Steve, W.: Email overload: exploring personal information management of email. In: CHI 1996 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Common Ground, vol. 96(1), pp. 276–283 (1996)
2. Nicholas, K.: Automated email activity management: an unsupervised learning approach. In: IUI 2005 Proceedings of the 10th International Conference on Intelligent User Interfaces, vol. 5(1), pp. 67–74 (2005)
3. Anirban, D.: Enhanced email spam filtering through combining similarity graphs. In: WSDM 2011 Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, vol. 11(1), pp. 785–794 (2011)

4. Khurum, N.J.: Automatic Personalized Spam Filtering through Significant Word Modeling. In: ICTAI 2007 Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, vol. 2(1), pp. 291–298 (2007)
5. Yiming, Y.: Personalized Email Prioritization Based on Content and Social Network Analysis. IEEE Intelligent Systems 25(4), 12–18 (2010)
6. Paul-Alexandru, C., Jörg, D., Wolfgang, N.: MailRank: using ranking for spam detection. In: CIKM 2005 Proceedings of the 14th ACM International Conference on Information and Knowledge Management, vol. 5(1), pp. 373–380 (2005)
7. Mingjun, L., Wanlei, Z.: Spam Filtering based on Preference Ranking. In: CIT 2005 Proceedings of the The Fifth International Conference on Computer and Information Technology, vol. 5(1), pp. 223–227 (2005)
8. Graham, P.: A plan for spam. Web Document (2002), `http://www.paulgraham.com/spam.html`
9. Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G., Spyropoulos, C.D.: An evaluation of Naive Bayesian anti-Spam filtering. In: Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, Barcelona, Spain, pp. 9–17 (2000)
10. Drucker, H., Wu, D., Vapnik, V.: Support Vector Machines for spam categorization. IEEE Transaction on Neural Networks 10(5), 1048–1054 (1999)
11. Ozgur, L., Gungor, T., Gurgen, F.: Adaptive anti-spam filtering for agglutinative languages: a special case for Turkish. Pattern Recognition Letters 25, 1819–1831 (2004)
12. Mitchell, T.: Bayesian Learning. In: Tom, M. (ed.) Machine Learning, p. 179. McGraw-hill, USA (1997)
13. Manning, C.D.: Naive Bayes text classification (2008), `http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html` (last accessed April 10, 2012)
14. Manning, C.D.: Text classification and Naive Bayes (2008), `http://nlp.stanford.edu/IR-book/html/htmledition/text-classification-and-naive-bayes-1.html` (last accessed April 13, 2012)
15. Ion, A.: An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In: SIGIR 2000 Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, vol. (1), pp. 160–167 (2000)
16. Manu, K.: Building Search Applications: Lucene, LingPipe, and Gate, p. 22. Mustru Publising, US (2008)
17. Carman, N.: The Social Network and Relationship Finder: Social Sorting for Email Triage. In: Conference on Email and Anti-Spam, p. 149 (2005)
18. Douglas, A.: The Learning Behind Gmail Priority Inbox (2010), `http://research.google.com/pubs/archive/36955.pdf` (last accessed June 12, 2012)

Developing Attention Focus Metrics for Autonomous Hypothesis Generation in Data Mining

Bing Wang, Kathryn E. Merrick, and Hussein A. Abbass

School of Engineering and Information Technology,
University of New South Wales, Canberra, Australia

Abstract. When facing a data mining task, human experts tend to be responsible for proposing the hypotheses that lead to the discovery of interesting patterns. Recently, there is interest in automating the hypothesis generation process to reduce the load on the human expert during data mining. However, if we want an artificial agent to undertake this new role, we also need new metrics to measure the success of the hypothesis generation mechanism. This paper explores the design of metrics for evaluating hypothesis generation algorithms in terms of differences in the way they focus attention in the data mining search-space. We demonstrate our new metrics applied to three stochastic search based prototype hypothesis generation algorithms. Results show that some differences in attention focus can be identified using our metrics. Directions for further work in attention focus metrics and hypothesis generation algorithms are discussed.

Keywords: Hypothesis generation, data mining, evolutionary computation.

1 Introduction

In data mining tasks, human experts are in general responsible for the high level analysis of hypothesis generation. For example, deciding the representation to describe discoverable patterns, the criteria by which a particular pattern can be considered as validly identified, and which specific search method to use. After all these options for data mining are selected, the problem becomes an optimization problem, the result of which can be automatically found by different, well studied algorithms with well understood evaluation metrics, e.g. accuracy based on least square error.

When facing a data mining task, human experts tend to propose hypotheses that can discover potential interesting patterns. However, this interestingness is usually biased by the expert's domain knowledge and attention. In this paper, we investigate the automation of the hypothesis generation process by using evolutionary computation (EC) algorithms, which could potentially compensate the bias brought by human experts and speed up the knowledge discovery in databases(KDD) [1] process. In general, the main motivation for using EC to execute this high level analysis of hypothesis generation is that the algorithms in EC family have flexible representation forms to cast data mining task components; few assumptions about problem domain are made by EC algorithms and they perform global search which could overcome the bias brought by human experts. In addition, if we want an artificial agent to undertake more roles in this process, we also need metrics to measure the success of the hypothesis generation mechanism, specially the attention focus of the hypothesis generation agent on different aspects of potential hypothesis. This paper explores design of metrics for evaluating attention focus in hypothesis generation, we also apply our new metrics to three prototype hypothesis generation algorithms. The rest of the paper is organised as follows: Background is discussed in Section 2. The proposed search attention metrics are introduced in Section 3, followed by stochastic search based hypothesis generation paradigm with applications on a data mining task in Section 4. Experiments are then presented in Section 5 and conclusions are drawn in Section 6.

2 Background

2.1 Hypothesis Generation

Autonomous hypothesis generation has been widely studied in different fields under different scenarios. Resilient robot described in [2] can recover from damage autonomously by continuously self modelling. The four-legged machine continuously infers its own structure by synthesizing multiple completing candidates of self-model hypotheses. The most accurate hypothesis is then used to generate an alternative new gait from a damaged situation. In functional genomics, King et al. [3] proposed an inductive logic programming method to automatically generate hypotheses about yeast metabolism pathway, which significantly reduces human invention in circles of scientific experiments. Within medical domain, Moss et al. [4] reported an ontology-driven tool which can detect anomalous patient responses to treatment and further suggest hypothesis to explain the anomaly. What these studies have in common is that they investigated methodologies to extend artificial intelligence techniques to higher level analysis which traditionally done by human experts. The proposed stochastic search based hypothesis generation paradigm in this study investigates similar concept in data mining methods of KDD process.

2.2 Attention Focus

Attention focus describes the behaviour that artificial agent selects a subset of sensor data and/or a subset of internal structure it considers while acting and learning. These forms of focus of attention are termed as perceptual selectivity and cognitive selectivity [5]. An agent can employ different methods for attention focus. Incorporating attention focus in agent design enables the autonomous generation of complex and creative learning behaviours of an agent.

Foner et al [5], proposed to use perceptual selectivity and cognitive selectivity to constraint the agent to perceiving certain subset of sensor data and its internal structure involved in responding. This method reduces the complexity of unsupervised learning without harming correctness. Unlike the predefined constraints on attention focus, Oudeyer and Kaplan [6] presented an Intelligent Adaptive Curiosity (IAC) mechanism to enable a robot to autonomously interact with certain subjects in which it maximizes its learning process. This permits autonomous mental development without requiring supervision. Graziano et al. [7] suggested using compression progress to achieve similar autonomous exploration behaviour. In our study, we intend to design hypothesis generation agents that can help overcome the bias introduced in human experts' data analysis work, that is, the agent does not only focus attention on certain type of hypotheses, the agent should shift attention for generating different types of hypotheses to extract as much knowledge as possible from data sets. In the following section, we first explore the design of evaluation metrics for the characterisation of attention focus shift in hypothesis generation.

3 Characterising Attention Focus in Hypothesis Generation

This section presents a particular interest of current work, which is the variance of behaviour of different search based hypothesis generation algorithms. We propose two attention focus metrics to analysis the behaviours of prototype stochastic search base hypothesis generation algorithms.

3.1 Complexity of Hypothesis Generation

One aspect of analysing hypothesis generation methods is the complexity of generated hypotheses. In the KDD process, easily interpretable patterns and patterns of simple forms are preferred as interesting. However, insights into the constituent components of given data set may also be gained by more sophisticated hypotheses, which also possess usefulness property. Therefore, we propose complexity as an attention focus metric for hypothesis generation. In this paper, we define complexity as average number of independent variables for each dependent variable in functions found. The definition of this specific data mining task for prototype hypothesis generation agent will be introduced in the next section. The metric of complexity can also be generalised to other forms for more complex data mining tasks.

$$C_{x_i} = \frac{\sum_{k=1}^{N_{x_i}} J_k}{N_{x_i}} \tag{1}$$

where C_{x_i} represents the complexity of variable x_i as a dependent variable. N_{x_i} is the number of functions found with x_i being dependent variable. J_k is the number of independent variable in the kth function among those functions with x_i being dependent variable. This metric measures the attention focus of different prototype agents on the span of variables included in hypotheses generated.

3.2 Perceptual Selectivity in Hypothesis Generation

Perceptual selectivity as a form of attention focus was first proposed in unsupervised learning for the purpose of reducing computational complexity. Hypothesis generation in data mining methods requires incorporation of different components, e.g. representation, evaluation, variables and so on. Each component also has multiple candidates for the generation process. These components constitute the perceptual environments for a hypothesis generation agent. Perceptual selectivity therefore well suits to the characterisation of the autonomous hypothesis generation process. We thus use this technique to characterise the agent's changing attentions. Specially in the prototype agents we define perceptual selectivity as frequency of independent variables.

$$F_{(x_j,x_i)} = \frac{N_{(x_j,x_i)}}{N_{x_i}} \tag{2}$$

where $F_{(x_j,x_i)}$ is the frequence of x_j as an independent variable appearing in functions found where x_i acts as dependent variable, $i \neq j$. $N_{(x_j,x_i)}$ represents the number of found unique functions where x_i and x_j both exist, while N_{x_i} is the number of found unique functions where x_i is dependent variable. This index could reveal whether different agents focus attention on different subset of independent variables in their perceptual environment.

4 Prototype Hypothesis Generation Algorithm for Multiple Linear Function Mining Task

This section introduces the proposed prototype stochastic search based hypothesis generation, its implementation on a simplified data mining task and the representation form of potential hypothesis for the defined data mining task.

4.1 Stochastic Search Based Hypothesis Generation

Searching techniques are traditionally used for optimization problems. In this study, we apply the stochastic search paradigm to autonomous hypothesis generation. The generation process is then realised by searching for the optimum combinations of hypothesis components that can reveal interesting patterns from given data set. This stochastic search based hypothesis generation relies on EC. EC makes few assumption about problem domain; its flexible chromosome encoding scheme makes it possible to represent different hypothesis components; its global search ability has the potential to overcome the bias in human experts' analysis.

In this paper, we use this stochastic search paradigm on a simplified data mining task with minimum components for hypothesis generation as a starting point, but it still captures the main concepts in our study. The data mining task is defined as follows: We assume a data set with V instances of vector

$X = (x_1, x_2, ..., x_n)$,where n is the dimensionality of X. Suppose there may be K possible models of the form:

$$x_p = f_k(x_{q(1)}, x_{q(2)}, ..., x_{q(j)}, ..., x_{q(J)}) \tag{3}$$

hidden in the data set, where $\{x_p\} \bigcap \{x_{q(1)}, x_{q(2)}, ..., x_{q(j)}, ..., x_{q(J)}\} = \phi$, $\{x_p\} \cup \{x_{q(1)}, x_{q(2)}, ..., x_{q(j)}, ..., x_{q(J)}\} \in X$, $J + 1 \leq n$, $k \in \{1, ..., K\}$. The knowledge discovery goal in this problem is set to discover linear functions from given data set. Therefore, f_k are all linear functions. The constituent components of a hypothesis for the above problem definition are that, which subset of the variables has linear relation and within this subset of variables, which variable acts as a dependent variable (DV). The detailed representation of hypothesis is given in next section.

4.2 Representation of Hypothesis

The binary chromosome coding example for the constituent components of potential hypothesis is shown in Table 1. Suppose the length of vector X is $n = 8$, The value 1 in one chromosome represents that the corresponding variable in vector X acts as dependent variable (DV) in a hypothesis, while 0 represents independent variable (IV). Such representation can include multiple hypotheses in one chromosome. The identification of hidden linear function is determined by the correlation of multiple determination (R^2) of each hypothesis tested again given data. If R^2 derived from multiple linear regression is greater than a predefined threshold, then preliminary hidden knowledge is identified. The hypothesis is then rechecked to eliminate redundant independent variables. Objective function for each chromosome is defined as the average R^2 of the hypotheses included in one chromosome. We selected three EC algorithms to implement the hypothesis generation agent, at the same time to investigate whether the proposed attention focus metrics could capture attention shifting in the three hypothesis generation agents. The selected algorithms are genetic algorithm (GA) [8], population based incremental learning (PBIL) [9] and differential evolution algorithm (DE) [10][11]. Our current data mining task is a first attempt to test the prototype hypothesis generation algorithm, therefore we keep this implementation task simple. Although DE, as a continuous optimization method, could not be directly based on discrete representation as the above settings, in the future study, with the complexity of data mining tasks increases, we will potentially need to extend the binary representation to other forms. Therefore, here, we still include DE into the experiments.

5 Experiments and Discussion

The proposed methods and metrics are applied to data set downloaded from UC Irvine machine learning repository (UCI) and locally generated artificial testing data set. The four data set from UCI are: (1) breast cancer Wisconsin

Table 1. Chromosome Coding for Hypothesis Generation

Chromosome	Corresponding Hypotheses
11100000	$x_1 = f(x_4, x_5, x_6, x_7, x_8)$
	$x_2 = f(x_4, x_5, x_6, x_7, x_8)$
	$x_3 = f(x_4, x_5, x_6, x_7, x_8)$
10001100	$x_1 = f(x_2, x_3, x_4, x_7, x_8)$
	$x_5 = f(x_2, x_3, x_4, x_7, x_8)$
	$x_6 = f(x_2, x_3, x_4, x_7, x_8)$

(prognostic) data set, (2) breast cancer Winsconsin (diagnostic) data set, (3) wine data set, (4) housing data set. Those instances that have missing values are deleted from original data set. Summary of experiment data set is in Table 2. Note that, the original data mining tasks of these data set are not important in this paper, we used these data set as we have no prior knowledge about these data set, the task is defined in section 4.1. Parameters used in experiments are summarised in Table 3. If the hypothesis is tested on input data and returns a R^2 value greater than 0.9, this hypothesis is a valid. Each hypothesis generation agent has 30 runs on the each data set.

As described in Section 4.1, the knowledge discovery goal in the defined data mining task is to discover unique linear functions from given data. Unique functions are unique separations of dependent variable and independent variables, at the same time, the R^2 of this function should be greater than 0.9 to be valid when testing against input data set. In addition, those functions having R^2 greater than 0.9 are re-checked to eliminate variables that do not contribute to the linear relation. For example, if x_1 is a function of x_2, x_3, and x_4 with R^2 greater than 0.9, if R^2 is still greater than 0.9 after the removal of x_2, x_2 is not included in the real linear function.

The average number of unique hidden linear functions found by each method is shown in Table 4 along with the standard deviations. For the first two data set, the best performance occurred when we used DE as the underlying search mechanism; while the PBIL found the least number of unique functions. The outcome suggests that DE keeps better diversity in its evolution than the other two methods do. DE ensures that in its evolution, child is not an exact copy of parents by selecting a random variable in chromosome and force it to crossover regardless of the crossover probability. Whereas, PBIL attempts to create a probability vector which can be considered as a prototype for those individuals having high fitness value, therefore it does not perform as well as other two methods in keeping diversity. GA achieves slightly better performance over the data set 3, but the differences are not statistically significant. As for data set 4 and 5, no difference among three agent is revealed in the results; this could attribute to that there is no hidden linear function in the search space (data set 4) and all hidden functions in search space have been identified (data set 5).

Table 2. Data Set Summary

Data	No. Attributes	No. Instances
1	34	194
2	31	569
3	14	198
4	14	506
5	8	60

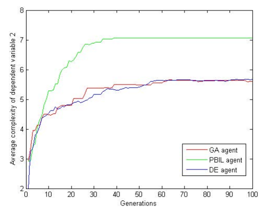
Table 3. Experiment Parameters

Parameters	GA	PBIL	DE
Population size	100	100	100
Generation size	100	100	100
Crossover rate	0.7	0.7	0.7
Mutation shift	n/a	0.02	n/a
Mutation rate	0.02	n/a	n/a
R^2	0.9	0.9	0.9

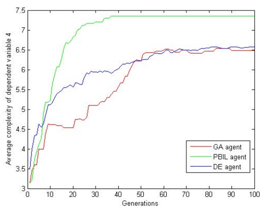
Table 4. Average Number of Unique Linear Functions over 30 Runs

Data	GA	PBIL	DE
1	6667.5 ± 531.7	3658.9 ± 280.3	10350.0 ± 200.2
2	10036.0 ± 560.9	6361.9 ± 443.2	15223.0 ± 224.1
3	0.9 ± 0.31	1.0 ± 0.0	0.7 ± 0.45
4	0.0 ± 0.0	0 ± 0.0	0.0 ± 0.0
5	7.0 ± 0.0	7 ± 0.0	7.0 ± 0.0

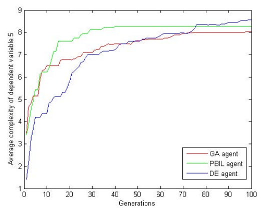
We applied the two proposed metrics on these three hypothesis generation agents. For the complexity attention focus metric, we only present experiment results visualizations on data set 1 in Figure 1 due to the space limitation of this paper. Attention focus analysis is executed along the dimension of generations, we calculated the average number of independent variables for each dependent variable in functions found accumulated over generations. Although there are general trends of agents focusing attention increasingly on complex hypotheses across the plots, variations of attention focus of different agents can still be spotted. We can see, from Figure 1, that PBIL agent in general focus attention on more complex hypotheses than the other two agents. In addition, the attention of PBIL agent on complexity stops to change after about 40 generations of evolution. That is, around 40 generations, the probability vector of PBIL agent reaches



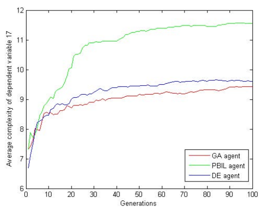
(a) Complexity analysis of DV x_2



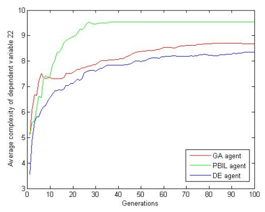
(b) Complexity analysis of DV x_4



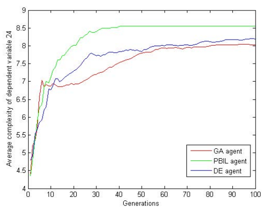
(c) Complexity analysis of DV x_5



(d) Complexity analysis of DV x_{17}



(e) Complexity analysis of DV x_{22}

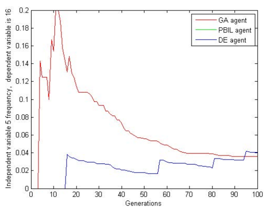


(f) Complexity analysis of DV x_{24}

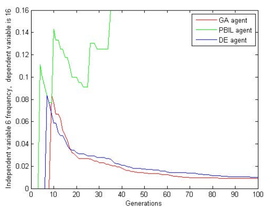
Fig. 1. Average complexity of dependent variable along generations

a stable stage, fixed mutation rate does not have much effect on deviating the probability vector from generating similar populations. Although the behaviours of GA agent and DE agent are similar in terms of attention on complexity, the complexity metric is able to reveal the differences of level of attention focus on complexity at different stage. For instances, in Figure 1 (b), after around 50 generation, DE agent and GA agent have similar level of attention on complexity measured value 6.5; But in the early generations, we can notice from the plot that DE agent shows more interest on complex hypotheses than GA agent does.

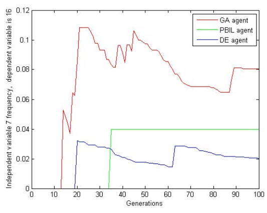
Figure 2 plots the perceptual selectivity metric analysis results of three agents using data set 2 considering DV x_{16}. This analysis is also executed along generation dimension. We calculated the frequency of different IVs of DV x_{16} in functions found accumulated over generations. The result shows much more variations than those in Figure 1. In Figure 2 (a), IV x_5 does not draw attention of PBIL agent throughout all generations, while the GA agent focuses on this variable in the initial few generations, DE agent considers it in late generations. In contrast, in Figure 2 (b), PBIL agent uses IV x_6 more frequently to generate hypotheses than GA and DE agent do. In Figure 2 (c) IV x_7 is most interesting to GA agent. IV x_8 gradually draws attentions of both GA agent and DE agent,shown in Figure 2 (d). The feature that three agents put different level of attention on the same IV through generations can also be observed from Figure 2 (e) and (f). The variations of frequency values suggest that PBIL agent, DE agent and GA agent have different perceptual selectivity when generating hypotheses in each time step. Similar to Figure 1, we notice here, the PBIL agent shows stable attention on certain IVs after about 40 generations. This again sug-



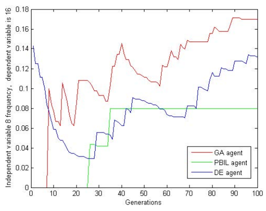
(a) Frequency of IV x_5, with DV x_{16}, in each Generation



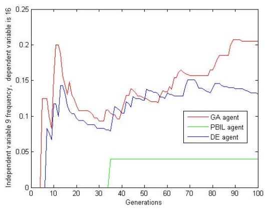
(b) Frequency of IV x_6, with DV x_{16}, in each Generation



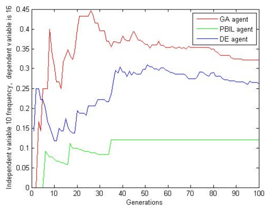
(c) Frequency of IV x_7, with DV x_{16}, in each Generation



(d) Frequency of IV x_8, with DV x_{16}, in each Generation



(e) Frequency of IV x_9, with DV x_{16}, in each Generation



(f) Frequency of IV x_{10}, with DV x_{16}, in each Generation

Fig. 2. Average frequency of independent variables along generation

gests that PBIL agent in current implementation doesn't keep diversity as well as other two agents.

Applying the proposed attention focus metrics to the prototype hypothesis generation agents does capture variations in the performances of different agents. These variations of attention attribute to the stochastic nature of underlying EC algorithms, however, further work is required to represent more complex hypotheses including more hypothesis components for the metrics to characterise. This and the other direction for future work is considered in next section.

6 Conclusion

This paper has proposed two attention focus metrics for the behaviour analysis of autonomous data mining hypothesis generation agents, and we applied these metrics to three prototype hypothesis generation algorithms. The experimental results suggest that the proposed metrics can capture variations of attention focus of different hypothesis generation algorithms. This study is our first step in developing attention focus metrics for autonomous hypothesis generation agents. Future work will proceed in two directions. The metrics for attention focus will be further extended to characterise more aspects of the search behaviours of autonomous hypothesis generation agents. In addition, hypothesis generation mechanisms will be refined to generate more complex hypotheses.

References

1. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery in Databases. AI Magazine 17, 37–54 (1996)
2. Bongard, J., Zykov, V., Lipson, H.: Resilient Machines Through Continuous Self-Modelling. Science 314, 1118–1121 (2006)
3. King, R.D., Whelan, K.E., et al.: Functional Genomic Hypothesis Generation and Experimentation by a Robot Scientist. Nature 427, 247–251 (2004)
4. Moss, L., Sleeman, D., et al.: Ontology-driven Hypothesis Generation to explain Anomalous Patient Responses to Treatment. Knowledge-Based Systems 23, 309–315 (2010)
5. Foner, L.N., Maes, P.: Paying Attention to What's Important: Using Focus Attention to Improve Unsurpervised Learning. In: Proceedings of The Third International Conference on the Simulation of Adaptive Behaviour, pp. 1–20 (1994)
6. Oudeyer, P.Y., Kaplan, F., Hafner, V.V.: Intrinsic Motivation Systems for Autonomous Mental Development. IEEE Transactions on Evolutionary Computation 11, 265–286 (2007)
7. Graziano, V., Glasmachers, T., et al.: Artificial Curiosity for Autonomous Space Exploration. Acta Futura, 1–16 (2011)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)
9. Baluja, S.: Population-Based Incremental Learning: A Method for Integrating Genetic Search based Function Optimization and Competitive Learning. Studies in Fuzziness and Soft Computing 170, 105–129 (1994)
10. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimazation. Springer (2005)
11. Abbass, H.A.: The Self-Adaptive Pareto Differential Evolution Algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation, pp. 831–836 (2002)

Emergent Self Organizing Maps for Text Cluster Visualization by Incorporating Ontology Based Descriptors

Kusum Kumari Bharti and Pramod Kumar Singh

Computational Intelligence and Data Mining Research Lab
ABV-Indian Institute of Information Technology and Management Gwalior
Morena Link Road, Gwalior, India
kkusum.bharti@gmail.com, pksingh@iiitm.ac.in

Abstract. Despite various advantages of traditional feature vector model for document representation, the well-known inherent deficiency in this model is “sovereign term assumption”. This deficiency makes it impossible to identify syntactically different but semantically related terms. In this paper, we demonstrate the use of semantic similarity measure for quantifying the relationship between related terms. Identifying such relationships help in reducing the difference between related documents. In this work, we use only noun terms for enriching the representation model. The natural visualization of clusters is investigated in this study using Emergent Self Organizing Map (ESOM). Experimental results show that incorporation of semantic relationship enhances the accuracy of clustering results.

Keywords: Emergent Self-Organized Map (ESOM), feature vector model, semantic similarity measure.

1 Introduction

Text clustering is a more precise subsection of unsupervised learning (also referred as clustering). It is also known as document clustering. It is the process of automatically grouping related documents based upon document’s intrinsic characteristics. It finds its application in real-life applications in various areas, e.g., news aggregation, information retrieval, search engine, topological hierarchy creation.

In current scenario, most documents follow non-linear data structure, i.e., documents deal with large number of diverse areas. For example, “Feature selection algorithm based on particle swarm optimization to improve the performance of document clustering” deals with three different disciplines “Feature selection”, “particle swarm optimization” and “document clustering”. Classical corpuse like the well known 20 Newsgroups[1] and Reuters-21578[2] dataset deal with a large

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/
[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/

number of diverse topics making the vocabulary very large. In this paper, we use Reuters-21578 dataset for our experimentation.

Vector Space Model (VSM) is a traditional method for representation of documents, where rows correspond to documents and columns corresponds to unique terms/descriptors, or vice versa. Many variations of VSM have been proposed in literature [1]. They differ in what they consider as terms or features. Traditional VSM model uses Bag-of-Words (BOW) representation. The main drawback of this representation is destruction of semantic relations between words. For example, stable phrases such as "Bill Gates" or "White House" are represented in the BOW as separated words so their actual meaning is lost. Given a BOW of a document in which words "White" and "House" occur, one can suggest that the document is somewhat related to color of a house, but not about official residence and principal workplace of the president of the United States. However, given a document representation that contains a phrase "White House", the reader will hardly make a mistake about the topic of discussion [16].

Traditional VSM model performs strict term based comparison in order to identify related terms and ignores semantic relationship between syntactically different but semantically related terms. For example, "sports" in one document and "athletics" in another document does not contribute to the similarity measure among these two documents. In dealing with inter-disciplinary structure of documents, identifying complex semantic along with reducing dimensions of VSM is striking and open research problem in the area of document clustering.

Self Organizing Map (SOM) [3] is an effective tool for projecting high dimensional non-linear data set onto 1 or 2 dimensional display. Thus, SOM helps in efficiently visualizing inherent non-linear relationships among documents. The most significant benefit of this procedure is that it reduces computational load considerably and also makes it possible to cluster large dataset. These properties make it unique, especially, for dimensionality reduction. These striking features of SOM have provoked researchers to use SOM for text clustering and visualization (e.g., [2]). In ESOM [4] the cluster boundaries are "indistinct", i.e., the degree of separation between regions of the map (clusters) are depicted by "gradients". This obviates the need to assign inter-disciplinary documents to more than one cluster. In this paper, we use ESOM for creating clusters of documents and use Wu & plamer [7] semantic similarity measure for identifying complex semantic of terms.

Rest of the paper is organized as follows. Section 2 provides an overview of the relevant literature to understand and present our study. In section 3, we describe our proposed document representation methodology based on frequent legitimated terms and semantically enrich reweighting scheme. Section 4 discusses the experiments and results. Finally, we conclude in section 5 with a hint for possible future research in this direction.

2 Literature Review

There are three main approaches to incorporate terms into the document representation: the first applies unigram only for document representation, the second includes bigram only excluding unigrams from the representation and the third includes incorporation of bigrams together with unigrams.

Traditionally documents are represented in a form of VSM, which uses BOW concept. In this representation model, every dimension corresponds to a single term. It sometime causes destruction of actual context of term usage. To weed out this problem, in early 90's Bag of Phrases was proposed as a competitive representation. Since then, dozens of articles have been published on this topic and some of them report significant improvement in text categorization, e.g., [15]. However, this improvement has been obtained only on rarely used datasets [16]. From experimental results it turn out that BOW representation is powerful enough and probably the results cannot be improved by replacing the BOW representation but only by extending it. In this paper, we use unigrams and unigrams incorporated with bigrams for document representation.

Next issue with traditional representation model is independent term assumption. This model cannot present semantic well as all the features are considered independent from each other. However, many terms which are syntactically different may be semantically related. For example, "dog" and "canine" have the same meaning, but in traditional representation model they will be treated as two independent dimensions. To identify this complex semantic one of the initial paper based on the concept of background knowledge has been proposed by Hotho et al. [14]. In this, the authors introduced a concept of using WordNet[3] for identifying semantic relationship between related terms. For identifying this relationship, they used synonymy and hypernymy relationships. Their experimental results show that incorporation of background knowledge improves the accuracy of clustering results. Later, continuing this concept Zheng et al. [12] used hypernymy, hyponymy, holonymy and meronymy relationships for identifying related terms. Recently, some authors, e.g., [11], [6], [13] analyzed that identification of semantic similarity between different terms are not self sufficient for reflecting the actual similarity between related documents. For example, documents containing "car" and "automobile" are more similar than "student" and "person". Therefore, more weight should be assigned to car and automobile than student and person. For incorporating this information Jing et al. [11], Fodeh et al. [6] and Sridevi et al. [13] embed the semantic relationship information directly in the weights of the corresponding words. They incorporated this information by re-adjusting the weight value through similarity measure. Jing et al. [11] devised a new similarity measure and Fodeh et al. [6] and Sridevi et al. [13] used existing path based Wu & Plamer semantic similarity measure for quantifying the similarity. In this paper, we extend the term weighting scheme presented by Sridevi et al. [13] to a weighted term weighting scheme. We assign 75% weightage to TF-IDF and 25% weightage to Wu & Plamer semantic similarity measure.

The partitional clustering algorithm are most widely used algorithm in the literature, e.g., [5], [10] for document clustering. The main problem with these clustering methods is that accuracy of the result depends on the number of clusters. It is possible that similar documents may get clustered in different

[3] http://wordnet.princeton.edu/

clusters if the number of clusters are too large and different documents may get clustered in the same cluster if number of clusters are too small. In this article, we use ESOM clustering, which automatically clusters related documents without explicitly defining the number of clusters and it is also able to create macro clusters at higher level and the micro clusters at lower level revealing great deal of structure.

3 Proposed Methodology

In this section, we present a detailed description of our proposed methodology. Usually, large percentage of a document contains uninformative terms. These terms unnecessary increase the number of dimensions in the representational model and reduce the accuracy. Therefore, documents must be preprocessed in order to remove these non-descriptive terms. The preprocessing steps include stopwords removal, stemming, and noun extraction. Stop words[4] are common words, e.g., a, an, the, who, be, which are necessarily required to be removed as they carry no weightage for clustering but make a substantial part of document. Stemming[5] converts the morphological/derivationally related words into single root form. Finally, we tokenize documents into unigrams (i.e., single terms)and bigrams (i.e., a pair of consecutive terms).

3.1 Unigram and Unigram-Bigram Generation

Unigram Generation: After tokenizing the documents into single terms, we apply frequency based constraint in order to select frequent single terms. Single terms, which appear in more than 4 and less than 200 documents have been considered as extremely infrequent and extremely frequent terms will not help in discriminating the documents. Rather they unnecessary increase the dimension of representation model. It generated 1959 frequent single terms [9] (hereinafter, referred as Uni).

Unigram-Bigram Generation: Here, we consider both unigrams and bigrams. The generation of single terms is same as above, however, we consider a pair of consecutive terms that appear in more than 2 documents. It generated 2993 frequent unigram-bigram [9] (hereinafter, referred as Uni_Bgrm).

Next we use the WordNet ontology to extract legitimate terms.

3.2 Legitimate Terms Extraction

WordNet[3] is a large semantic lexical database for the English language created and maintained by Princeton University by Professor George A. Miller since

[4] http://www.fromzerotoseo.com/stopwords-remove/

[5] http://www.comp.lancs.ac.uk/computing/research/stemming/general/porter.htm

1985. We consider a term as noun if its entry is present in the WordNet as noun. In this way, we extract noun terms without any overhead of parsing sentences in given sets of document. Unigrams and Bigrams both are given as input to the WordNet ontology to extract noun terms.

Legitimate Unigram Generation: After selecting legitimate terms, next, we extract frequent legitimate term for removing rarely occurring terms. For selecting frequent legitimate term we use the same concept as we used for the unigram generation. It resulted in 968 legitimate frequent unigrams (hereinafter, referred as UniS).

Legitimate Unigrams-Bigrams Generation: Here, we consider both legitimated unigrams and bigrams. For selecting frequent legitimate unigrams-bigrams, we use the same concept as mentioned in the unigram-bigram generation phase. It resulted in 1240 frequent legitimate unigram-bigram terms (hereinafter, referred as UniS_BgrmS).

Next step is term weighting. It is one of the most important step of text clustering algorithm. It is used to show the significance of terms with respect to the documents. Numerous terms weighting schemes have been proposed over the years. TF-IDF and its variant are commonly used for term weighting. The term weighting scheme used in our work is presented next. Weight vector for Uni and Uni_Bgrm are calculated using Equ. 1 and weight vector for UniS and UniS_BgrmS are calculated using Equ. 2.

3.3 Term Weighting

The vectors representing the documents are constructed based on the frequency of occurrence of the legitimate terms with respect to the documents. The raw frequencies are transformed to a variant of the usual TF-IDF measure.

$$tf\_idf(i,d) = (\sqrt{[wf_{id}]})\ln(\frac{N}{df_i}) \qquad if\ wf_id \geq 1; \quad 0\ otherwise \tag{1}$$

where *tf_idf(i,d)* is the IDF of $\imath^{(th)}$ word in the $\jmath^{(th)}$ document, f_{id} is the word frequency (i.e., frequency of the the $\imath^{(th)}$ word in $\jmath^{(th)}$ document), N is the total number of documents in the corpus and df_i is the document frequency of $\imath^{(th)}$ word (i.e., the number of documents in the corpus that include the $\imath^{(th)}$ word). The $(\sqrt{[wf_{id}]})$ is introduced in place of the log transform term of word frequencies $(1 + \ln(wf_{id})$ in the usual TF-IDF expression to reduce the "dampening" effect of the log function on the word frequencies.

The term reweighing scheme based on semantic measure is calculated based on Wu & Plamer and as given in Equ 2.

$$w_{iN} = 0.75 \times tf\_idf_i \times 0.25 \times \left[\sum_{t=1, Sim_{wu\&pl}(w_{iN}, w_{jN}) \geq \alpha}^{m} Sim_{wu\&pl}(w_{iN}, w_{jN}) \times w_{jN} \right] \tag{2}$$

where m is the total number of features, w_i stands for term i and α is the minimum similarity threshold value.

$$Sim_{wu\&Pl}(w_i, w_j) = \frac{2 \times N}{N_1 + N_2} \tag{3}$$

Where tf_idf_i stands for the concept term i and w_i stands for concept term weighted reweighting, N is depth of Least Common Superconcept (LCS) from root, N_1 is the depth of w_i and N_2 is the depth of w_j.

4 Experimental Setup

The experiments are conducted on Retures-21578 dataset using Databionic Esom tool[6] on a system with core $i5$ processor and 2 GB RAM in Windows 7 environment. We use two performance measures, trustworthiness and preservation, to judge the performance of our proposed model.

Trustworthiness of the Visualization. Trustworthiness [8] determines, whether two visually similar documents are really similar. Here, N is the total number of data points, $U_k(x_i)$ is the set of those data points that are in the neighborhood of point x_i in the visualization display but not in the original data space, and $r(x_i, x_j)$ is the rank of the data point x_j in the ordering according to distance from x_i in the original data space. Trustworthiness of the visualization, $T(k)$, is defined by Equ. 4

$$T(k) = 1 - \frac{2}{(Nk(2N - 3k - 1))} \sum_{i=1}^{N} \sum_{x_j \in U_k(x_i)} \left[\Big((r(x_i, x_j) - k) \Big) \right] \tag{4}$$

Preservation of the Original Neighbourhoods. Preservation [8] determines whether two really similar documents are visually similar also. Here, N is the total number of data points, $\widehat{r}(x_i, x_j)$ is the rank of the data point x_j in the ordering according to distance from x_i in the visualization display, and $V_k(x_i)$ is the set of those data points that are in the neighborhood of the data point x_i in the original space but not in the visualization. Preservation $P(k)$, is defined by Equ. 5

$$P(k) = 1 - \frac{2}{(Nk(2N - 3k - 1))} \sum_{i=1}^{N} \sum_{x_j \in V_k(x_i)} \left[\Big((\widehat{r}(x_i, x_j) - k) \Big) \right] \tag{5}$$

Where k is the sets of nearest neighbour of observed datapoint.

[6] http://databionic-esom.sourceforge.net/

4.1 Results and Discussion

For demonstrating the effectiveness of proposed model, we train the ESOM with UniS, Uni, UniS_BgrmS and Uni_Bgrm one by one and then perform their comparative analysis.

First we train the ESOM with four different feature vector models, i.e., Uni, UniS, Uni_Bgrm, and UniS_BgrmS. Summarization of all four feature vector models are given in Table. 1.

Table 1. Summarization of Feature Vector Models

Model Name	*Term*	*WordNet*	*Frequency Constraint*	*Term Weighting*
Uni	Single Term	No	More than 4 and less than 200 documents	Equation 1
UniS	Single Term	Yes	More than 4 and less than 200 documents	Equation 2
Uni_Bgrm	Single and Bigram Terms	No	More than 4 and less than 200 documents for single term, more than 2 documents for bigram term	Equation 1
UniS_BgrmS	Single and Bigram Terms	Yes	More than 4 and less than 200 documents for single term, more than 2 documents for bigram term	Equation 2

After training the ESOM with these feature vector models one by one, we compute the trustworthiness and preservation of the models based on varying number of neighbourhoods, k. Here, k varies from 10 to 60.

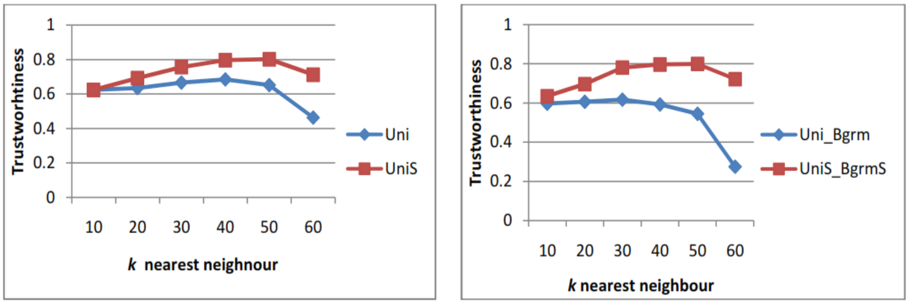


Fig. 1. Performance comparison on the basis of Trustworthiness

From Fig. 1, we observe that UniS and UniS_BgrmS have higher trustworthiness than Uni and Uni_Bgrm based representation respectively. This improvement confirms that identification of semantic relationship between different terms help in reducing the difference between related documents.

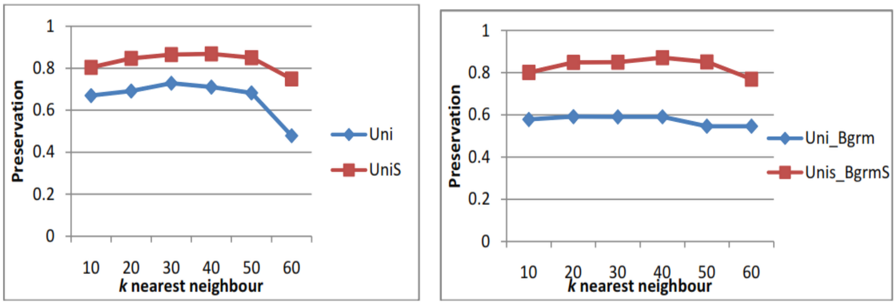


Fig. 2. Performance comparison on the basis of Preservation

From Fig. 2, we observe that the incorporation of semantic relationship not only improves the trustworthiness of model but also helps in improving the preservation of the map. We achieve these improvements because we identify syntactically different but semantically related terms thus, successfully reduces the difference between related documents. In other words, it increases the chance to place similar documents at same place.

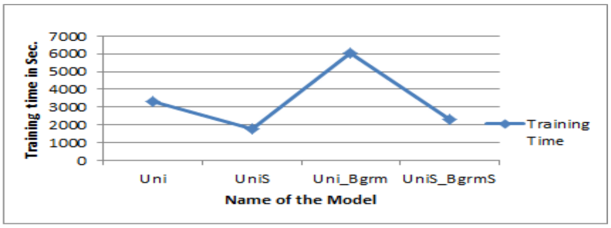


Fig. 3. Performance comparison on the basis of training time

Fig. 3 shows that our proposed approach not only improves the accuracy of the ESOM map but also reduces the ESOM training time. A possible reason for reduced computation time is the number of features in case of legitimate terms (i.e., UniS and UniS_BgrmS) is considerably less than frequent terms (i.e., Uni and Uni_Bgrm).

5 Conclusion and Future Work

In this paper, we present new documents representation method that incorporate semantic relationship information in representing text document. Instead of accumulating additional concepts to the traditional term based representation model, we readjusted the term weight according to the Wu & Plamer semantic similarity measure. Instead of using all terms for document representation, we used noun terms only. The experimental results show that the new weighted term weighting scheme is more effective and efficient in improving the performance of ESOM than traditional term weighting scheme (i.e., variant of TF-IDF).

Though we have used Wu & Plamer semantic similarity measure to enrich the representation model, other similarity measures may also be used. We use frequency based constraints to identify the descriptive terms, though other methods are also available in the literature, e.g., mutual information, chi-square, document frequency and term variance. In addition, we can use other media like Wikipedia to identify the semantic similarity between different terms.

References

1. Kalogeratos, A., Likas, A.: Text document clustering using global term context vectors. In: Knowledge Information System, vol. 31, no. 3, pp. 1-20 (2012)
2. Yen, G.G., Wu, Z.: A Self-Organizing Map Based Approach for Document Clustering and Visualization. In: International Joint Conference on Neural Networks, pp. 3279–3286 (2006)
3. Kohonen, T.: Self Organizing Maps, 3rd edn. Springer, Berlin (2001)
4. Ultsch, A., Moerchen, F.: ESOM-Maps: Tools for clustering, visualization, and classification with Emergent SOM. Technical Report No. 46, Dept. of Mathematics and Computer Science, University of Marburg (2005)
5. Thangamani, M., Thangaraj, P.: Ontology Based Fuzzy Document Clustering Scheme. In: Modern Applied Science, pp. 148–156 (2010)
6. Fodeh, S., Punch, B., Tan, P.N.: On ontology-driven doucment clusteirng using core semantic features. Know Inf. Syst. 28(20), 395–421 (2011)
7. Wu, Z., Plamer, M.: Verb Semantics And Lexical Selection. In: Proceeding of the 32nd Annual Meeting of the Association for Computational Linguistics, pp. 133–138 (1994)
8. Kaski, S., Oja, J.N.M., Venna, J., Toronen, P., Castren, E.: Trustworthiness and metrics in visualizing similarity of gene expression. BMC Bioinformatics 4(48), 48–61 (2003)
9. Singh, P.K., Machavolu, M., Bharti, K., Suda, R.: Analysis of Text Cluster Visualization in Emergent Self Organizing Maps Using Unigrams and Its Variations after Introducing Bigrams. In: Deep, K., Nagar, A., Pant, M., Bansal, J.C. (eds.) Proceedings of the International Conference on SocProS 2011. AISC, vol. 131, pp. 967–978. Springer, Heidelberg (2012)
10. Hu, G., Zhou, S.: Towards effective document clustering: A constrained K-means based approach. Information Processing & Management 44(4), 1397–1409 (2008)
11. Jing, L., Ng, M., Huang, J.: Knowledge-based vector space model for text clustering. Knowledge and Information Systems 25(1), 37–55 (2009)
12. Zheng, H.T., Kang, B., Kim, H.: Exploiting noun phrases and semantic relationships for text document clustering. Information Science 179(13), 2249–2262 (2009)

13. Sridevi, U., Nagaveni, N.: Semantically enhanced document clustering based on pso algorithm. European Journal of Scientific Research 57(3), 485–493 (2011)
14. Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. In: 3rd IEEE International Conference on Data Mining, pp. 541–544 (2003)
15. Mladeni, C., Grobelnik, M.: Word sequences as features in text-learning. In: Proceedings of the Seventh Electrotechnical and Computer Science Conference, pp. 145–148 (1998)
16. Bekkerman, R., Allan, J.: Using Bigrams in Text Categorization. Technical Report IR-408 (2004)

Online Handwriting Recognition Using Multi Convolution Neural Networks

Dũng Việt Phạm

Computer Network Centre, Vietnam Maritime University, 484 Lach Tray st., Ngo Quyen dist., Hai Phong city, Vietnam
dungpv@vimaru.edu.vn

Abstract. This paper presents a library written by C# language for the online handwriting recognition system using UNIPEN-online handwritten training set. The recognition engine based on convolution neural networks and yields recognition rates to 99% to MNIST training set, 97% to UNIPEN's digit training set (1a), 89% to a collection of 44022 capital letters and digits (1a,1b) and 89% to lower case letters (1c). These networks are combined to create a larger system which can recognize 62 English characters and digits. A proposed handwriting segmentation algorithm is carried out which can extract sentences, words and characters from handwritten text. The characters then are given as the input to the network.

Keywords: artificial intelligent, convolution, neural network, UNIPEN, pattern recognition.

1 Introduction

Originated in late 1950's, Artificial neural networks (ANN) did not gain much popularity until 1980s when a computer boom era. Today, ANNs are mostly used for solution of complex real world problems. This paper provides brief information of one very specific neural network (a convolutional neural network) built for one very specific purpose (to recognize handwritten digits and letters). The author also created a library for the neural network written by C# language which has shown the best performance on handwriting recognition task (MNIST and UNIPEN) using two essential techniques: elastic distortion that vastly expanded the size of the training set and convolution neural network. By using a network prototype in the library, the recognition system can create, load multi different neural networks on runtime. Furthermore, the system can combine several component networks to recognize a larger pattern dataset.

2 Convolution Neural Network (CNN)

The ability of multi-layer neural networks trained with gradient descent to learn complex, high-dimentional, non-linear mappings from large collections of examples make them obvious candidates for image recognition tasks. In the traditional model of pattern recognition, a hand-designed feature extractor gathers relevant information from input and eliminates irrelevant variabilities. A trainer classifier (normally, a standard, fully-connected multi-layer neural network can be used as a classifier) then categorizes the resulting feature vectors into classes. However, it could have some problems which should influent to the recognition results. The convolution neural network solves this shortcoming of traditional one to achieve the best performance on pattern recognition task.

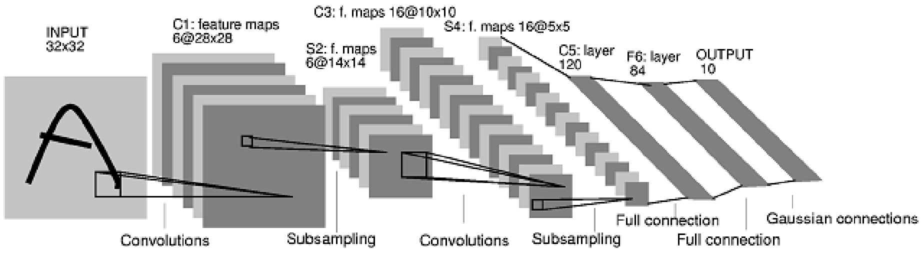


Fig. 1. A Typical Convolutional Neural Network (LeNET 5)[1]

The CNNs is a special form of multi-layer neural network. Like other networks, CNNs are trained by back propagation algorithms. The difference is inside their architecture. The convolutional network combines three architectural ideas to ensure some degree of shift, scale, and distortion invariance: local receptive field, shared weights (or weight replication) spatial or temporal sub-sampling. They have been designed especially to recognize patterns directly from digital images with the minimum of pre-processing operations. The preprocessing and classification modules are in a single integrated scheme. The architecture details of CNN have been described comprehensively in articles of Dr. Yahn LeCun and Dr. Patrice Simard [1],[3].

The typical convolutional neural network for handwritten digit recognition is shown in figure. 1. It consists a set of several layers. The input layer is of size 32 x32 and receives the gray-level image containing the digit to recognize. The pixel intensities are normalized between −1 and +1. The first hidden layer C1 consists six feature maps each having 25 weights, constituting a 5x5 trainable kernel, and a bias. The values of the feature map are computed by convolving the input layer with respective kernel and applying an activation function to get the results. All values of the feature map are constrained to share the same trainable kernel or the same weights values. Because of the border effects, the feature maps' size is 28x28, smaller than the input layer.

Each convolution layer is followed by a sub-sampling layer which reduces the dimension of the respective convolution layer's feature maps by factor two. Hence the sub-sampling maps of the hidden layer S2 are of size 14x14. Similarly, layer C3 has 16 convolution maps of size 10x10 and layer S4 has 16 sub-sampling maps of size 5x5. The functions are implemented exactly as same as the layer C1 and S2 perform. The S4 layer's feature maps are of size 5x5 which is too small for a third convolution layer. The C1 to S4 layers of this neural network can be viewed as a trainable feature extractor. Then, a trainable classifier is added to the feature extractor, in the form of 3 fully connected layers (a universal classifier).

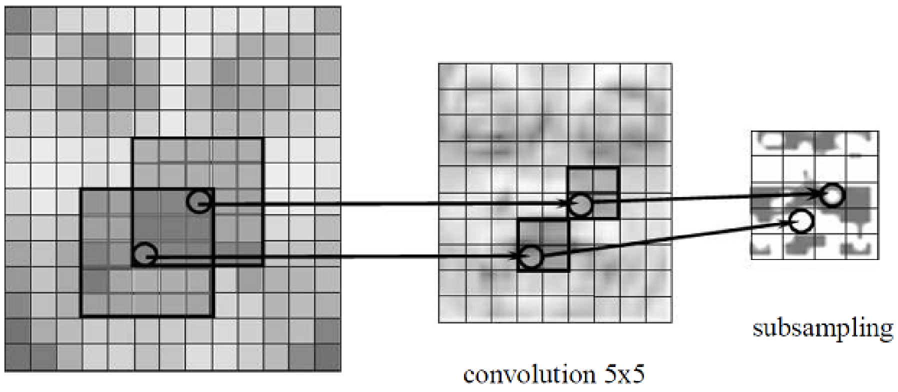
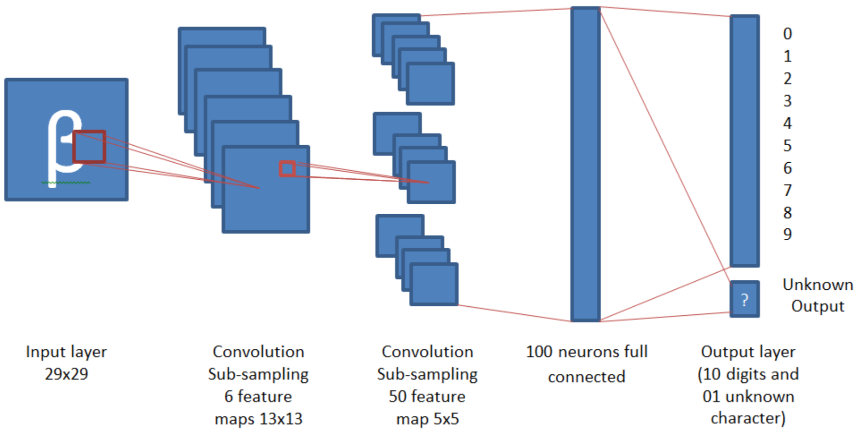


Fig. 2. An input image followed by a feature map performing a 5 × 5 convolution and a 2 × 2 sub-sampling map

Dr. Partrice Simard in his article "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis," [3] presented a different model of CNN for handwritten digit recognition which integrated convolution and sub-sampling processes in to a single layer. This model extracts simple feature maps at a higher resolution, and then converts them into more complex feature maps at a coarser resolution by sub-sampling a layer by a factor two. The width of the trainable kernel is chosen be centered on a unit (odd size), to have sufficient overlap to not lose information (3 would be too small with only one unit overlap), but yet to not have redundant computation (7 would be too large, with 5 units or over 70% overlap). Padding the input (making it larger so that there are feature units centered on the border) does not improve performance significantly. With no padding, a sub-sampling of two, and a trainable kernel of size 5x5, each convolution layer reduces the feature map size from n to (n-3)/2. Since the initial MNIST input using in this model is of size 28x28, the nearest value which generates an integer size after 2 layers of convolution is 29x29. After 2 layers of convolution, the feature of size 5x5 is too small for a third layer of convolution. The first two layers of this neural network can be viewed as a trainable feature extractor. Then, a trainable classifier is added to the feature extractor, in the form of 2 fully connected layers (a universal classifier).

Back Propagation

Back propagation is the process that updates the change in the weights for each layer, which starts with the last layer and moves backwards through the layers until the first layer is reached. Standard back propagation does not need to be used in the network library because of slow convergence. Instead, the technique called “Stochastic diagonal Levenberg-Marquardt method”, which was proposed by Dr. LeCun in his article "Efficient BackProp” [2], has been applied.



0
1
2
3
4
5
6
7
8
9
?
Unknown Output
Input layer 29x29
Convolution Sub-sampling 6 feature maps 13x13
Convolution Sub-sampling 50 feature map 5x5
100 neurons full connected
Output layer (10 digits and 01 unknown character)

Fig. 3. A convolution network based on Dr. Partrice Simard's model

3 The UNIPEN Trainset

In a large collaborative effort, a wide number of research institutes and industry have generated the UNIPEN standard and database [5]. Originally hosted by NIST, the data was divided into two distributions, dubbed the training set (train_r01_v07 set) and devset. Since 1999, the International UNIPEN Foundation (iUF) hosts the data, with the goal to safeguard the distribution of the training set and to promote the use of online handwriting in research and applications.

Due to UNIPEN training set is collection of particular datasets from different research institutes, these datasets are decomposed using some specific procedure. In order to be able to compare my system's recognition results to published researches, the UNIPEN training set is used as training input to my recognizer. However, my approach is a little bit different; some general points in the structure of these datasets have been found to create a procedure which can decompose all datasets in the training set correctly in most cases.

The UNIPEN format is described in [5],[6],[15]. The format of a UNIPEN data file has KEYWORDS which are divided to several groups like: *Mandatory declarations, Data documentation, Alphabet, Lexicon, Data layout, Unit system, Pen trajectory¸ Data annotations*. In order to get the information and categorize these keywords, a collection of classes based on the above groups have been created which can help the system to get and categorize all necessary information from data file.

4 Image Pre-processing and Segmentation

Segmentation is an important step to pattern recognition system. Normally, projection techniques are applied to separate lines, words, and characters in a text image. However, it will be difficult if characters are organized in confusion. Therefore, a new algorithm has been developed to solve this issue.

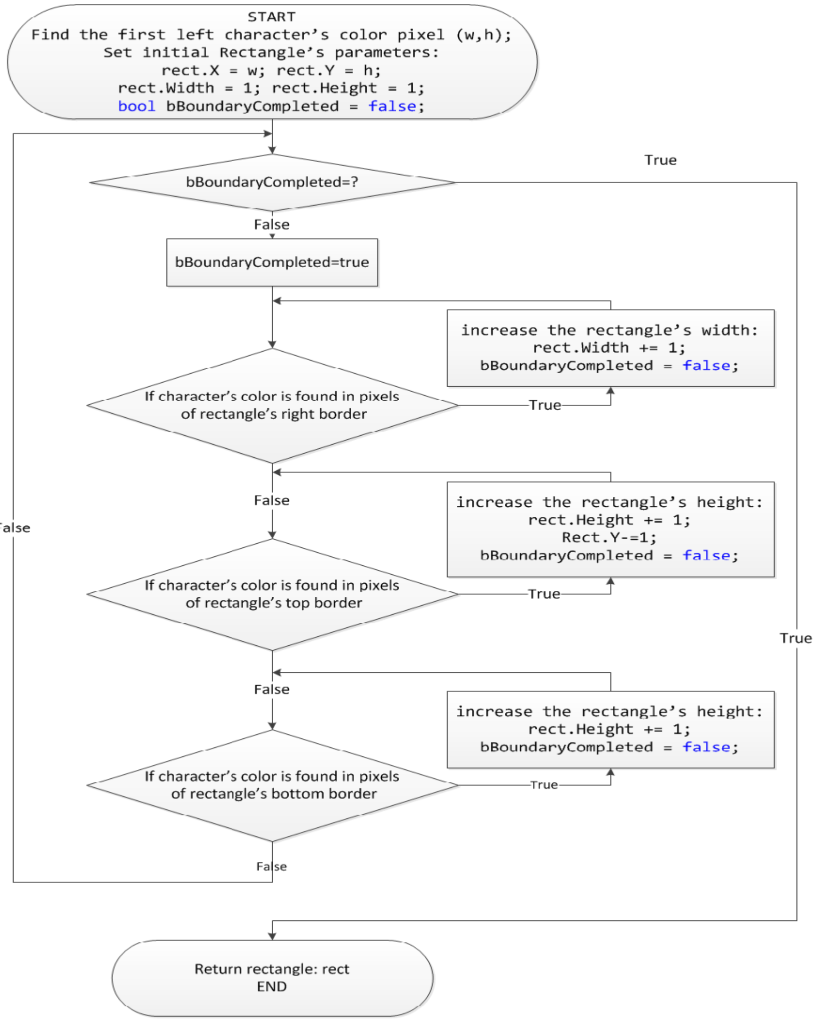

START
Find the first left character's color pixel (w,h);
Set initial Rectangle's parameters:
rect.X = w; rect.Y = h;
rect.Width = 1; rect.Height = 1;
bool bBoundaryCompleted = false;
bBoundaryCompleted=?
True
False
bBoundaryCompleted=true
If character's color is found in pixels of rectangle's right border
True
increase the rectangle's width:
rect.Width += 1;
bBoundaryCompleted = false;
False
If character's color is found in pixels of rectangle's top border
True
increase the rectangle's height:
rect.Height += 1;
Rect.Y-=1;
bBoundaryCompleted = false;
False
If character's color is found in pixels of rectangle's bottom border
True
increase the rectangle's height:
rect.Height += 1;
bBoundaryCompleted = false;
False
False
True
Return rectangle: rect
END

Fig. 4. New algorithm for getting isolated character's rectangle boundary.

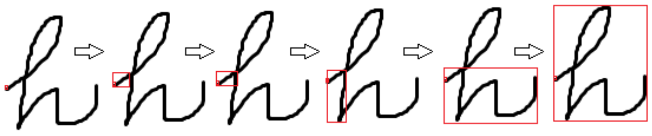


Fig. 5. Steps of isolated character segmentation

The figure 5 presents a sample of applying the above algorithm to a hand written character. Getting character's rectangle boundary is started from the first left pixel of the character. The boundary is expanded step by step from left to right, from top to bottom until the boundary can wrap the character. A similar algorithm can be applied to get the character's boundary from the topmost pixel.

By changing horizontal and vertical steps, the system can get not only isolated characters but also words or sentences without changing algorithm.

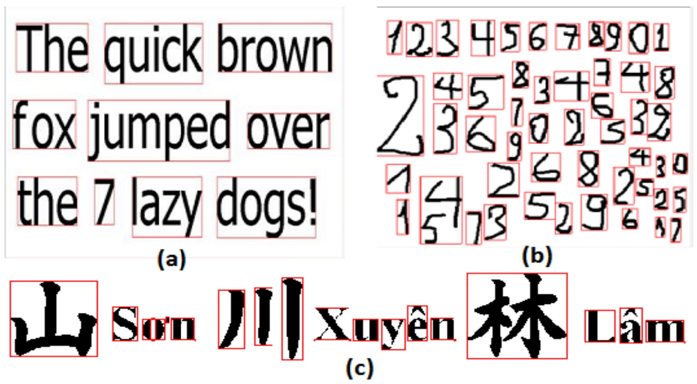


Fig. 6. Samples of word segmentation and isolated character segmentation

Using this technique together with other well-known segmentation methods can help the system to recognize characters better in complex text images.

5 Recognition System Using Multi Neural Networks

The recognition results of the convolution network are really high to small patterns collection such as digit, capital letters or lower case letters etc. However, when we want to create a larger neural network which can recognize a bigger collection like digit and English letters (62 characters) for example, the problems begin appear. Finding an optimized and large enough network becomes more difficult, training network by large input patterns takes much longer time. Convergent speech of the network is slower and especially, the accuracy rate is significant decrease because bigger bad written characters, many similar and confusable characters etc. Furthermore, assuming we can create a good enough network which can recognize accurately English characters but it certainly cannot recognize properly a special character outsize its outputs set (a Russian or Chinese character) because it does not have expansion capacity. Therefore, creating a unique network for very large patterns classifier is very difficult and may be impossible.

The proposed solution to the above problems is instead of using a unique big network we can use multi smaller networks which have very high recognition rate to these own output sets. Beside the official output sets (digit, letters…) these networks have an additional unknown output (unknown character). It means that if the input pattern is not recognized as a character of official outputs it will be understand as an unknown character (Figure 3).

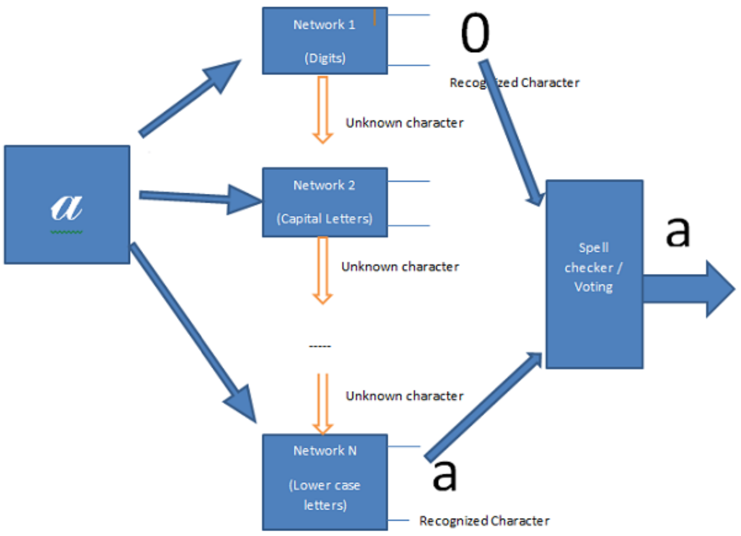


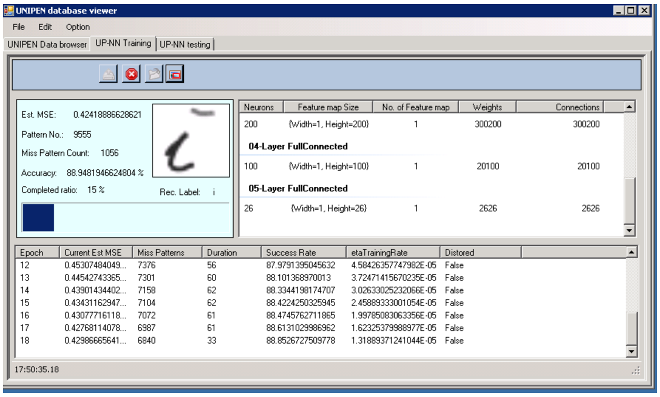
Fig. 7. A handwriting recognition system using multi neural networks

For a large pattern collection like handwritten characters, there are so many similar characters which can make not only machine but also human confuse in some cases such as: O, 0 and o; 9, 4,g,q etc. These characters can make networks misrecognize. By using an additional spellchecker/voting module at the output, the system can significant increate recognition rate. The input pattern is recognized by all component networks. These outputs (except unknown outputs) then will be set as the inputs of the spellchecker/voting module. The module will bases on previous recognized characters, internal dictionary and other factors to decide which one will be the most accurated recognized character.

This solution overcomes almost limits of the traditional model. The new system includes a several small networks which are simple for optimizing to get the best recognition results. Training these small networks takes less time than a huge network. Especially, the new model is really flexible and expandable. Depending on the requirement we can load one or more networks; we can also add new networks to the system to recognize new patterns without change or rebuilt the model. All these small networks have reusable capacity to another multi neural networks system.

6 Recognition Results

In order to evaluate the library to a handwritten recognition system, the author experiments the library on two different handwritten training sets are MNIST and UNIPEN. The results can reach to 99% accuracy rate to MNIST training set [10], 97% to UNIPEN digits, 89% to UNIPEN digits and capital letters (1a,1b) and 89% to UNIPEN lower case letters (1c)[13].



Est. MSE: 0.42418886628621
Pattern No.: 9555
Miss Pattern Count: 1056
Accuracy: 88.9481946624804 %
Completed ratio: 15 %
Rec. Label: i

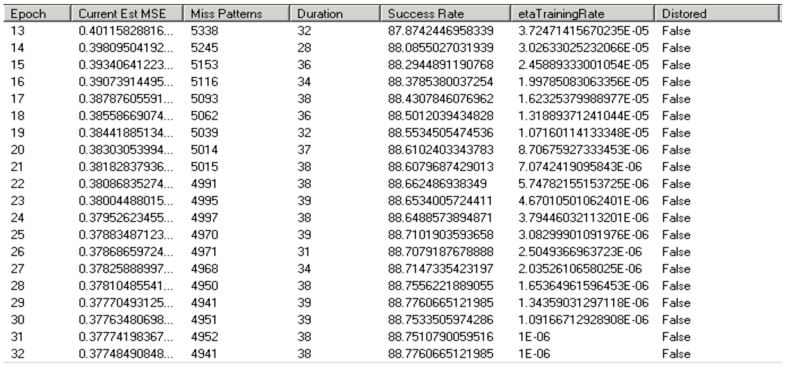
Neurons	Feature map Size	No. of Feature map	Weights	Connections
200	{Width=1, Height=200}	1	300200	300200
04-Layer FullConnected				
100	{Width=1, Height=100}	1	20100	20100
05-Layer FullConnected				
26	{Width=1, Height=26}	1	2626	2626

Epoch	Current Est MSE	Miss Patterns	Duration	Success Rate	etaTrainingRate	Distored
12	0.45307484049...	7376	56	87.9791395045632	4.58426357747982E-05	False
13	0.44542743365...	7301	60	88.101368970013	3.72471415670235E-05	False
14	0.43901434402...	7158	62	88.3344198174707	3.02633025232066E-05	False
15	0.43431162947...	7104	62	88.4224250325945	2.45889333001054E-05	False
16	0.43077716118...	7072	61	88.4745762711865	1.99785083063356E-05	False
17	0.42768114078...	6987	61	88.6131029986962	1.62325379988977E-05	False
18	0.42986665641...	6840	33	88.8526727509778	1.31889371241044E-05	False

17:50:35.18

Fig. 8. Network training interface using UNIPEN trainset (experiment in 1c)

The figure 8 is the network training interface of the demo program. The training can reach to 89% accuracy after 48 epochs to lower case letters set 1c (the first training time is 30 epochs, the second time is 18 epochs). After the first 30 epochs, the etaTrainingRate was too small which influenced to network training performance. So the network was trained again in second time with bigger initial etaTrainingRate = 0.00045.

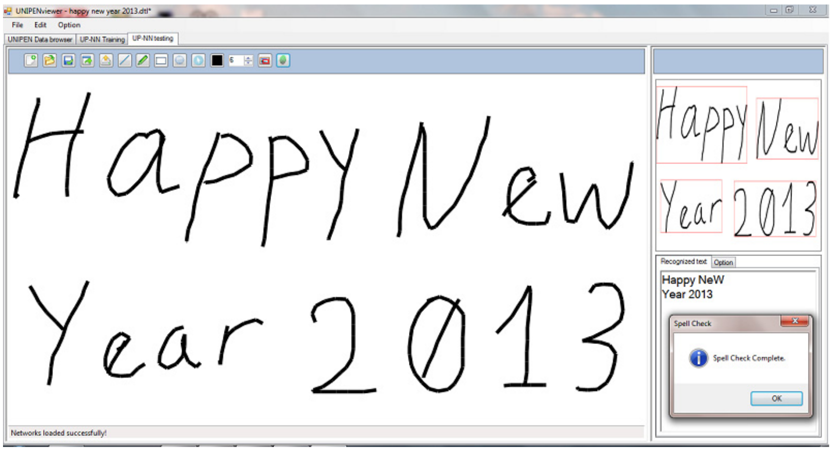


Epoch	Current Est MSE	Miss Patterns	Duration	Success Rate	etaTrainingRate	Distored
13	0.40115828816...	5338	32	87.8742446958339	3.72471415670235E-05	False
14	0.39809504192...	5245	28	88.0855027031939	3.02633025232066E-05	False
15	0.39340641223...	5153	36	88.2944891190768	2.45889333001054E-05	False
16	0.39073914495...	5116	34	88.3785380037254	1.99785083063356E-05	False
17	0.38787605591...	5093	38	88.4307846076962	1.62325379988977E-05	False
18	0.38558669074...	5062	36	88.5012039434828	1.31889371241044E-05	False
19	0.38441885134...	5039	32	88.5534505474536	1.07160114133348E-05	False
20	0.38303053994...	5014	37	88.6102403343783	8.70675927333453E-06	False
21	0.38182837936...	5015	38	88.6079687429013	7.07424190958 43E-06	False
22	0.38086835274...	4991	38	88.662486938349	5.74782155153725E-06	False
23	0.38004488015...	4995	39	88.6534005724411	4.67010501062401E-06	False
24	0.37952623455...	4997	38	88.6488573894871	3.79446032113201E-06	False
25	0.37883487123...	4970	39	88.7101903593658	3.08299901091976E-06	False
26	0.37868659724...	4971	31	88.7079187678888	2.5049366963723E-06	False
27	0.37825888997...	4968	34	88.7147335423197	2.0352610658025E-06	False
28	0.37810485541...	4950	38	88.7556221889055	1.65364961596453E-06	False
29	0.37770493125...	4941	39	88.7760665121985	1.34359031297118E-06	False
30	0.37763480698...	4951	39	88.7533505974286	1.09166712928908E-06	False
31	0.37774198367...	4952	38	88.7510790059516	1E-06	False
32	0.37748490848...	4941	38	88.7760665121985	1E-06	False

Fig. 9. Statistics of network training' parameters after 62 epochs

Figure 9 is network training's parameters statistics of the digits and capital letters recognition network (36 outputs network). By using Stochastic diagonal Levenberg-Marquardt method in back propagation process, the convergent speech of network becomes much faster than standard back propagation. After 65 epochs the accuracy rate of the network can reach to 89%.

In order to recognize a larger character set such as English characters (62 characters), a recognition system based on the model presented in figure 5 has been created. This system is a combination of three high recognition rate neural networks: digit (97%), capital letters (89%) and low case letters (89%). The system has proved its efficient recognition capacity by using an additional spell checker module.



Happy NeW
Year 2013

Spell Check Complete.

Networks loaded successfully!

Fig. 10. Mouse drawing characters recognition using multi networks [13]

All the library, demo program, source code and training results can be downloaded at [13].

7 Conclusion

The paper presented a method of handwriting recognition using artificial convolution neural network. By the combination of convolution neural network, elastic distortion technique and Stochastic diagonal Levenberg-Marquardt method, the experimental neural networks can reach to positive results. Furthermore, the proposed model using multi component neural networks also presented an ability of creating efficient and flexible recognition systems to large pattern sets such as English characters set etc. By using a spell checker and voting module at the output, the proposed system can choose the most accurated characters from high recognition rate component networks. Hence, it can get the better recognition results to a traditional one.

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
2. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient BackProp. In: Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 1524, pp. 9–50. Springer, Heidelberg (1998)
3. Simard, P.Y., Steinkraus, D., Platt, J.: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 958–962. IEEE Computer Society, Los Alamitos (2003)
4. Lauer, F., Suen, C.Y., Bloch, G.: A Trainable Feature Extractor for Handwritten Digit Recognition. Elsevier Science (February 2006)

5. Guyon, I., Schomaker, L., Plamondon, R., Liberman, R., Janet, S.: Unipen project of on-line data exchange and recognizer benchmarks. In: Proceedings of the 12th International Conference on Pattern Recognition, ICPR 1994, Jerusalem, Israel, pp. 29–33. IAPRIEEE (October 1994)
6. Vuurpijl, L., Niels, R., van Erp Nijmegen, M.: Verifying the UNIPEN devset
7. Parizeau, M., Lemieux, A., Gagné, C.: Character Recognition Experiments using Unipen Data. In: Parizeau, et al. (eds.) Proc. of ICDAR 2001, Seatle, September 10-13 (2001)
8. List of publications by Dr. Yann LeCun, `http://yann.lecun.com/exdb/publis/index.html`
9. O'Neill, M.: Neural Network for Recognition of Handwritten Digits, `http://www.codeproject.com/Articles/16650/Neural-Network-for-Recognition-of-Handwritten-Digi`
10. Dung, P.V.: Neural Network for Recognition of Handwritten Digits in C#, `http://www.codeproject.com/Articles/143059/Neural-Network-for-Recognition-of-Handwritten-Digi`
11. Dung, P.V.: Library for online handwriting recognition system using UNIPEN database, `http://www.codeproject.com/Articles/363596/Library-for-online-handwriting-recognition-system`
12. Dung, P.V.: UPV – UNIPEN online handwriting recognition database viewer control, `http://www.codeproject.com/Articles/346244/UPV-UNIPEN-online-handwriting-recognition-database`
13. Dung, P.V.: Large pattern recognition system using multi neural networks, `http://www.codeproject.com/Articles/376798/Large-pattern-recognition-system-using-multi-neura`
14. Modified NIST ("MNIST") database (11,594 KB total), `http://yann.lecun.com/exdb/mnist/index.html`
15. The UNIPEN Project, `http://unipen.nici.kun.nl/`

A Genetic Programming Approach to Hyper-Heuristic Feature Selection

Rachel Hunt[1], Kourosh Neshatian[3], and Mengjie Zhang[2]

[1] School of Mathematics, Statistics and Operations Research
[2] School of Engineering and Computer Science
Victoria University of Wellington, PO Box 600, Wellington, New Zealand
[3] Department of Computer Science and Software Engineering
University of Canterbury, Private Bag 4800, Christchurch, New Zealand
huntrach1@myvuw.ac.nz, kourosh.neshatian@canterbury.ac.nz,
mengjie.zhang@vuw.ac.nz

Abstract. Feature selection is the task of finding a subset of original features which is as small as possible yet still sufficiently describes the target concepts. Feature selection has been approached through both heuristic and meta-heuristic approaches. Hyper-heuristics are search methods for choosing or generating heuristics or components of heuristics, to solve a range of optimisation problems. This paper proposes a genetic-programming-based hyper-heuristic approach to feature selection. The proposed method evolves new heuristics using some basic components (building blocks). The evolved heuristics act as new search algorithms that can search the space of subsets of features. The classification performance (accuracy) of classifiers are improved by using small subsets of features found by evolved heuristics.

1 Introduction

Feature manipulation is a useful and often necessary process of altering the input space of a machine learning task in order to improve the learning quality and performance [1]. In many cases, some features of objects in a dataset are unnecessary and lead to performance deterioration; the sheer number of available features means that the time taken to find solutions is longer than is desired, or costly feature extraction means using more features incurs a greater cost.

A *feature* is a function which maps an object to one of its properties. In classification each example in a given dataset has values for a number of features which are used to determine which class the example belongs to.

Feature selection (FS) is the task of finding a subset of the original features which is as small as possible and yet still sufficiently describes the target concepts [2]. Unnecessary or redundant features are eliminated, reducing the dimensionality, hopefully improving the learning performance, and making the outcome (model) obtained easier to interpret by humans. FS is a combinatorial optimisation problem [3] as it concerns finding a subset from a finite set of features subject to certain constraints and objectives [4].

Combinatorial problems have been approached with heuristics, meta-heuristics and hyper-heuristics. Heuristics are problem specific techniques which search for high-quality solutions that do not guarantee optimality [5]. Example heuristics for FS are RELIEF [2], FOCUS [6], sequential forward selection [7], greedy hill climbing heuristics [8], and linear forward search [9].

Meta-heuristics are problem independent (provided the right encoding) strategies which "guide" the search process in order to find optimal, or near-optimal solutions [4]. Meta-heuristic techniques which have been used for FS include evolutionary algorithms such as genetic algorithms [10], particle swarm optimisation [11], genetic programming (GP) [1], and local search methods.

Hyper-heuristics (HH) can be thought of as heuristics to choose heuristics as they search a space of heuristics instead of searching the space of solutions directly [12]. More formally HH are search methods for choosing or generating heuristics or components of heuristics, to solve a range of optimisation problems with the aim of increasing the level of generality (robustness) of search methods [3]. In this paper we are interested in hyper-heuristics.

There is very limited work using HH for FS. To our awareness there is none using GP. Abdullah et al. [13] developed an online constructive HH approach using rough sets for attribute reduction. A roulette wheel selection mechanism was used to choose the most appropriate of four low level heuristics, which constructs the solution. Rough sets are used to evaluate dependency value through solving the attribute reduction problem. Results show that this approach produces good quality solutions in comparison to other meta-heuristic approaches and, on some datasets, outperforms other approaches.

The goal of this work is to use GP to evolve heuristics for feature selection. We are not looking for just any specific features or specific subsets of features, but expect the evolved heuristics to find good solutions (subsets of features) which are smaller than the set of all features and improve the performance on classification problems.

The remainder of this paper is organised as follows. Section 2 describes the background relating to this paper. Section 3 describes the new hyper-heuristic approach used for feature selection tasks. Section 4 describes the experiment design and Section 5 discusses the results. Section 6 concludes the paper.

2 Background

This section presents a summary of existing work in both feature selection and the use of hyper-heuristics.

2.1 Feature Selection

There are wrapper and filter based approaches to feature selection. Wrapper approaches use a learning algorithm when exploring the search space. Candidate solution evaluations are costly, as each requires a classifier to be trained and tested. The filter approach does not use any learner's feedback to evaluate a solution (they are independent of the learning algorithm). Other heuristics which are more computationally efficient are used. Filter approaches are computationally less expensive and more general than the wrapper approaches [11], but accuracy (effectiveness) results are often not as good as the wrapper approaches. In this work we will use a wrapper approach.

Sequential Forward Selection (SFS) uses a simple hill climbing search proposed by Whitney [7]. The search is initialised with the empty subset. At each step all the possible single feature additions are evaluated, and the feature which gives the best score (classification accuracy of the wrapper) is permanently added to the feature subset. SFS [7] works up towards a pre-determined subset size, however SFS has since been used so it will terminate when there is no single feature whose addition will increase the current best score [9].

Sequential Backward Selection (SBS) proposed by Marill and Green [14] also uses a simple hill climbing search. The search is initialised with the entire set of features. At each step the feature whose removal leads to the best score is permanently removed from the feature subset. SBS terminates when there is no feature whose removal will increase the current best score.

Linear forward search [9] uses a user-specified constant, k, to limit the number of features considered for forward selection. One method ranked features and then considered only the best k in forward selection. A second method considered only the next k best features at each stage of forward selection. Results showed that LFS was faster, found smaller subsets and increased the accuracy in comparison to standard forward selection on several datasets.

Xue et al. [11] proposed two wrapper-based FS approaches: single feature ranking, and binary particle swarm optimisation based feature subset ranking. Results showed that the two proposed methods could achieve better results (classification accuracy and number of features) than linear forward selection and greedy stepwise backward selection. A limitation of this approach is the relatively long evolutionary training time, thus it may not be suitable for online applications.

Neshatian et al. [15] proposed a GP-based ranking system, which ranks individual features on the strength of the relationships between the original features and the target class. A scoring system ranks the features based on the frequency of feature occurrence in good models. Results showed that the rankings provided by this system showed strong connections to the actual importance of features. A GP-based system for measuring the relevance of subsets of features to target concepts in binary classification tasks was proposed by Neshatian et al [16]. Results showed that this system can detect relevant feature subsets and can outperform other ranking methods in difficult situations.

2.2 GP Based Hyper-Heuristics

Hyper-heuristics have the "goal of automating the design and adaption of heuristic methods in order to solve hard computation search problems" [17]. Burke et al. [3] explored the suitability of GP as a HH. The advantages that GP offers include variable length encoding, and that the GP output can be an executable data structure. Humans can also easily identify the problem domain, search space, and terminal and function sets. The disadvantages include that each GP run gives a different best-of-run heuristic, so multiple GP runs are needed to establish what quality of heuristics the approach can produce, and the large number of parameter values which must be selected. Problem domains such as timetabling, vehicle routing, job scheduling, cutting and packing, have been identified as having great potential for use of a HH approach [12].

Fukunaga [18] proposed the Composite heuristic Learning Algorithm for SAT Search (CLASS). CLASS is a GP system that automatically discovers satisfiability testing (SAT) local search heuristics. A special purpose composition operator, based on existing SAT heuristics, was used to create new individuals. The heuristics evolved were shown to be competitive with well-known SAT local search algorithms.

Burke et al. [19] proposed a GP system which automated the heuristic generation process, and produced human-competitive heuristics for the online bin packing problem. The GP system evolved a control program that rates the suitability of each bin for the next piece, using the attributes of the pieces to be packed and the bins.

Nguyen et al. [20] investigate a GP based HH approach which evolves adaptive mechanisms (GPAM). GP chooses from a set of low level heuristics and constructs an adaptive mechanism, which is evolved simultaneously with the problem solution. Results showed that GPAM was a robust HH method, providing good quality solutions that performed competitively with existing HH on the MAX-SAT, bin packing and flow-shop scheduling problem domains.

3 Genetic Programming Hyper-Heuristic Method for Feature Selection

This section describes the new approach to FS. We are looking for heuristics (algorithms) and need a representation for them. A heuristic is an executable program, as are genetic programs. Hence we use GP individuals to represent heuristics. The heuristic operates in the search space of the problem. In this work the problem domain is FS, and we need a representation for the search space of all possible feature subsets. We first introduce the representation of the FS search space, then the representation of the heuristic, before describing the heuristic components.

3.1 Representation: Feature Selection Search Space

A common representation of a FS solution is to use a string of zeros and ones with length n, where a zero or one at the i-th position specifies the absence or presence of the i-th feature in the solution. This representation is common in genetic algorithms for FS [21,22], but can be improved by mapping the string to a graph of subsets of features in which adjacent nodes are obtained by inclusion or exclusion of a feature from the existing node (subset). One such graph is depicted in Figure 1. The graph can give more topological information to a FS algorithm [23]. The size of the search space of a FS problem grows exponentially with respect to the number of features in any given classification task. With n features in a classification task, the search space includes 2^n points, one for each candidate subset (solution).

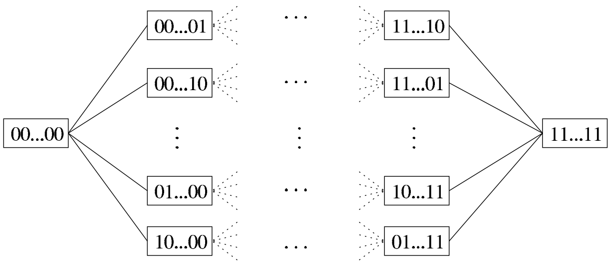


Fig. 1. Search space of the feature subsets in the form of a graph where each node represents a feature subset with a string of zeros and ones showing the absence and presence of the corresponding original features

The left-most node, $[00\ldots00]$ is the empty set, from which we can move to any feature subset with cardinality one. If we consider moving from a subset with cardinality k to a subset with either one more or one less feature than the current subset there are n possible moves. There are k possible moves to a subset of cardinality $k-1$, and $n-k$ possible moves to a subset of cardinality $k+1$.

3.2 Representation: Heuristic Search Space

The aim is to evolve heuristic programs that perform FS by searching the space of subsets of features. The idea is that sophisticated heuristics can be composed by putting some smaller primitive building blocks (primitive heuristics) together and organising them in an appropriate manner. We use GP for this purpose. A GP individual is represented by a tree which performs basic heuristic steps on a given initial subset of features. The heuristic search space is therefore the space of all possible trees.

Figure 2 shows an example GP tree. All non-terminal nodes (parent nodes) are of type 'Op' which means they execute their child nodes from left to right. The terminals nodes are primitive heuristics. The tree in Figure 2 contains two abstract primitive heuristics, $Move1$ and $Move2$. These 'move' functions change the current selected subset to another one by moving on the graph. The primitive heuristics are the means of moving around the search space (the space of subsets of features). In the next subsection we introduce two concrete instances of primitive heuristics that are used in our proposed system.

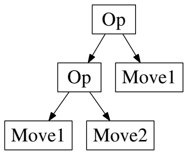


Fig. 2. An example GP individual representing a heuristic

3.3 Primitive Functions

From two existing heuristic approaches (SFS and SBS) we have extracted two primitive operators (basic heuristics):

- **Greedy Left (GL):** We define a move left to be the removal of one feature. GL evaluates all possible left moves (i.e. all subsets with one fewer feature than the current subset) and moves to the subset which allows the greatest increase (or smallest decrease) in fitness in comparison to the current subset.
- **Greedy Right (GR):** We define a move right to be the addition of one feature. GR evaluates all possible right moves (i.e. all subsets with one more feature than the current subset) and moves to the subset which allows the greatest increase (or smallest decrease) in fitness in comparison to the current subset.

GL and GR are basic heuristics that will be used to explore the graph of subsets of features. The operators move from a given subset to a subset with either one more or one fewer features. They are used as the terminal set for our GP system.

The terminals are linked by the operator *op* which acts as a place holder in the GP tree and simply passes the current feature subset and associated fitness value to its child nodes and then on to its parent node.

3.4 Fitness Function

The proposed GP system searches the space of possible heuristics (i.e. GP is the hyper-heuristic). In this work each GP program is a search algorithm which searches the space of possible subsets of features in order to find a good quality subset (i.e. GP programs are heuristics). Finally for each final subset selected by a GP program there is a search for a classifier model based on this subset.

The goodness of a GP individual is judged based on the goodness of the final subset it finds by traversing the terminals of the tree from left to right (starting from some initial subset). The goodness of a subset is judged based on the performance of a given classifier using that subset, i.e., the goodness of the subset is evaluated using a wrapper approach [24]: the selected subset is passed to an external classifier, J48, which constructs a model and returns the classification accuracy of the model. The goodness of the final subset found by a GP individual is then used as the fitness of the program. As evaluating all subsets visited is computationally expensive, we store visited subsets and their fitness value in a *map* (i.e. cache fitness values). J48 was used in the training (evolutionary) process for evaluating the evolving the heuristics (subsets of features) as this is a wrapper method.

4 Experiment Design

In this section we provide implementation details of the proposed system.

4.1 Datasets

We selected three datasets from the UCI Machine Learning Repository [25]. Each dataset is a binary classification task and we split 66%/34% into training and testing sets. Table 1 presents a summary of the details of the datasets.

Table 1. Properties of datasets used in experiments.

Dataset	# Features	# Instances	# Classes	
Ionosphere	34	351	2	(Good/Bad)
WBC Original	9	683	2	(Malignant/Benign)
Pima Diabetes	8	768	2	(Diabetes/Not)

Ionosphere: each data instance represents radar returns that either show evidence of structure ("Good") or no structure ("Bad") in the ionosphere.

Wisconsin Breast Cancer Original (WBC): each data instance represents a breast tumor that is either malignant or benign.

Pima Indian Diabetes: each data instance represents a female of Pima Indian heritage over the age of 21 that either shows or does not show signs of diabetes.

4.2 Parameters

We used a tree-based GP approach, and the ECJ V20 [26] package. The evolutionary parameters were chosen according to values reported in the literature [27,20]. A population of 100 programs was used; the initial population was generated using the ramped half-and-half method, and evolved for 50 generations. Minimum tree depth 2, maximum tree depth 6, crossover rate 90%, mutation rate 5%, reproduction rate 5%, and tournament selection with tournament size of 7. We used the J48 decision tree classifier in the Weka software package [28].

The terminal set consists of the two elementary operators *greedy-left* an *greedy-right* and the function set consists of the operator *op* which are described in Section 3.

We initialise all GP individuals with subset $[1, 0, 1, 0, 1...]$.

5 Results

There were 40 independent GP runs performed with each dataset and the same 40 random seeds were used for each dataset. At the end of each of the 40 GP runs, the best GP program (heuristic) is used to select a subset of features. We take this subset of features and project the test set through. This gives a test set with a reduced number of features, which is passed to J48 classifier inducer (learner). The performance of the classifier on the test set is obtained by 10-fold cross validation. We take the classification accuracy on the test set as the measure of the performance of the heuristic.

Table 2. Classification results on test set using subsets selected by evolved heuristics, including average training times and the number of explored subsets

	Full Test Set		Selected Subset		# Subset	Evolutionary
	Accuracy(%)	Size	Accuracy(%)	Avg. size	Evaluations	training time(s)
Ionosphere	92.50	34	96.52±0.53	7.83	82703	3160
WBC	96.05	9	96.51±0.00	5.15	267	6.81
Pima Diabetes	71.26	8	74.31±0.12	1.98	120	4.60

Table 2 shows the average performance over 40 GP runs on each of the three datasets. For example, for the Ionosphere dataset, the classification accuracy on the test set without FS is 92.50%. In comparison the mean classification accuracy on the test set with feature selection performed by the generated heuristics is 96.52%±0.53, an increase by 4.02%. The mean size of the selected subset is 7.83 out of the 34 features. It is of interest to note that from 40 GP runs only 12 distinct feature subsets were evolved. 14 features were not selected in any of these subsets. This indicates that the evolved heuristics (best individuals) somewhat agree with each other on the goodness of features.

On the WBC dataset the improvement in classification accuracy obtained by using the proposed approach was the smallest, increasing from 96.05% to 96.51 ± 0.00. However, the average number of features has been reduced from 9 to 5.15. Only two distinct feature subsets were evolved in the 40 independent runs.

On the Pima Diabetes dataset the mean classification accuracy was 74.31 ± 0.12, an improvement over the accuracy of 71.26% obtained using all eight features. Five distinct feature subsets were evolved. The average number of selected features were reduced from 8 to 1.98.

In all cases the standard deviation of classification accuracy is small, which indicates a reasonably consistent performance across the 40 GP runs, and in no GP run for any dataset was the full set of features selected.

Table 2 also shows the average training times and the number of explored subsets. For example for the Ionosphere dataset, 82703 subsets (out of more than 16 billion subsets) were examined during the evolution process. The mean evolutionary training time was 3160 seconds. As expected with most FS algorithms, the number of explored subsets and the training (evolution) time is proportional to the size of the problem (in terms of the number of features).

In this work our approach was to evolve heuristics, so a direct comparison to existing feature selection heuristics is not entirely relevant. However to give an indication of the comparison, LFS and greedy stepwise (with backward selection) in Weka [28] selected a subset of size nine which gave classification accuracy of 96.67% on Ionosphere, the subset of size nine with accuracy 96.05% on the WBC dataset, and for the Pima Diabetes dataset a subset of size four which gave classification accuracy of 73.18% on the test set. This suggests that the proposed GPHH approach can automatically evolve multiple heuristics (FS algorithms) that perform better or at least as well as long-established FS algorithms.

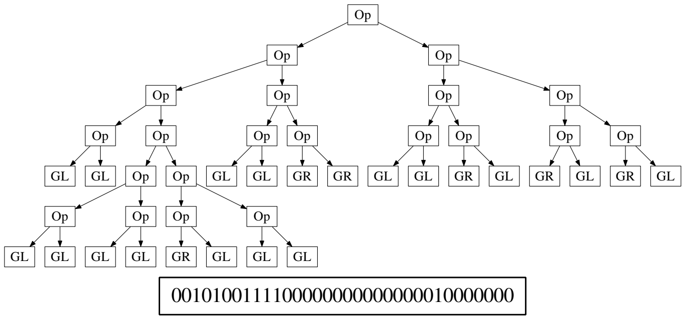
5.1 Example Heuristic

Figure 3 shows a GP tree (program) created during evolution for FS on the Ionosphere dataset. Starting with a given subset of features the program is executed by first generate an *inorder traversal* of the tree and then executing the resulting sequence (ignoring the 'op' operators). Thus for the program in Figure 3, the search in the space of subsets of features starts with $GL, GL, GL, GL, Gl, GL, GR, GL \dots$ and ends with $\dots, GR, GL$.

The box at the bottom of Figure 3 shows the output of the tree. That is, the tree selects the subset $\{X_3, X_5, X_8, X_9, X_{10}, X_{11}, X_{27}\}$. Training and testing a decision tree classifier using only these seven selected features using 10-fold cross-validation, yields an average performance (accuracy) of 96.67%, which is much better 92.50% achieved using all the 34 features.

6 Conclusions

The goal was to use GP to evolve heuristics that are able to find solutions, subsets of features, that are smaller than the full set of features and give better



```
0010100111100000000000000010000000
```

Fig. 3. A GP tree (heuristic) evolved on Ionosphere dataset (top) and the output generated from example input with this tree (bottom). The output cannot be generated from the tree alone, but must be given an initial feature subset.

classification performance. The goal was successfully achieved by using a tree-based representation for heuristics, introducing new primitive operators (*Greedy Left* and *Greedy Right*) to move in the graph of subsets of features, and using GP to search the space possible heuristics (trees). The results show that the proposed approach for feature selection (FS) successfully evolves heuristics which are able to select a smaller subset of features which give a higher classification accuracy than the full feature set.

This paper represents the first work of using GP to evolve heuristics (computer algorithms) for FS and it opens the door of using GP to evolve search algorithms for FS. While showing promise, this paper only tested this idea on three datasets. For future work, we will apply this approach to more datasets with a larger number of features (say, with over 300 features) to reveal whether this approach can be treated as a general approach to producing promising search algorithms for FS. We will also investigate new program representations and fitness functions to evolve new feature selection algorithms and compare them with more established algorithms in the literature. It would also be interesting to investigate new ways of initialising the individual chromosomes to reveal whether the performance of the evolved heuristics will be significantly affected.

References

1. Neshatian, K.: Feature Manipulation with Genetic Programming. PhD thesis, Victoria University of Wellington, New Zealand (2010)
2. Kira, K., Rendell, L.A.: The feature selection problem: Traditional methods and a new algorithm. In: Swartout, W.R. (ed.) pp. 129–134. AAAI, AAAI Press / The MIT Press (1992)
3. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.: Exploring Hyper-heuristic Methodologies with Genetic Programming. In: Computational Intelligence: Collaboration, Fusion and Emergence, pp. 177–201. Springer (2009)
4. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35(3), 268–308 (2003)
5. Burke, E., Kendall, G.: Introduction. In: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, pp. 5–18. Springer, Heidelberg (2003)
6. Almuallim, H., Dietterich, T.G.: Learning with many irrelevant features. In: Proceedings of the Ninth National Conference on Artificial Intelligence, pp. 547–552. AAAI Press (1991)
7. Whitney, A.W.: A direct method of nonparametric measurement selection. IEEE Trans. Comput. 20(9), 1100–1103 (1971)
8. Caruana, R., Freitag, D.: Greedy attribute selection. In: Proceedings of the Eleventh International Conference on Machine Learning, pp. 28–36. Morgan Kaufmann (1994)
9. Gutlein, M., Frank, E., Hall, M., Karwath, A.: Large-scale attribute selection using wrappers. In: IEEE CIDM, pp. 332–339 (2009)
10. Vafaie, H., De Jong, K.A.: Genetic algorithms as a tool for feature selection in machine learning. In: ICTAI, pp. 200–203 (1992)
11. Xue, B., Zhang, M., Browne, W.N.: Single feature ranking and binary particle swarm optimisation based feature subset ranking for feature selection. In: Proceedings of the 35th Australasian Computer Science Conference, pp. 27–36 (2012)

12. Ross, P.: Hyper-heuristics. In: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, pp. 529–556. Springer (2003)
13. Abdullah, S., Sabar, N.R., Nazri, M.Z.A., Turabieh, H., McCollum, B.: A constructive hyper-heuristics for rough set attribute reduction. In: IEEE ISDA, pp. 1032–1035 (2010)
14. Marill, T., Green, D.: On the effectiveness of receptors in recognition systems. IEEE Transactions on Information Theory 9(1), 11–17 (1963)
15. Neshatian, K., Zhang, M., Andreae, P.: Genetic Programming for Feature Ranking in Classification Problems. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 544–554. Springer, Heidelberg (2008)
16. Neshatian, K., Zhang, M.: Genetic Programming for Feature Subset Ranking in Binary Classification Problems. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 121–132. Springer, Heidelberg (2009)
17. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: A classification of hyper-heuristics approaches. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, pp. 449–468. Springer (2010)
18. Fukunaga, A.S.: Automated discovery of local search heuristics for satisfiability testing. Evol. Comput. 16(1), 31–61 (2008)
19. Burke, E.K., Hyde, M.R., Kendall, G., Woodward, J.: Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007, pp. 1559–1565. ACM, New York (2007)
20. Nguyen, S., Zhang, M., Johnston, M.: A genetic programming based hyper-heuristic approach for combinatorial optimisation. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011, pp. 1299–1306. ACM, New York (2011)
21. Smith, M.G., Bull, L.: Genetic programming with a genetic algorithm for feature construction and selection. Genetic Programming and Evolvable Machines 6, 265–281 (2005); Published online: August 17, 2005
22. Oh, I.S., Lee, J.S., Moon, B.R.: Hybrid genetic algorithms for feature selection. IEEE Transactions on Pattern Analysis and Machine Intellignece, 1424–1437 (2004)
23. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research 5, 1205–1224 (2004)
24. Kohavi, R., John, G.: Wrappers for feature subset selection. Artificial Intelligence 97, 273–324 (1997)
25. Frank, A., Asuncion, A.: UCI Machine Learning Repository (2010)
26. Luke, S.: ECJ 20 – A Java-based evolutionary computation research system
27. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann, San Francisco (1998)
28. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. 11, 10–18 (2009)

A New Approach to Vision-Based Fire Detection Using Statistical Features and Bayes Classifier

Ha Dai Duong and Dao Thanh Tinh

Faculty of Information Technology, Le Quy Don Technical University,
100 Hoang Quoc Viet, Cau Giay, Ha Noi, Vietnam
duonghadai@yahoo.com, tinhdt@mta.edu.vn
http://fit.lqdtu.edu.vn/default.aspx

Abstract. Computer vision - based fire detection has recently attracted a great deal of attention from the research community. In this paper, the authors propose and analyse a new approach for identifying fire in videos. In this approach, we propose a combined algorithm for detecting the fire in videos based on the changes of the statistical features in the fire regions between different frames. The statistical features consist of the average of the red, green and blue channel, the coarseness and the skewness of the red channel distribution. These features are evaluated, and then classified by Bayes classifier, and the final result is defined as fire-alarm rate for each frame. Experimental results demonstrate the effectiveness and robustness of the proposed method.

Keywords: Fire detection, Pattern recognition, Bayes classification.

1 Introduction

Two main applications of vision-based fire detection are: (1) monitoring fires and burn disasters from surveillance systems [1], and (2) automated retrieval of events in newscast videos [2]. These applications play an important role in modern society. Recently, there have been a number of efficient methods proposed for vision-based fire detection in [1]-[6].

In [1], Chao-Ching Ho analysed the spectral, spatial and temporal characteristics of the flame and smoke regions in the image sequences. Then, the continuously adaptive mean shift vision tracking algorithm was employed to provide feedback of the flame and smoke real-time position at a high frame rate. P. V. K. Borges and E. Izquierdo, in [2], analysed the frame-to-frame changes of specific low-level features such as color, area size, surface coarseness, boundary roughness, and skewness within estimated fire regions to describe potential fire regions and used Bayes classifier to indicate a frame contains fire or not. In [3], Celik T. et al. developed two models, one for fire detection and the other for smoke detection. For fire detection, the concepts from fuzzy logic were used to make the classification fire and fire-like coloured objects. For smoke detection, a statistical analysis was carried out using the idea that the smoke shows grayish colour with different illumination. In [4], the authors also used a probabilistic metric to threshold potential fire pixels. This was achieved by multiplying the probabilities of each individual color channel being fire. Habiboglu et al. proposed a video-based fire detection method, in [6], which used color, spatial and temporal information by dividing the video into spatio-temporal blocks and used covariance-based features extracted from these blocks to detect fire. However, when the flickering behaviour of flames cannot be visible in video, the method might perform poorly.

The majority of the vision-based fire detection methods employ some type of hybrid model combining color, geometry and motion features. In general, fire detection systems first use some key features as a precondition to generate seed areas for candidate fire regions, then use the other features to determine the existence of fire in candidate fire regions. In most existing approaches, the color information and their derived descriptors like area size, surface coarseness, boundary roughness and skewness were mainly used for classification features [2]. However, using the color channels are not reliable enough to perform classification. Similarly, the classifying tasks based on the features derived from the color information can not perform properly.

In contrast to some works, we propose a new approach to solve the problem of fire detection by determining motion region, then this region is used to establish a vector of fire features. Bayes classifier uses the vector to indicate the motion region is contained fire or not. The number of detected fire regions in a frame is used to compute fire-alarm rate of the frame. The rest of this paper is organized as follows: Section 2 analyses some significations of fire in a region, these significations are motion (Section 2.1), color (Section 2.2), red channel distribution skewness (Section 2.3) and surface coarseness (Section 2.4); Section 3 presents a fire detection algorithm, it includes a brief review of Bayes classifier (Section 3.1) and a description of algorithm (Section 3.2); Section 4 presents some experimental results, followed by a relevant conclusion in Section 5.

2 Statistical Features of Fire Regions

Intuitively, fire has some unique visual signatures such as color, geometry, and motion. Some proposed methods use these characteristics as interdependent parameters, for example, area size, boundary roughness, coarseness, skewness in [2] depended on definition of color. In this proposal, the authors evaluate these visual signatures of fire as independent features. A potential fire region is determined as a motion region. The statistical features in potential fire region include color, skewness of red channel histogram, and surface coarseness.

2.1 Motion

In order to detect possible changes, which may be caused by fire, this proposal estimates the change of regions by regions. Formally, we assume F_{t1} is frame at $t1$, and F_{t2} is frame at $t2 = t1 + \Delta t$. The motion of fire leads to the changes of intensity in some pixels between F_{t1} and F_{t2}. This work determines the difference between F_{t1} and F_{t2} as follow:

1. Divide F_{t1} and F_{t2} into $N \times M$ grid, where N is number of rows and M is number of columns, Fig. 1 is an example;
2. Compute the correlation coefficient between a region on the grid of F_{t1} and corresponding region on the grid of F_{t2};
3. Decide the region on F_{t2} is a motion region if the correlation coefficient is less than or equal to a pre-defined threshold T.

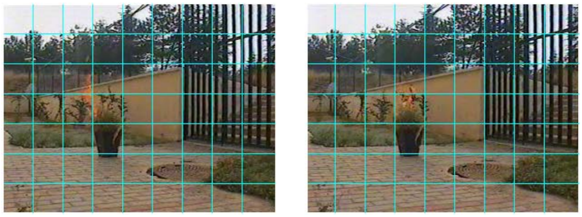


Fig. 1. F_{t1} and F_{t2} are divided into $N \times M$ grid

Let W, H be the width and height of F_{t1} and F_{t2}, divide F_{t1} and F_{t2} into $N \times M$ grid, where N is number of rows and M is number of columns. Then each region of the grid has size of $D_W = W/M$ in width and $D_H = H/N$ in height. Denote A_{kl}, B_{kl} as regions at k-th row and l-th column ($k = 1, 2, .., N, l = 1, 2, .., M$) on the grids of F_{t1} and F_{t2} respectively. The correlation coefficient between A_{kl} and B_{kl} is defined as

$$CC_{kl} = \frac{\sum_{i=1}^{D_H} \sum_{j=1}^{D_W} A_{kl}(i,j) \times B_{kl}(i,j)}{\sqrt{\sum_{i=1}^{D_H} \sum_{j=1}^{D_W} A_{kl}(i,j)^2} \times \sqrt{\sum_{i=1}^{D_H} \sum_{j=1}^{D_W} B_{kl}(i,j)^2}}. \tag{1}$$

The region B_{kl} on F_{t2} is considered as motion region if:

$$CC_{kl} \leq T \tag{2}$$

where T is a decision threshold. Each region satisfied Eq. (2) is a potential region for next step.

2.2 Color

As other works in the field of fire detection, this paper concerns only on the color of flames belongs to the red-yellow range, which are the most common type of flames seen in the nature. Other types of flames, such as blue liquefied petroleum gas flames, are not considered in this paper. For the type of flames considered, most papers presented in the fire detection literature assumed that for a given fire pixel, the value of red channel is greater than the green channel, and the value of the green channel is greater than the value of blue channel. However, laboratory experiments show that above assumption ignores some pixel in fire

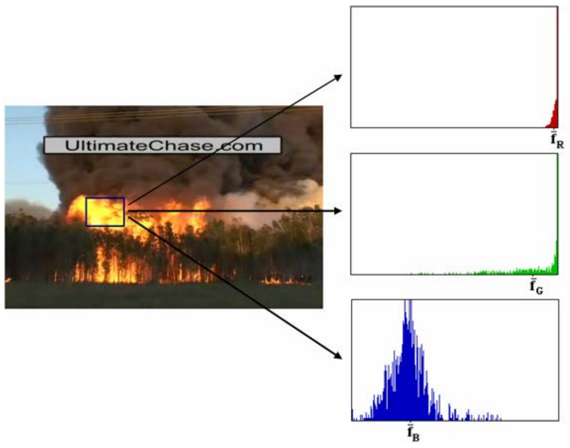


Fig. 2. Histogram and average value of red, green, and blue channels in a fire region

blobs, then any features extracted from potential fire blobs may be unreliable. To overcome that difficulty, this proposal considers the average value of each channel in a region instead of themselves. For example in Fig. 2, average value of red is greater than the average value of green and the average value of green greater than average value of blue channel.

The average value of red, green, and blue channels in a fire region are computed as

$$\bar{f_R} = \frac{1}{D_H \times D_W} \sum_{i=1}^{D_H} \sum_{j=1}^{D_W} B_{kl}^R(i,j), \tag{3}$$

$$\bar{f_G} = \frac{1}{D_H \times D_W} \sum_{i=1}^{D_H} \sum_{j=1}^{D_W} B_{kl}^G(i,j), \tag{4}$$

and

$$\bar{f_B} = \frac{1}{D_H \times D_W} \sum_{i=1}^{D_H} \sum_{j=1}^{D_W} B_{kl}^B(i,j), \tag{5}$$

where $B_{kl}^R(i,j)$, $B_{kl}^G(i,j)$, and $B_{kl}^B(i,j)$ are the red, green, and blue channels representation of $B_{kl}(i,j)$, respectively. This proposal use $\bar{f_R}$, $\bar{f_G}$, and $\bar{f_B}$ as three features of a potential region for classification in Section 3.

2.3 Skewness

Denote $p_R(v)$ as normalized histogram of red channel in a region, v is a intensity, then $p_R(v)$ gives the estimate of the probability of occurrence of intensity v. In this case, the third moment of $p_R(v)$ measures its symmetry with respect to the mean, it is also called the skewness (see [7], [8]). The skewness is zero when the distribution is symmetric, positive if the distribution shape is more spread to the right and negative if it is more spread to the left, as illustrated in Fig. 3. This causes the skewness of red channel distribution to have a negative value. For this reason, the skewness is used as an useful feature to identify fire regions.

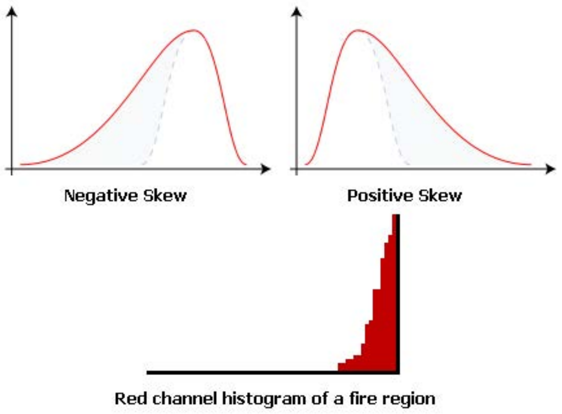


Fig. 3. Illustration of negative, positive skew and red channel histogram of a fire region

Denote s_R as the skewness of the red channel distribution of candidate region B_{kl}, it is defined as

$$s_R = \frac{\frac{1}{D_H \times D_W} \sum_{i=1}^{D_H} \sum_{j=1}^{D_W} (B_{kl}^R(i,j) - \bar{f}_R)^3}{\sigma_{f_R}^3} \tag{6}$$

where σ_{f_R} is the standard deviation of $p_R(v)$. This proposal uses s_R as one feature of a candidate region for classification in Section 3.

2.4 Coarseness

This proposal considers the spatial color variation in each potential region to distinguish between fire region and fire-like coloured region. Unlike other false-alarm regions, for example a region of red t-shirt in Fig. 4, fire regions have a significant amount of variability in the pixel values. The variance is a well-known metric (see [7], [8]) to indicate the amount of coarseness or the spatial color variation in the pixel values. Hence, this work uses the variance of the region as a feature to eliminate non-fire region from a candidate region.

Denote c as the coarseness of candidate region B_{kl}, it is defined as

$$c = \frac{\frac{1}{D_H \times D_W} \sum_{i=1}^{D_H} \sum_{j=1}^{D_W} (B_{kl}(i,j) - \bar{f}_R)^2}{\sigma_{f_R}^2} \tag{7}$$

In this proposal, c is fifth feature of a candidate region for classification.

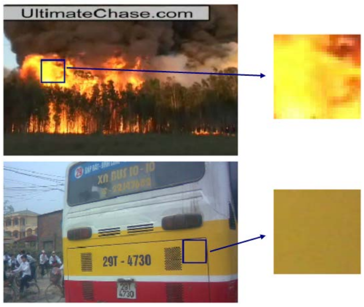


Fig. 4. Spatial variation in fire and in a fire-like coloured object

3 Fire Detection Algorithm

3.1 Bayes Classifier

Bayesian decision theory is a fundamental statistical approach to the problem of pattern recognition. It makes the assumption that the decision problem is posed in probabilistic terms. For each candidate region, five independent features as aforementioned are evaluated and constructed a vector $\mathbf{x}$ as

$$\mathbf{x} = \begin{bmatrix} \bar{f_R} \\ \bar{f_G} \\ \bar{f_B} \\ s_R \\ c \end{bmatrix}, \tag{8}$$

and this proposal uses $\mathbf{x}$ to indicate candidate region contained fire or not by applying Bayes classifier.

Without the lost of generality, assume that we have Ω different classes. Using the Bayes inference, we can represent

$$P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{p(\mathbf{x})}, \tag{9}$$

where ω_i is i-th class, $i = 1..\Omega$. One of the most useful and widely used ways to represent pattern classifier is by use of the discriminant functions

$$g_i(\mathbf{x}) = \ln p(\mathbf{x} \mid \omega_i) + \ln P(\omega_i). \tag{10}$$

The classifier is said to assign a feature vector $\mathbf{x}$ to class ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$. If the densities $p(\mathbf{x} \mid \omega_i)$ are multivariate normal - Eq. (10) will take the form

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \mu_i)^T C_i^{-1}(\mathbf{x} - \mu_i) - \frac{K}{2}\ln 2\pi - \frac{1}{2}\ln |C_i| + \ln P(\omega_i) \tag{11}$$

For the case when the covariance matrices C_i are different for each category, the resulting discriminant functions will be inherently quadratic

$$g_i(\mathbf{x}) = \mathbf{x}^T W_i \mathbf{x} + w_i^T \mathbf{x} + \omega_{i0} \tag{12}$$

where

$$W_i = -\frac{1}{2} C_i^{-1}, \tag{13}$$

$$w_i = C_i^{-1} \mu_i, \tag{14}$$

and

$$\omega_{i0} = -\frac{1}{2} \mu_i^T C_i^{-1} \mu_i - \frac{1}{2} \ln |C_i| + \ln P(\omega_i), \tag{15}$$

where μ_i is the mean vector, C_i is the covariance matrix of class i.

Let ω_1 represents the fire class and ω_2 represents the non-fire class. For two-category case a single discrimination function is used

$$g(\mathbf{x}) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x}), \tag{16}$$

and the following decision rule is used: Decide ω_1 if $g(\mathbf{x}) > 0$; otherwise decide ω_2.

In order to classify the class fire from the class non-fire, the Bayes classifier needs to estimate the mean and the variance of each class. Therefore, it requires a statistical training, based on observed values, to determine a decision function that separates the classes, this task is presented in experimental section.

3.2 Algorithm

Different from most existing fire detection algorithms in which they return two-state fire alarm containing fire or not for a input frame (or video). This paper proposes a new algorithm that give a fire-alarm rate ($FiAR$) in range $[0-1]$ for each frame. The fire-alarm rate is defined as

$$FiAR = \begin{cases} n/\Theta, & \text{if } n < \Theta \\ 1, & \text{Otherwise} \end{cases} \tag{17}$$

where n is number of region contained fire in current frame, Θ is a decision threshold, it is found out by practising. For input frame f, our proposed algorithm does following steps:

1. Determine all motion region;
2. For each motion region B_{kl}:
   (a) Evaluate vector of features $\mathbf{x}$ as presented in (8);
   (b) Indicate B_{kl} is contained fire or not by using (16).
3. Calculate the fire-alarm rate by using (17);
4. Show the $FiAR$ as output of algorithm.

A block diagram of the algorithm is given in Fig. 5.

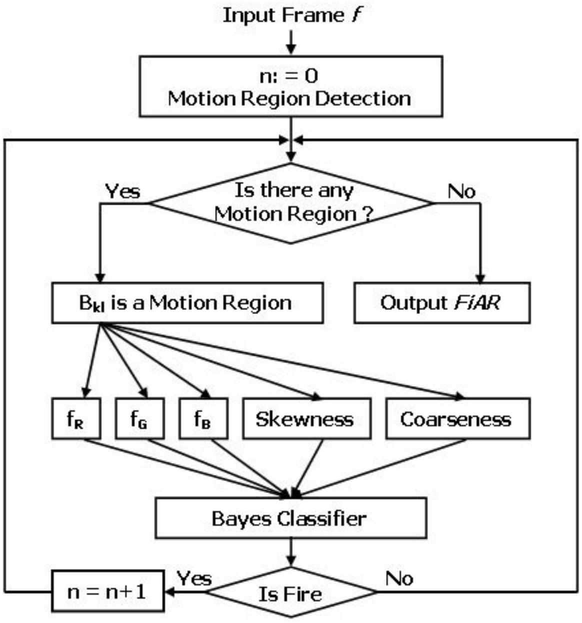


Fig. 5. Block diagram of proposed algorithm

4 Experiments

To evaluate the proposed approach, we collected 18 videos and used the data sets available from the site http://signal.ee.bilkent.edu.tr/VisiFire which consists of 10 fire videos and 8 non-fire videos. The video resolution is 320×240 and the frame rate varies from 15 Hz to 30 Hz.

For training data, we extracted 8645 fire regions and 11170 non-fire regions from 9 test videos, the region is divided in size of 8×6. Then a vector of features $\mathbf{x}$ is established for each region. These vectors of fire and non-fire class are applied for training phase to get classifier decision function (16).

4.1 Experiment 1

Assume that, when number of fire pixels is equal or greater than $n/4$, where n is number of pixels in frame f, then the accurate fire-alarm rate, denote $FiAR_A$, is 1. Approximately, the accurate fire-alarm rate can be defined as

$$FiAR_A = \begin{cases} m/\Phi, & \text{if } m < \Phi \\ 1, & \text{Otherwise} \end{cases} \tag{18}$$

where m is number of fire regions indicated manually in current frame, and $\Phi = (H \times W)/4$.

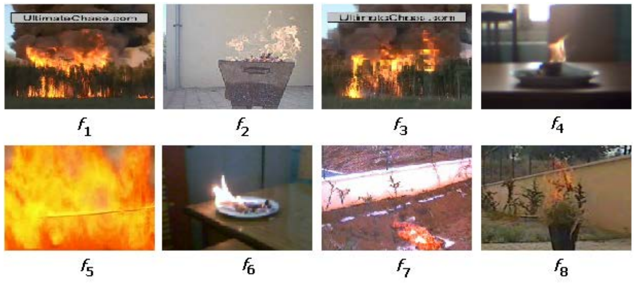


Fig. 6. Samples for fire-alarm rate comparing

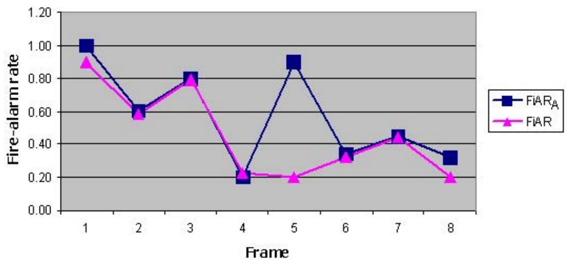


Fig. 7. Comparison of fire-alarm rate

This experiment shows the result of comparing between $FiAR$ defined by (17) and $FiAR_A$ defined by (18), and 8 frames in Fig. 6 are used.

The results of comparing is showed in Fig.7. In this, 7/8 frames have same fire-alarm rate. In the case of fame f_5, in spite of having large accurate fire-alarm rate, the proposed fire-alarm rate is small. The cause of this situation is number of detected motion region in a large fire is small.

4.2 Experiment 2

This experiment uses assumption that, frame f contains fire if $FiAR > 0$, to compare with three others: Method 1 [4], Method 2 [1] and Method 3 [2]. Data for this experiment includes 10 fire videos and 8 non-fire videos. The video resolution is 320×240 and the frame rate varies from 15 Hz to 30 Hz. There are approximately 11,676 frames. Table 1 shows the performance comparison of the proposed method and others. The proposed method outperforms other algorithms in terms of consistently increasing accuracy of fire detection and decreasing error rate.

5 Conclusions

In this paper, a new approach to vision-based fire detection is presented. By using fire characteristics in a region as a vector of statistical features, this work

Table 1. Performance comparison of the proposed method and other methods

Method	False-Positive	False-Negative
Proposed Method	0.160%	0.025%
Method 1	0.270%	0.260%
Method 2	0.680%	0.028%
Method 3	0.290%	12.360%

employed Bayes classifier for the vector to distinguish fire or non-fire region. Then a fire detection algorithm and a fire-alarm rate are presented. The experiments show that the fire-alarm rate can be used as a meaningful alarm. Moreover, the proposed method provided the output which can reach the most accurate in false-positive and false-negative. In spite of not comparing about the complexity and time consuming of our algorithm with others, it is easy to recognize this algorithm is an efficient approach for video-based fire detection.

Acknowledgments. This work was funded by the Vietnam National Foundation for Science and Technology Development (NAFOSTED), Grant No 102.-012010.12. and the Reseach Fund RFit@LQDTU, Faculty of Information Technology, Le Quy Don Technical University.

References

1. Ho, C.-C.: Machine vision-based real-time early flame and smoke detection. Meas. Sci. Technol. 20(4), 045502, 13 p (2009)
2. Borges, P.V.K., Izquierdo, E.: A probabilistic approach for vision-based fire detection in videos. IEEE Trans. Circuits Syst. Video Technol. 20(5), 721–731 (2010)
3. Celik, T., Ozkaramanl, H., Demirel, H.: Fire and smoke detection without sensors: image processing - based approach. In: Proc. 15th European Signal Processing Conf., pp. 1794–1798 (2007)
4. Ko, B.C., Cheong, K., Nam, J.: Fire detection based on vision sensor and support vector machines. Fire Safety J. 44(3), 322–329 (2009)
5. Duong, H.D., Tinh, D.T.: A Novel Computational Approach for Fire Detection. In: Proc. of KSE 2010 The Second International Conference on Knowledge and Systems Engineering, Hanoi, Vietname, pp. 9–13 (2010)
6. Habiboglu, Y.H., Günay, O., Çetin, A.E.: Covariance matrix-based fire and flame detection method in video. Machine Vision and Applications, doi: 10.1007/s00138-011-0369-1
7. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Addison-Wesley, Reading (1992)
8. Theodoridis, S., Koutroumbas, K.: Pattern Recognition. Academic, New York (2006)
9. Fire detection sample video clips, `http://signal.ee.bilkent.edu.tr/VisiFire`

Automatic Discovery of Optimisation Search Heuristics for Two Dimensional Strip Packing Using Genetic Programming

Su Nguyen[1], Mengjie Zhang[1], Mark Johnston[1], and Kay Chen Tan[2]

[1] Victoria University of Wellington, Wellington, New Zealand
[2] National University of Singapore, Singapore
{su.nguyen,mengjie.zhang}@ecs.vuw.ac.nz, mark.johnston@msor.vuw.ac.nz
eletankc@nus.edu.sg

Abstract. This paper presents a genetic programming based hyper-heuristic (GPHH) for automatic discovery of optimisation heuristics for the two dimensional strip packing problem (2D-SPP). The novelty of this method is to integrate both the construction and improvement procedure into a heuristic which can be evolved by genetic programming (GP). The experimental results show that the evolved heuristics are very competitive and sometimes better than the popular state-of-the-art optimisation search heuristics for 2D-SPP. Moreover, the evolved heuristics can search for good packing solutions in a much more efficient way compared to the other search methods.

Keywords: genetic programming, bin packing, heuristics.

1 Introduction

Cutting stock problems are optimisation problems with many applications in industry. The goal is to minimise waste when cutting stock sheets of material into smaller pieces. This paper considers the two dimensional rectangular strip packing problem (2D-SPP) which arises in many industries such as furniture, clothing and glass production. For these problems, a given set of rectangular pieces must be arranged onto a large rectangular sheet with a fixed width and a height to be minimised. Lodi et al. [20] present a good overview of two dimensional packing problems. They classified 2D-SPP based on whether *rotation* of rectangular pieces or *guillotine cut* is applied. In this paper, we focus on the 2D-SPP in which pieces are not allowed to rotate and guillotine cut is not required [20,21].

Some exact optimisation methods such as linear programming [15] and tree search [13,16] have been proposed to solve 2D-SPP. Since this problem is known to be NP-hard [20,21], the exact methods can only be used to solve small problem instances. For that reason, heuristic methods are needed to deal with larger problems instances. The focus of the proposed heuristic methods is the placement policy and the order in which the pieces will be packed. Baker et al. [4] proposed the bottom-left-fill (BLF) heuristic which tries to place the pieces as low as possible and as far to the left as possible. Chazelle [12] proposed a more efficient implementation of this heuristic and presented an optimal method to determine the ordered list of positions where a piece can be placed. Belov et al. [5] modified BLF to Bottom-left-right (BLR) used in their iterative heuristics. They also proposed a Substitution (SubKP) approach that fills every most bottom-left free hole in a greedy one-dimensional way (only considering the piece's width) by solving a one-dimensional knapsack problem. Another popular construction heuristic for 2D-SPP is *best-fit* (BF) proposed by Burke et al. [7]. This heuristic keeps track of, and updates, all available slots in the stock sheet after each piece is placed. It handles each placement by choosing the lowest slot and filling it with the piece with the largest width that can fit into that slot. If no piece can fit into the slot, the slot will be merged with its lowest neighbour to make a wider slot and the slot structure will be updated. This heuristic was shown to be better than previously published heuristics. It was also enhanced by simulated annealing (BF+SA) [10] to deal with the limitation of BF at the end of the packing. More recently, Burke et al. [11] proposed a genetic programming based hyper-heuristic (GPHH) method [8] to learn construction heuristics for 2D- SPP. The evolved heuristics were shown to be very promising when they were compared with BF [7] and some meta-heuristic methods. They also provided some interesting insights on the generality of the evolved heuristics. However, the evolved heuristics still cannot achieve results competitive with those obtained by the state-of-the-art optimisation methods.

The limitation of construction heuristics is that they cannot always guarantee a good solution. For that reason, they are often hybridised with other meta-heuristic methods. Jakobs [18] applied a genetic algorithm (GA) to find a good ordering of pieces which will be packed by a simple bottom left (BL) heuristic. Some GA methods [3,17] were also proposed based on this idea with different packing heuristics. Hopper and Turton [17] presented experimental results from different meta-heuristic methods with BLF and BL [18] as the construction heuristic. The combinations of meta-heuristic methods and BLF [5] are shown to give better performance. Alvarez-Valdes et al. [2] proposed reactive GRASP (R-GRASP), a state-of-the-art heuristic for 2D-SPP. This method was able to outperform other meta-heuristic methods and it is still one of the best methods now. The problem with this method is that its performance depends on many parameters [11]. Belov et al. [5] proposed the SVC(SubKP) which iteratively changes the profits of pieces to be used in the SubKP construction heuristic. This method is very competitive as compared to R-GRASP [2] when tested on a large number of instances. Burke et al. [9] presented a simple but effective squeaky wheel optimisation (SWO) method for 2D-SPP. The key idea of SWO is to iteratively construct a solution by taking into account the elements of the problem that result in bad results in previous iterations. When dealing with 2D-SPP, Burke et al. [9] iteratively constructed a solution and penalised the pieces that exceed the lower bound when placed in the stock sheet. Their method used a variant of BF [7] as the construction heuristic where the pieces with the highest accumulative penalty will be placed in the slot first instead of the fittest piece.

The results showed that SWO was competitive and sometimes better than other complex algorithms such as BF+SA, R-GRASP, and SVC(SubKP). More recently, Aggoun et al. [1] also reported on an constraint programming method to deal with practical constraints in industrial packing problems.

In this paper, we propose a GPHH method to evolve effective and efficient heuristics for 2D-SPP. This work is motivated by the recent advances in computing power that allows us to automatically discover new heuristic methods for hard optimisation problems. The previous studies showed that effective methods for solving 2D-SPP require a good construction heuristic and iterative procedure to search for a good ordering of pieces. Burke et al. [11] was a great effort to handle the first requirement. Meanwhile, the second requirement is still needed to be fulfilled manually. This paper aims to develop a GPHH method that can satisfy these two requirements simultaneously. We will focus on the following three research objectives: (1) developing a new GPHH method to evolve heuristics for 2D-SPP, (2) comparing the results obtained by the evolved heuristics with other state-of-the-art methods, and (3) analysing the behaviour of the evolved heuristics.

The rest of this paper is organised as follows. Section 2 will describe the proposed GPHH method. Section 3 provides the experimental results, the comparisons between the evolved heuristics with the existing methods, and some insights into the evolved heuristics. Conclusions are given in Section 4.

2 The New Method

The biggest challenge when evolving a search heuristic for 2D-SPP is how to handle the construction heuristic and iterative search procedure in such a way that they can be easily be evolved by the proposed GPHH. A few methods have been proposed in the GPHH literature for evolving local search heuristics [14,6]. Although the evolved local search heuristics were shown to be very effective, these studies mainly focus on the improvement phases. In this section, we propose a simple approach to integrating both construction and improvement heuristics to be evolved by our GPHH.

2.1 Representation of Search Heuristics

The construction part of our search heuristics is similar to that in [11]. The heuristics evolved by their GPHH [11] is a mathematical function to determine the piece, its orientation and the slot to place it based on some features of the pieces and the current slot structure. In each placement step, the heuristic will calculate the score for each combination of piece, orientation, allocation, and slot, and the combination with the highest score will be applied for the next placement (orientation, allocation, and slot is fixed in our evolved heuristics to reduce the computational effort of the proposed GPHH). Since these evolved heuristics consider many pieces of information when deciding where to put a piece, they can provide better results compared to BF [7]. However, these evolved heuristics cannot estimate the impact of each placement decision on the final solution, which makes them less competitive compared to complex optimisation

Table 1. Terminal and Function sets

Function	+, −, ∗, protected division %, min, max, abs, and If
pW	the width of the piece
pH	the height of the piece
pA	the area of the piece
sH	the height of the slot
sWL	the difference between slot and piece width
shW	the width of the sheet
shH	the lower bound for optimal height of the sheet multiplied by 1.5
#	ephemeral random constant
pP	the average penalty of the pieces
ppX	the X-coordinate where the piece is placed in the previous solution
ppY	the Y-coordinate where the piece is placed in the previous solution

search methods. To handle this problem, our evolved heuristics will also include the statistics from previous packing solutions, which allow the evolved heuristics to iteratively correct the mistakes made in previous placement decisions.

The complete function set and the terminal set used to learn the search heuristics are given in Table 1. In the function set, we include min, max, abs, and If to allow GP to evolve sophisticated heuristics. The lower part of the table shows all the terminals. The first eight terminals are the same as the ones used in [11] and are self explainable. The last three terminals provide the information about previous packing solutions. While ppX and ppY give the location where the piece is placed in the previous packing solution, pP measures the average penalty of the piece from all previous packing solutions, which indicates the difficulty of placing the piece without increasing the height of the sheet. The way a penalty is calculated here is the same as used in [9], which penalises the piece an amount equal to its height if its top corners exceed the lower bound (simply calculated by dividing the total area of all pieces by the width of the sheet).

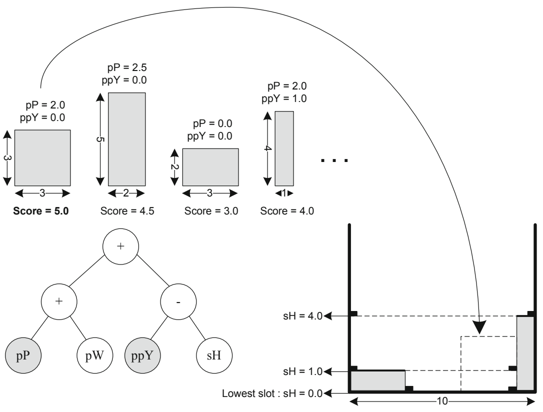


Fig. 1. Illustration of an evolved heuristic

An illustration of how an evolved heuristic is used is shown in Fig. [1]. In the figure, two pieces have been placed onto the sheet and we need to figure out which piece should be placed next. Similar to the BF heuristic, the lowest slot will be considered and we check whether there is any piece that can fit into this slot. If the slot is too narrow for any piece, it will be raised to merge with the lowest neighbour slot (refer to [7,9] for a more detailed description of BF). Otherwise, the evolved heuristic (i.e. $\mathtt{pP} + \mathtt{pW} + \mathtt{ppY} - \mathtt{sH}$) will calculate the score for each unpacked piece that can fit into the lowest slot and the piece with the highest score will be placed next. In this example, the first piece has the highest score since it has the highest sum of its width, penalty and previous placement position in the vertical axis (the height of the lowest slot in this case is zero and does not influence the scores of pieces). In our heuristic, we will place the selected piece next to the *tallest neighbouring* [7] piece in the sheet, which is the piece on the right in Fig. [1]. This heuristic will be applied until all pieces are placed onto the sheet. When the complete packing solution is obtained, `ppX`, `ppY` (initially zero) and `pP` (initially one) are updated. The evolved heuristic will iteratively generate new packing solution based on the new updated values of `ppX`, `ppY` and `pP`. This iterative procedure is similar to that used in SWO [9]. However, the new placement decisions depend on the evolved heuristic instead of depending only on the penalties of the pieces.

2.2 Fitness Function

To measure the quality of an evolved heuristic, it is applied to solve a set of instances $\mathbb{I} = \{I_1, ..., I_T\}$ in the training set and the resulting objective values h_{max} (the height of the highest rectangle piece in the packing solution) from all instances are recorded. Since the objective values obtained by a heuristic $\mathcal{H}$ for each instance are very different, we will measure the quality of a packing solution by the relative deviation of its objective value from its lower bound as shown in equation ([1]).

$$dev(\mathcal{H}, I_n) = \frac{Obj(\mathcal{H}, I_n) - LB(I_n)}{LB(I_n)} \tag{1}$$

In this equation, $Obj(\mathcal{H}, I_n)$ is the minimum h_{max} obtained by applying $\mathcal{H}$ to instance I_n, and $LB(I_n)$ is the lower bound for instance I_n. The fitness of $\mathcal{H}$ on the training set is calculated by equation ([2]).

$$dev_{average}(\mathcal{H}) = \frac{\sum_{I_n \in \mathbb{I}} dev(\mathcal{H}, I_n)}{|\mathbb{I}|} \tag{2}$$

2.3 The New GPHH Algorithm

Algorithm [1] shows how GP can be used to evolve search heuristics for 2D-SPP. A number of instances will be selected to train the evolved heuristics. In each generation, all heuristics in the population are evaluated by applying them to solve each training instance. For an instance, an evolved heuristic iteratively generates packing solutions as discussed in section [2.1] until the maximum number of iterations (*maxIteration*) is reached. The parameters for the algorithm are shown in Table 2. The initial GP population is created using the ramped-half-and-half method [19]. Tournament selection of size 7 is used to select an individual for the genetic operators. When solving each instance, the evolved heuristic will be applied with *maxIteration* of 100 or we stop when h_{max} is equal to the lower bound. Our training set is selected from the classes $\mathbb{N}1$–$\mathbb{N}8$ including randomly generated instances with known optimal solutions [11]. In order to obtain heuristics with good generality, we will use 10 instances from the classes $\mathbb{N}4$, $\mathbb{N}5$, and $\mathbb{N}6$ with 40, 50 and 60 pieces respectively which are shown to provide good results in [11]. There are $10 \times 3 = 30$ instances in total used to train the search heuristics.

Algorithm 1. GPHH to evolve search heuristics for 2D-SPP

```
load training instances $\mathbb{I} \leftarrow \{I_1, I_2, \ldots, I_T\}$
randomly initialise the population $P \leftarrow \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_{popsize}\}$
$\mathcal{H}^* \leftarrow null$, $fitness(\mathcal{H}^*) = +\infty$ and $generation \leftarrow 0$
while $generation \leq maxGeneration$ do
    foreach $\mathcal{H}_i \in P$ do
        foreach $I_k \in \mathbb{I}$ do
            $iteration \leftarrow 0$
            $obj^* = +\infty$
            while $iteration \leq maxIteration$ do
                $h_{max} \leftarrow$ use $\mathcal{H}_i$ to construct a packing solution for $I_k$
                if $h_{max} < obj^*$ then
                    $obj^* \leftarrow h_{max}$
                end
                if $h_{max} = LB(I_k)$ then
                    break
                end
                update ppX, ppY and pP
                $iteration \leftarrow iteration + 1$
            end
            $Obj(\mathcal{H}_i, I_k) \leftarrow obj^*$
        end
        evaluate $fitness(\mathcal{H}_i, I_k)$ by using equation (2)
        if $fitness(\mathcal{H}_i) < fitness(\mathcal{H}^*)$ then
            $\mathcal{H}^* \leftarrow \mathcal{H}_i$
            $fitness(\mathcal{H}^*) \leftarrow fitness(\mathcal{H}_i)$
        end
    end
    $P \leftarrow$ apply reproduction, crossover, mutation to $P$
    $generation \leftarrow generation + 1$
end
return $\mathcal{H}^*$
```

Table 2. Parameters of the proposed GPHH

Population Size	1000	Crossover rate	90%	Mutation rate	5%
Reproduction rate	5%	Generations	50	Max-depth	8

3 Results

Thirty independent runs of the proposed GPHH are performed and the search heuristic resulting from each run is recorded. These evolved heuristics are tested on a set of popular benchmark instances in the 2D-SPP literature [7,17,22]. The performance of the evolved heuristics are reported in Table 3 and compared to those of BF [7], BF+SA [10], GA+BLF and SA+BLF [17] (only the best result from these two algorithms is shown and referred to as MH in Table 3), R-GRASP [2], and SWO [9]. The results of BF+SA, R-GRASP and SWO are obtained from 60 second runs (for each instance) [11]. The average and the best results for each instance obtained by the 30 evolved heuristics with $maxIteration = 100$ are shown in the columns *Evolved*. The *Best Evolved* columns show the results from 60 second runs (coded in Java and run on Intel Core i5-2400 3.10 GHz CPUs, single thread) of the evolved heuristic $\mathcal{H}_{30}^{*}$ with the smallest fitness among 30 evolved heuristics.

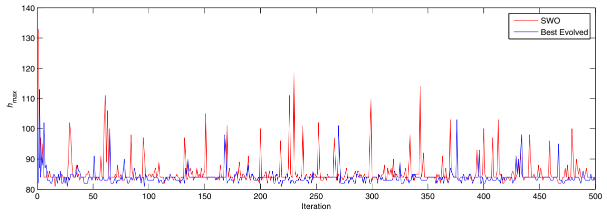
The first and second columns in Table 3 show the names of the 2D-SPP instances and their corresponding optimal h_{max}. The next three columns show the h_{max} obtained by BF, BF+SA, and MH. Two columns of R-GRASP present the average and the best h_{max} obtained by this method in 10 independent runs. For SWO and $\mathcal{H}_{30}^{*}$, h_{max}, $Iter_{best}$, and t_{best} respectively indicate the best objective value, the number of iterations (number of generated packing solution) and the time (in seconds) to obtain those values. It is noted that the evolved heuristics can outperform BF, BF+SA and MH in most instances. The evolved heuristics also show very competitive results as compared to R-GRASP and SWO. The results of the evolved heuristics are very close to the results obtained by R-GRASP and SWO and sometimes better (the bold values in Table 3 indicates that $\mathcal{H}_{30}^{*}$ provides results better than those obtained by other methods). Given that the evolved heuristics only runs for 100 iterations during the training phase, these results show that they are not only efficient but also very effective. When we allow the best evolved heuristic $\mathcal{H}_{30}^{*}$ to run longer, the obtained results are very interesting. For $N1$–$N13$ instances, $\mathcal{H}_{30}^{*}$ totally outperforms R-GRASP and SWO. It is also able to find the optimal solutions for N4 and N13, which cannot be found by R-GRASP and SWO. For $c1p1$–$c7p3$ and the nice and path instances, $\mathcal{H}_{30}^{*}$ is still very competitive compared to R-GRASP and SWO and sometimes better than the best results obtained by these two algorithms.

Regarding the computation time, $\mathcal{H}_{30}^{*}$ is very efficient. Since we only allow the the evolved heuristics to run for 100 iterations during the training/evolving phase, this restricted computation budget force the evolved heuristics to be more effective through their search. In most cases, $\mathcal{H}_{30}^{*}$ can find the best solutions in much fewer iterations compared to SWO. This observation shows that the incorporation of the construction heuristic and the iterative procedure in the evolved heuristics can help guide the search more effectively; and therefore can reduce the computational effort. Moreover, similar to SWO, the evolved heuristics are also deterministic and only require one run compared to stochastic approaches such as BF+SA, R-GRASP or MH, which require multiple runs to obtain the best result.

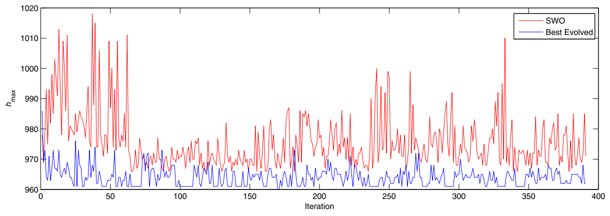
Table 3. Performance of evolved heuristic on the zero waste instances

Inst.	*Opt.*	*BF*	*BF*	*MH*	*R − GRASP*		*SWO* (60s)			*Evolved*		*Best Evolved* (60s)		
name			*+SA*		*mean*	*best*	h_{max}	$Iter_{best}$	t_{best}	*mean*	*best*	h_{max}	$Iter_{best}$	t_{best}
$N1$	40	45	40	40	40	40	40	2	< 0.1	40	40	40	1	< 0.01
$N2$	50	53	50	51	50	51	50	2888	0.5	50.9	50	50	159	0.08
$N3$	50	52	51	52	51	51	50	1259185	41.9	51.1	51	50	6639	0.23
$N4$	80	83	82	83	81	81	81	16	< 0.1	81.6	81	**80**	12447	0.48
$N5$	100	105	103	106	102	102	101	558000	41.8	103.1	102	101	15811	0.86
$N6$	100	103	102	103	101	101	101	4065	0.8	101.6	101	101	353	0.03
$N7$	100	107	104	106	101	101	101	55	0.2	101.3	101	101	102	0.01
$N8$	80	84	82	85	81	81	81	52446	8.3	82	81	81	635	0.08
$N9$	150	152	152	155	151	151	151	14819	3.6	151.6	151	151	107	0.02
$N10$	150	152	152	154	151	151	151	329	0.9	151.6	151	151	70	0.04
$N11$	150	152	153	155	151	151	151	81	0.7	151.1	151	151	9	0.01
$N12$	300	306	306	312	303.2	303	304	128	0.9	302.3	301	**302**	26	0.09
$N13$	960	964	964	–	963	963	966	65	9.0	961.9	961	**960**	170	20.48
$c1p1$	20	21	20	20	20	20	20	26	< 0.1	20.1	20	20	42	0.02
$c1p2$	20	22	20	21	20	20	21	11	0.01	21	20	20	41	0.02
$c1p3$	20	24	20	20	20	20	20	2	< 0.1	20	20	20	5	< 0.01
$c2p1$	15	16	16	16	15	15	16	12	< 0.1	15.8	15	16	1	< 0.01
$c2p2$	15	16	16	16	15	15	15	1312	5.0	15.9	15	16	2	< 0.01
$c2p3$	15	16	16	16	15	15	15	17	< 0.1	15.2	15	15	1	< 0.01
$c3p1$	30	32	31	32	30	30	30	6360	0.7	31.1	31	31	26	< 0.01
$c3p2$	30	34	31	32	31	31	31	22	< 0.1	31.4	31	31	60	< 0.01
$c3p3$	30	33	31	32	30	30	30	7168	0.6	31	30	31	13	< 0.01
$c4p1$	60	63	61	64	61	61	61	2360	0.4	61.9	61	61	709	0.04
$c4p2$	60	62	61	63	61	61	61	774	0.5	62.2	61	61	450	0.02
$c4p3$	60	62	61	62	61	61	61	290	0.6	61.3	61	61	74	< 0.01
$c5p1$	90	93	91	94	91	91	91	13	< 0.1	91.5	91	91	65	0.01
$c5p2$	90	92	91	95	91	91	91	1922	0.9	91.9	91	91	417	0.04
$c5p3$	90	93	92	95	91	91	91	8594	1.4	91.7	91	91	147	0.02
$c6p1$	120	123	122	127	121.9	121	122	137	0.5	122.2	122	121	1101	0.18
$c6p2$	120	122	121	126	121.9	121	121	2114	1.4	122.6	121	121	1340	0.20
$c6p3$	120	124	122	126	121.9	121	122	544	0.7	122.2	121	121	1052	0.17
$c7p1$	240	247	244	255	244	244	243	22142	15.2	244.1	243	**242**	3934	2.26
$c7p2$	240	244	244	251	242.9	242	242	2352	2.3	243.5	242	**241**	51508	26.02
$c7p3$	240	245	245	254	243	243	243	1632	1.8	244	243	**242**	1659	0.88
$Nice1$	100	107.4	104	108.2	103.9	103.7	103.7	388299	12.3	106.8	101.4	**101.4**	136	0.09
$Nice2$	100	108.5	104.4	112	104.7	104.6	104.9	253034	21.7	107.6	106	**103.9**	282775	17.72
$Nice3$	100	107	105	113	104.5	104	104.6	51117	14.1	106	104.6	**103.8**	18036	3.66
$Nice4$	100	105.3	104.7	113.2	103.8	103.6	103.8	50353	48.3	104.9	103.6	**102.8**	2077	1.38
$Nice5$	100	103.5	103.5	111.9	102.4	102.2	103.3	9607	49.2	103.7	102.2	**102.1**	2553	9.66
$Nice6$	100	103.7	103.8	–	102.3	102.2	102.9	857	19.9	103.1	101.8	**101.7**	672	10.14
$Path1$	100	110.1	103.1	106.7	104.2	104.2	106.9	4106	0.8	109.1	106.8	106.9	15897	0.26
$Path2$	100	113.8	103.4	107	101.9	101.8	101.7	727801	58.4	103	101.8	102.5	7302	0.32
$Path3$	100	107.3	103	109	102.7	102.6	102.9	19687	6.6	104.9	103.6	104.5	699	0.10
$Path4$	100	104.1	103.4	108.8	102.3	102	102	31767	30.7	103.8	102.6	103.1	1902	0.98
$Path5$	100	103.7	103.5	111.1	103.2	103.1	103.2	4829	24.6	104.5	103.4	103.4	567	1.97
$Path6$	100	102.8	102.9	–	102.7	102.5	102.8	2904	59.8	104	102.6	102.5	694	8.39

To better understand the evolved heuristic, in Fig. 2(b) we have plotted the values of h_{max} obtained through the search of the evolved heuristic $\mathcal{H}^*_{30}$ against those obtained by SWO. In the two benchmark instances $N4$ and $N13$, although their h_{max} values at the few first iterations are quite similar, there are a lot more disturbances in the search of SWO compared to $\mathcal{H}^*_{30}$. It is noted that $\mathcal{H}^*_{30}$ also has disturbances in its search but the frequency as well as the magnitudes of these disturbances are much smaller than those of SWO. This feature allows $\mathcal{H}^*_{30}$ to balance between its exploration and exploitation abilities, which makes it more effective and more efficient than SWO.



(a) Instance $N4$



(b) Instance $N13$

Fig. 2. Behaviours of SWO and the best evolved heuristic $\mathcal{H}^*_{30}$

4 Conclusions

This paper proposed a novel GPHH method for evolving optimisation search heuristics for 2D-SPP. The key idea of this method is the representation of the evolved heuristics in GP which helps the evolved heuristics to handle both the construction and the iterative search procedure simultaneously. The experimental results showed that the evolved heuristics can efficiently solve 2D-SPP and the results obtained by the evolved heuristics are very competitive with, and sometimes better than the state-of-the-art search methods for 2D-SPP. In several instances, the evolved heuristics can also find optimal or near optimal results that cannot be found by other methods. This is the first time that evolved heuristics automatically generated by a GPHH can compete with the state-of-the-art methods for 2D-SPP. In future studies, we will perform an extensive analysis of the evolved heuristics to understand better how they can solve the 2D-SPP and how to enhance their performance. Also, we will extend this work to deal with other hard combinatorial optimisation problems, e.g., three dimensional bin packing, job scheduling, and time-tabling.

References

1. Aggoun, A., Beldiceanu, N., Carlsson, M., Fages, F.: Integrating rule-based modelling and constraint programming for solving industrial packing problems. ERCIM News 2010(81) (2010)
2. Alvarez-Valdes, R., Parreño, F., Tamarit, J.M.: Reactive GRASP for the strip-packing problem. Computers and Operations Research 35(4), 1065–1083 (2008)

3. Babu, A.R., Babu, N.R.: Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms. International Journal of Production Research 37(7), 1625–1643 (1999)
4. Baker, B.S., Coffman, E.G., Rivest, R.L.: Orthogonal packings in two dimensions. SIAM Journal on Computing 9, 846–855 (1980)
5. Belov, G., Scheithauer, G., Mukhacheva, E.A.: One-dimensional heuristics adapted for two-dimensional rectangular strip packing. Journal of the Operational Research Society 59, 823–832 (2007)
6. Burke, E.K., Hyde, M.R., Kendall, G.: Grammatical evolution of local search heuristics. IEEE Transactions on Evolutionary Computation (2011) (to appear)
7. Burke, E.K., Kendall, G., Whitwell, G.: A new placement heuristic for the orthogonal stock-cutting problem. Operations Research 52(4), 655–671 (2004)
8. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Tech. Rep. Computer Science Technical Report No. NOTTCS-TR-SUB-0906241418-2747, School of Computer Science and Information Technology, University of Nottingham (2010)
9. Burke, E.K., Hyde, M.R., Kendall, G.: A squeaky wheel optimisation methodology for two-dimensional strip packing. Computers and Operations Research 38(7), 1035–1044 (2011)
10. Burke, E.K., Kendall, G., Whitwell, G.: A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem. INFORMS Journal on Computing 21(3), 505–516 (2009)
11. Burke, E., Hyde, M., Kendall, G., Woodward, J.: A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics. IEEE Transactions on Evolutionary Computation 14, 942–958 (2010)
12. Chazelle, B.: The bottom-left bin-packing heuristic: An efficient implementation. IEEE Transactions on Computers 32(8), 697–707 (1983)
13. Christofides, N., Whitlock, C.: An algorithm for two-dimensional cutting problems. Operations Research 25(1), 30–44 (1977)
14. Fukunaga, A.: Automated discovery of local search heuristics for satisfiability testing. Evolutionary Computation 16, 21–61 (2008)
15. Gilmore, P.C., Gomory, R.E.: A linear programming approach to the cutting-stock problem. Operations Research 9(6), 849–859 (1961)
16. Hifi, M., Zissimopoulos, V.: A recursive exact algorithm for weighted two-dimensional cutting. European Journal of Operational Research 91(3), 553–564 (1996)
17. Hopper, E., Turton, B.: An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. European Journal of Operational Research 128(1), 34–57 (2001)
18. Jakobs, S.: On genetic algorithms for the packing of polygons. European Journal of Operational Research 88(1), 165–181 (1996)
19. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press (1992)
20. Lodi, A., Martello, S., Vigo, D.: Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. INFORMS Journal on Computing 11(4), 345–357 (1999)
21. Lodi, A., Martello, S., Vigo, D.: Recent advances on two-dimensional bin packing problems. Discrete Applied Mathematics 123, 379–396 (2002)
22. Mumford-Valenzuela, C.L., Vick, J., Wang, P.Y.: Heuristics for large strip packing problems with guillotine patterns: an empirical study. In: Resende, M.G.C., de Sousa, J.P., Viana, A. (eds.) Proceedings of the Metaheuristics International Conference (MIC 2001), pp. 501–522 (2001)

Solving Graph Coloring Problem by Fuzzy Clustering-Based Genetic Algorithm

Young-Seol Lee and Sung-Bae Cho

Dept. of Computer Science, Yonsei University, Seoul 120-749, Korea
tiras@sclab.yonsei.ac.kr, sbcho@cs.yonsei.ac.kr

Abstract. The graph coloring problem is one of famous combinatorial optimization problems. Some researchers attempted to solve combinatorial optimization problem with evolutionary algorithm, which can find near optimal solution based on the evolution mechanism of the nature. However, it sometimes requires too much cost to evaluate fitness of a large number of individuals in the population when applying the GA to the real world problems. This paper attempts to solve graph coloring problem using a fuzzy clustering based evolutionary approach to reduce the cost of the evaluation. In order to show the feasibility of the method, some experiments with other alternative methods are conducted.

Keywords: cluster based GA, graph coloring, fuzzy clustering.

1 Introduction

The graph coloring problem is one of the most famous NP-hard problems. The problem requires assignment of colors to each vertex in a given graph with a constraint that two colors assigned to two adjacent vertices must be different. The key point of the problem is to find a minimal number of colors for the color assignment. Some practical applications for optimal resource assignment are related to graph coloring problem. Because of the NP-completeness of the coloring problem, many heuristic methods [1, 2, 3] have been developed.

Genetic algorithm is one of the best solutions to solve the graph coloring problem, and it was often used to find optimal solution for many problems such as traveling salesman problem [4], the quadratic assignment problem [5] and the bin-packing problem [6] and have shown competitive performance. However, genetic algorithm requires large population size and much cost to evaluate the population to discover optimal solution. In some cases with expensive evaluation cost such as game AI, robot AI and interactive GA, it is difficult to evaluate a large population in practical time. Some researchers have developed a method to estimate fitness values of the whole population by evaluating the part of population [7]. The main point here is to select the part of population and reduce evaluation cost for the graph coloring problem.

In this paper, we present a fuzzy clustering-based genetic algorithm to reduce the cost of coloring problem. It evaluates only the part of individuals and estimates fitness values of similar individuals with fuzzy integrals. This algorithm allows us to reduce fitness evaluation cost, while maintaining the performance.

2 Background

2.1 Graph Coloring

Graph coloring is an optimization problem to determine a minimum number (the *chromatic number*) of color classes C_1, C_2, ..., C_k in an undirected graph G which has two components such as nodes $V= \{v_1, ..., v_n\}$, and edges $E =\{e_{ij} \mid \exists$ an edge between v_i and $v_j\}$. The constraint is that v_i and v_j are not in the same color class for each edge $e_{ij} \in E$ [8].

Suppose that $c(v_i)$ be the color (represented by a positive integer) assigned to the node v_i, a proper coloring satisfies the constraint:

$$\in \qquad \in e_{ij} \in E, c(v_i) \in c(v_j) \tag{1}$$

In many cases, a random graph, which includes randomly generated edges between nodes with given density $d \in [0,1]$, is used to evaluate the performance of an algorithm to solve graph coloring. It is difficult to color optimally random graphs having more than 100 nodes [9]. In this paper, we also use random graphs for performance test.

2.2 Clustering Algorithms

Clustering algorithm is to group similar data items automatically [10] without any prior knowledge. These groups are named as clusters. Cluster analysis has been applied in many fields such as image analysis and data mining [11, 12]. There are three general categories of clustering techniques: hierarchical clustering, partitional clustering, and overlapping clustering.

Hierarchical clustering algorithm constructs hierarchical structures within clusters. There are two different approaches for hierarchical clustering such as bottom-up approach and top-down approach. Bottom-up approach starts with n clusters, and repeats to merge similar clusters. Top-down approach divides one cluster with all data items into some clusters [10, 13]. There are several hierarchical clustering algorithms such as single-linkage algorithm, complete-linkage algorithm, average-linkage algorithm, and Ward's method.

Partitional clustering usually generates clusters that partition the data into similar groups. The goal of the algorithms is to assign data items that are close together into a cluster. In many partitional algorithms, the number of clusters is determined in advance. K-means clsutering and hard c-means (HCM) clustering are good examples of partitional clustering [10, 13].

Overlapping clustering is similar to partitional clustering except for no discrete clusters. In overlapping clustering, each cluster can be overlapped with others and an item can belong to more than one cluster. Fuzzy c-means (FCM) algorithm and b-clump algorithm are included in this category [14, 15].

2.3 Related Works

There are many researchers to develop heuristic methods for a graph coloring problem. Omari *et al.* developed new heuristic graph coloring algorithms based on known heuristic algorithms which are the Largest Degree Ordering (LDO) and Saturation Degree Ordering (SDO) [1]. Hertz *et al.* used tabu search to solve the coloring problem [3]. It avoided cycling and local minima using tabu list of forbidden movements. Similarly, Chams *et al.* applied a simulated annealing algorithm to the graph coloring problem in [16]. Brelaz proposed heuristic methods to color the vertices of a graph which rely on the comparison of the degrees and structure of a graph [2]. Lotfi *et al.* developed a heuristic based graph coloring algorithm with little computational effort to deal with large scale scheduling problems [17]. Johnson *et al.* applied a simulated annealing to solve graph coloring problem [9]. Klotz *et al.* developed a heuristic algorithm using backtracking [18].

Some researchers have studied to find solutions using evolutionary algorithms. Porumbel *et al.* presented a hybrid evolutionary algorithm (named as Evocol) for the graph coloring [19]. Eiben *et al.* used adaptive evolutionary algorithm that periodically changes the fitness function during evolution [20]. Fleurent *et al.* proposed a hybrid algorithm combining genetic algorithm and tabu search [7]. It improved random mutation by tabu search and developed a new crossover operator based on conflicting nodes (adjacent nodes having the same color). The hybrid algorithm has shown good results on the benchmark tests. However, it takes too much computing times to obtain the results for some large instances. Costa *et al.* compared sequential algorithm and evolutionary algorithm and presented a hybrid algorithm called *EDM* with two different approaches [21].

3 Proposed Method

The whole algorithm repeats some operations for evolution process. It firstly generates initial population and applies clustering technique to the population before evaluation of them. Only one individual in each cluster is evaluated to estimate fitness to reduce the cost [22]. The individual is decided by the centroid in the cluster. Fitness values of all the population are calculated from the membership of each cluster. The above process is iterated until optimal solution is found or the number of iteration exceeds the fixed number. The algorithm is applied to other domains [23, 24]. It is similar to general evolutionary approach except for *clustering* as shown in Fig. 1.

3.1 Encoding and Fitness Function

There are some encoding schemes such as order-based encoding and string-based encoding as reported in [7]. The string-based encoding is used because it usually

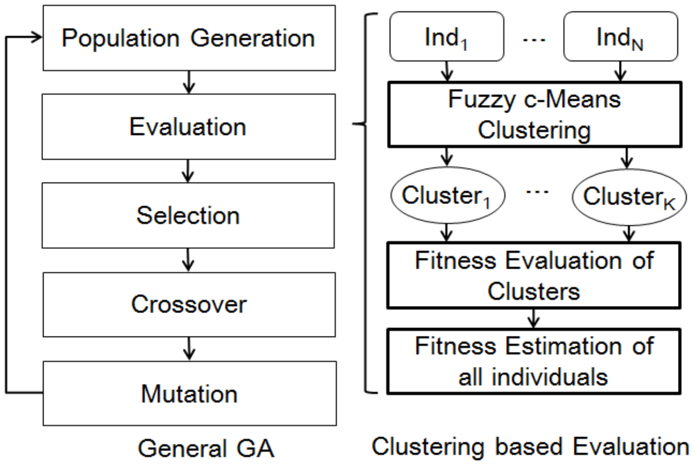


Fig. 1. Fuzzy clustering based evaluation

generates smoother change of fitness than order-based encoding. In the encoding scheme, an individual s for a graph $G=(V, E)$ with n nodes and k the number of available colors is denoted by $s=<c(v_1), c(v_2), ..., c(v_n)>$ which corresponds to an assignment of the k colors to the nodes of the graph. Fig. 2 shows an example of string-based encoding, where v_i denotes the ith node in the graph and 1, 2 and 3 represent first, second, and third colors, respectively. $|S|$, which is the size of the search space S, is k^n and it can become very large if the graph has more than 100 nodes.

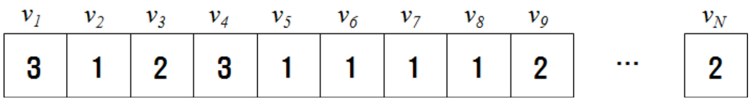


Fig. 2. An example of string-based gene encoding with 3 colors

For each individual s, the fitness $f(s)$ can be calculated with the number of violated color constraints as shown in (2).

$$f(s) = \sum_{(v_i,v_j)\in E} c(v_i,v_j) \text{ where } c(v_i,v_j) = \begin{cases} 1 \text{ if } c(v_i)=c(v_j) \\ 0 \text{ elsewhere} \end{cases} \tag{2}$$

where v_i and v_j are the ith and the jth nodes in a given graph and $c(v_i)$ and $c(v_j)$ represent the colors of the ith and the jth nodes, respectively. The algorithm aims to reduce $f(s)$ until $f(s)=0$ for the fixed k to solve a k-coloring problem. 1 point crossover is used and a color of nodes in gene code is changed by mutation.

3.2 Fitness Estimation Using Fuzzy c-means Clustering

In order to reduce evaluation cost, we separate individuals into several groups and evaluate only one individual in each group. A fuzzy clustering algorithm is used to separate the individuals instead of a hard clustering algorithm. The fuzzy clustering approach is more likely to overcome the local minimum than hard clustering approach because it makes soft boundaries among clusters through the use of fuzzy membership values [11].

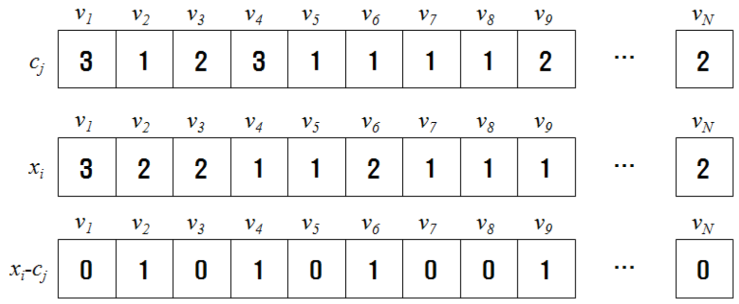
The fuzzy c-means algorithm is the most widely-used fuzzy clustering algorithm proposed by Bezdeck [25]. It provides fuzzy membership values which mean how much each individual belongs to a specific cluster. The fuzzy membership has a value between 0 and 1. If the value is closer to 0, it indicates a weaker association to the corresponding cluster. On the contrary, the value closer to 1 indicates a stronger association to the cluster. It is based on minimization of the objective function in (3).

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \| x_i - c_j \|^2, \quad 1 \le m \le \infty \tag{3}$$

where m is any real number greater than 1, u_{ij} is the degree of membership of x_i in the cluster j, x_i is the ith of d-dimensional measured data, c_j is the d-dimension center of the cluster and $\|*\|$ is any norm expressing the similarity between any measured data and the center. In this case, $\|x_i - c_j\|$ means the distance between centroid c_j of the jth cluster and an individual x_i. It is calculated as shown in (4).

$$\| x_i - c_j \| = \sum_{l=1}^{N} c(x_i^l, c_j^l) \ \text{where}\ c(x_i^l, c_j^l) = \begin{cases} 1 \ \text{if}\ c(x_i^l) = c(c_j^l) \\ 0 \ \text{elsewhere} \end{cases} \tag{4}$$

$$x_i^l : l\text{th node of } x_i, \quad c_j^l : l\text{th node of } c_j$$



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	…	v_N
c_j	3	1	2	3	1	1	1	1	2	…	2
x_i	3	2	2	1	1	2	1	1	1	…	2
x_i-c_j	0	1	0	1	0	1	0	0	1	…	0

Fig. 3. An example of distance between an individual x_i and a cluster j

Fig. 3 shows an example of the calculation of the distance which is based on the comparison of colors of nodes between c_j and x_i.

Fuzzy partitioning is carried out through an iterative optimization of the objective function shown previously, with the update of membership u_{ij} and the cluster centers c_j by (5).

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|} \right)^{\frac{2}{m-1}}}, \quad c_{ij} = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{i=1}^{N} u_{ij}^m} \tag{5}$$

The iteration stops when $max_{ij}\{|u_{ij}^{(k+1)}-u_{ij}^{(k)}|\}<\varepsilon$, where ε is a termination criterion between 0 and 1, whereas k denotes the iteration step. This procedure converges to a local minimum or a saddle point of J_m.

The fitness values of all individuals are estimated from the fitness of the part of the individuals and the similarity between the individuals using fuzzy integrals [13], which are the integrals of a real function with respect to a fuzzy measure.

Let $X=\{x_1, x_2, ..., x_n\}$ be a set of individuals in the population and $C=\{c_1, c_2, ..., c_c\}$ is a set of clusters, and the fitness values of the cluster centers $F=\{f_1, f_2, ..., f_c\}$. The fitness values of an individual x_i can be estimated based on m_{ki} that means the degree of membership value of the ith individual to the kth cluster center. m_{ki} represents the membership of a cluster as a value between 0 and 1. Because the number of clusters is discrete, the fuzzy integral of k can be calculated by the sum of the values. The estimated fitness value of e_i is as the following equation:

$$e_i = \sum_{k=1}^{c} m_{ki} \times f_k \tag{6}$$

4 Experimental Results

4.1 Experimental Settings

A randomly generated graph is used to perform experiments. It is illustrated with a matrix which includes cells with 1 or 0 as shown in Fig. 4. ‘1’ in the matrix means that the pair of nodes is connected and 0 means not connected. The graph in the experiments has 150 nodes and the edges between two nodes are randomly generated with a fixed probability with 0.05.

Table 1 summarizes the parameters of genetic algorithms in the experiments. To compare the performance with alternative methods, various methods summarized in Table 2 are applied to coloring problem in a given random graph. Fuzzy c-means is different from other clustering methods due to including fuzziness of the membership [24].

4.2 Experimental Results

Fig. 5 and 6 show the comparison of fitness changes of an experiment with number of color k = 5. In these figures, x axis represents the number of generations and y axis

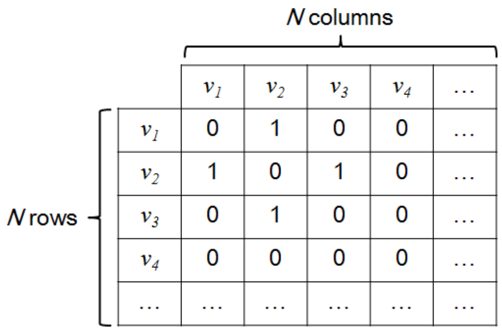


Fig. 4. An example of a randomly generated graph for graph coloring

Table 1. Experimental Environment

Environment	Value	Environment	Value
Length of chromosome	150	Fuzziness parameter	1.2
Crossover rate	0.9	Number of clusters	10
Mutation rate	0.005	Number of colors	$5 \le n \le 50$
Max generation	300		

Table 2. Methods in experiments

Methods	Description
FCM	**Fuzzy c-means clustering based GA with population size 100 and cluster size 10**
STD	Standard GA with population size 100
KMS	K-means clustering based GA with population size 100 and cluster size 10
SMP	Standard GA with population size 10
HCM	Hard c-means clustering based GA with population size 100 and cluster size 10
SLK	Single-linkage clustering based GA with population size 100
ALK	Average-linkage clustering based GA with population size 100
CLK	Complete-linkage clustering based GA with population size 100

denotes the number of conflicts. Each line illustrates a method to solve graph coloring problem.

As expected, standard genetic algorithm (STD) finds the minimum number of colors among various methods. However, it takes so much evaluation time to discover the solution. It requires almost ten times more evaluation than other methods. The proposed method (FCM) shows the second minimum colors and the second minimum time. Standard genetic algorithm with small population is not suitable to solve this problem because it cannot yield competitive solutions. Hard clustering techniques such as hard c-means clustering (HCM) and k-means clustering (KMS) can generate their solution for short evaluation time, but they show worse performance than FCM. The proposed method is a reasonable and competitive solution in some problems which require too much evaluation cost. Table 3 summarizes minimum number of conflicts and average time for three runs.

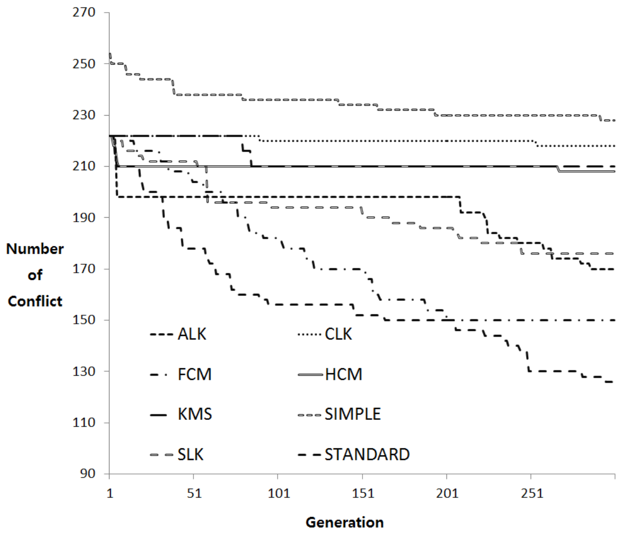


Fig. 5. Maximum fitness change with color k=5

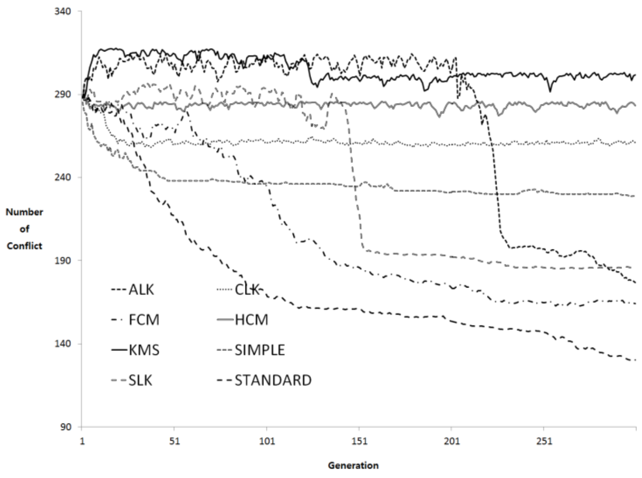


Fig. 6. Average fitness change with color k=5

Table 3. Summary of experiments

	Minimum colors	Avg. time	Diff. time with STD	Avg. conflicts	Diff. conflicts with STD
FCM	**24**	**230.53**	**1516.91**	**19.73**	**10.09**
STD	16	1747.44	0	9.64	0
KMS	27	180.76	1566.68	31.16	21.52
SMP	≥ 50	245.67	1501.77	40.13	30.49
HCM	28	242.85	1504.59	29.16	19.52
SLK	24	258.20	1489.24	23.69	14.05
ALK	27	256.57	1490.87	25.47	15.83
CLK	28	244.97	1502.47	26.67	17.03

5 Concluding Remarks

This paper proposed an efficient genetic algorithm to solve graph coloring problem in short time. The method decreases the evaluation cost by reducing the number of evaluation using fuzzy clustering technique. It generates some clusters from all individuals, and evaluates only one individual in each cluster. The fitness of the other individuals is indirectly estimated by the evaluated fitness and membership value. The proposed method shows competitive performance to standard genetic algorithm and the alternative methods.

There are some problems to be solved on the proposed method. The performance of the method is still lower in comparison with the standard genetic algorithm. We have to raise the performance to a similar level of standard genetic algorithm. A novel fuzzy integral including heuristics will become an important issue to be considered as a future work. Also, we will compare other methods like Porumbel, Galinier and Chiarandini's works and confirm a statistical validation of the comparison.

Acknowledgments. This research was supported by the KCC(Korea Communications Commission), Korea, under the R&D program supervised by the KCA(Korea Communications Agency) (KCA-2012-12-911-04-005).

References

1. Omari, H.A., Sabri, K.E.: New graph coloring algorithms. J. Mathematics and Statistics 2(4), 439–441 (2006)
2. Brelaz, D.: New methods to color vertices of a graph. Communcations of ACM 22, 251–256 (1979)
3. Hertz, A., De Werra, D.: Using tabu search techniques for graph coloring. Computing 39, 345–351 (1987)
4. Freisleben, B., Merez, P.: New Genetic Local Search Operators for the Traveling Salesman Problem. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 890–899. Springer, Heidelberg (1996)

5. Merez, P., Freisleben, B.: A genetic local search approach to the quadratic assignment problem. In: 7th International Conference on Genetic Algorithms, pp. 465–472 (1997)
6. Falkenauer, E.: A hybrid grouping genetic algorithm for bin-packing. J. Heuristics 2(1), 5–30 (1996)
7. Fleurent, C., Ferland, J.A.: Genetic and hybrid algorithms for graph coloring. Annals of Operations Research 63, 437–463 (1995)
8. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization - Algorithms and Complexity. Prentice Hall (1982)
9. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: an experimental evaluation: part ii, graph coloring and number partitioning. Operations Research 39(3), 378–406 (1991)
10. Gose, E., Johnsonbaugh, R., Jost, S.: Pattern Recognition and Image Analysis. Prentice Hall (1996)
11. Anderberg, M.R.: Cluster Analysis for Applications. Academic Press (1973)
12. Fukunaka, K.: Introduction to Statistical Pattern Analysis. Academic Press (1990)
13. Haritigan, J.A.: Clustering Algorithms. John Wiley & Sons (1975)
14. Hoppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. John Wiley & Sons (1999)
15. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. IEEE Trans. of Pattern Analysis and Machine Intelligence, PAMI-13(8), 841-847 (1991)
16. Chams, M., Hertz, A., De Werra, D.: Some experiments with simulated annealing for coloring graphs. European Journal of Operational Research 32, 260–266 (1987)
17. Lotfi, V., Sarin, S.: A graph coloring algorithm for large scale scheduling problems. Computers & Operations Research 13(1), 27–32 (1986)
18. Klotz, W.: Graph coloring algorithms. IEICE Trans. Information and Systems 5, 1–9 (2002)
19. Porumbel, D.C., Hao, J.-K., Kuntz, P.: Diversity Control and Multi-Parent Recombination for Evolutionary Graph Coloring Algorithms. In: Cotta, C., Cowling, P. (eds.) EvoCOP 2009. LNCS, vol. 5482, pp. 121–132. Springer, Heidelberg (2009)
20. Eiben, A.E., Van Der Hauw, J.K., Van Hemer, J.I.: Graph coloring with adaptive evolutionary algorithms. J. of Heuristics 4(1), 25–46 (1998)
21. Costa, D., Hertz, A., Dubuis, O.: Embedding a sequential procedure within an evolutionary algorithms for coloring problems in graphs. J. of Heuristics 1(1), 105–128 (1995)
22. Jin, Y., Sendhoff, B.: Reducing Fitness Evaluations Using Clustering Techniques and Neural Network Ensembles. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 688–699. Springer, Heidelberg (2004)
23. Kim, H.-S., Cho, S.-B.: An efficient genetic algorithms with less fitness evaluation by clustering. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 887–894. IEEE (2001)
24. Yoo, S.-H., Cho, S.-B.: Partially Evaluated Genetic Algorithm Based on Fuzzy c-Means Algorithm. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN VIII. LNCS, vol. 3242, pp. 440–449. Springer, Heidelberg (2004)
25. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press (1981)
26. Chen, X.S., Ong, Y.S., Lim, M.H., Tan, K.C.: A Multi-Facet Survey on Memetic Computation. IEEE Transactions on Evolutionary Computation 15(5), 591–607 (2011)
27. Ong, Y.S., Lim, M.H., Chen, X.S.: Research Frontier: Memetic Computation - Past, Present & Future. IEEE Computational Intelligence Magazine 5(2), 24–36 (2010)

Efficient Neuroevolution for a Quadruped Robot

Xu Shengbo, Hirotaka Moriguchi, and Shinichi Honiden*

Department of Computer Science
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
{s-jyo,hmori,honiden}@nii.ac.jp

Abstract. In this research, we investigate whether CoSyNE and CMA-NeuroES algorithms can efficiently optimize neural policy of a quadruped robot. Both of these algorithms are proven to optimize connection weights efficiently on Pole Balancing benchmark. Due to their good results on that benchmark, they are expected to be efficient on other control problems like gait generation. In this research we experimentally show that CMA-NeuroES have higher scalability to optimize Artificial Neural Networks for generating gaits of quadruped robots in comparison with CoSyNE. The results can be helpful for researchers and practitioners to choose the optimal Neuroevolution algorithm for generating gaits.

Keywords: neural network, neuroevolution, CoSyNE, CMA-ES, Simplex, evolution, CMA-NeuroES.

1 Introduction

Various research studies have been done on generating gaits of quadruped robots with Artificial Neural Networks (ANNs) [1–3], especially optimizing neural policy (controller) of legged robots using Neuroevolution (NE) algorithms has been proven effective [1,2]. These NE algorithms evolve connection weights and network architecture via evolutionary algorithms [4,5].

However, it requires a large amount of trial-and-error to optimize networks with Neuroevolution algorithms. Since one episode takes a few tens of seconds, repeating trial-and-error on physical robots is a daunting task for human. Therefore, methods to evolve neural policies efficiently (i.e. with smaller number of episodes) are desirable.

In comparative research studies, two NE algorithms, CoSyNE and CMA-NeuroES are proven to optimize connection weights of ANN efficiently on the Pole Balancing (PB) benchmark [6,7]. In PB benchmark NE algorithms are required to optimize ANNs so that they can stabilize a single or multiple poles on a wheeled cart by applying external force to the cart. The performance of an algorithm is measured by the number of episodes it takes to achieve the task. The PB benchmark abstracts a general unstable control problem and has been used for over 30 years [6]. Algorithms that worked well on it have been expected

* Shinichi Honiden is also affiliated to National Institute of Informatics, Japan.

to be efficient on other control problems as well, although more complicated problems like Octpus and Helicoptor have recently been proposed as benchmark problems.

In this research, we investigate whether CoSyNE and CMA-NeuroES can efficiently optimize ANNs for generating gaits of quadruped robots. In other words the objective of this paper is to investigate the scalability of CoSyNE and CMA-NeuroES. To date, no research study has applied these two algorithms to generate gaits. Therefore our contribution will help researchers and practitioners to choose the right NE algorithm to use for generating gaits.

The quadruped robot we used was simulated in Bullet Physics simulator[1](Fig.1). NE algorithms are required to optimize connection weights of a fixed topology network in order to maximize the walking speed of quadrupeds. Since optimization of connection weights can be achieved by function optimization of real valued function, any off-the-shelf function optimization algorithm can be applied. For comparison, we applied Nelder-Mead Simplex algorithm [8] and Random Search as baselines. We compared CMA-NeuroES with CoSyNE in terms of efficiency in the results until 200 episodes and ability in the results until 10,000 episodes. From these two comparisons, we conclude whether there is scalability for optimizing a practical ANNs for generating gaits using these two algorithms, or not.

Experimental results show that CMA-NeuroES generates fast-walking gaits efficiently. On the other hand, CoSyNE does not scale for this practical problem. Furthermore, we discuss why scalability differs in both cases.

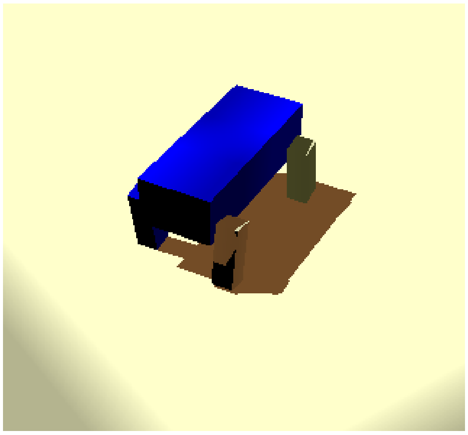


Fig. 1. The quadruped modeled within Bullet Physics

[1] It is physical simulator which has noise. http://bulletphysics.org/wordpress/

2 Related Work

Many research studies have been conducted on automatically generating gaits of quadruped robots with EC [1–3,9].

Hornby et al. [9] proposed to parameterize the gait such as the swing time for each leg. They optimized such gait parameters of AIBO, a commercial robot developed by Sony, by means of EC. In [1–3], researchers used ANNs as policies of quadruped robots and evolved them using NE algorithms. In both lines of research studies, it was proven that gaits generated with EC outperformed the ones designed by human hands.

Although, Yosinski et al. [1] and Clune et al. [2] proposed to use HyperNEAT [10] to generate gaits, we do not compare them on this work. This is because HyperNEAT is designed to represent and evolve large-scale ANNs [10] and not to evolve ANNs efficiently. We rather focus on CMA-NeuroES and CoSyNE that are proven to be efficient for benchmark tasks in this research.

3 Algorithms

In this section, we describe the four compared algorithms. All algorithms try to optimize connection weights of ANNs. We assume that the network topology is fixed.

3.1 CMA-NeuroES [7,11]

CMA-NeuroES is an NE algorithm based on covariance matrix adaptation evolution strategy (CMA-ES), a reputedly efficient function optimization algorithm. The search via CMA-ES is mutation-based [11]. Search points are updated with the following mutation:

$$x_l^{k+1} = m^k + \sigma^k z_l^k, \tag{1}$$

where k is the generation number and x_l^k is an individual in the k-th generation. $z_l^k \sim N(0, \mathbf{C}^k)$ is a random vector generated with multivariate normal distribution parameterized with zero mean, and covariance matrix $\mathbf{C}^k$. σ^k is the scale of distribution, and m^k is the weighted average of top-ranked individuals. As the individuals evolve, the covariance matrix $\mathbf{C}$ and step-size parameter σ are adapted to maximize the likelihood of repeating previous successful steps. You may refer to [7] for more detailed explanation of CMA-NeuroES.

CMA-NeuroES is known to outperform other evolutionary algorithms such as CoSyNE in the PB benchmark [7].

In the experiment, all parameters other than σ_{init} are set according to [11]. As for initial value of step-size parameter $\sigma_{init} = \sigma^0$, we tested two values $\sigma_{init} = 1$ and $\sigma_{init} = 5$.

3.2 CoSyNE [6]

In CoSyNE, each individual ($\mathbf{x}_1, \cdots, \mathbf{x}_m$) consists of a set of connection weights that parameterize a predefined ANN with fixed topology. Fig.2 shows the example of individual representation.

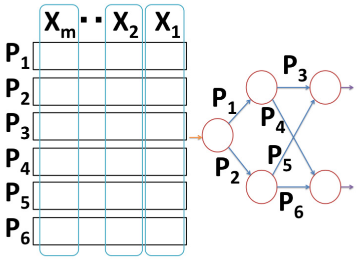


Fig. 2. Example of individual representation in CoSyNE. Each subpopulation $P_1, \cdots, P_6$ on the left corresponds to a weight of individual ANN on the right. Each $\mathbf{x}_1, \cdots, \mathbf{x}_m$ consists of 6 weights, which configure the entire network.

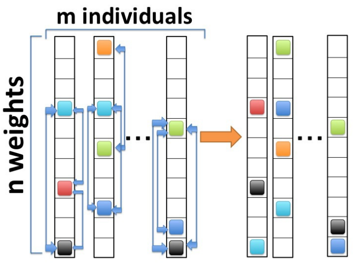


Fig. 3. Example of the probabilistic permutation. Each column on the left corresponds to $\mathbf{x}_i$ in Fig.2. Cells in a column are marked based on Eq.(2). Then, marked cells are swapped randomly as described in the figure. Right side of the diagram shows the result of permutation.

Population in CoSyNE is evolved through mutation, reproduction, crossover, and selection. In the reproduction, the top quarter of the population is selected in terms of fitness to reproduce new individuals, and the new ones replace the bottom quarter of the population. The biggest difference between CoSyNE and other evolutionary algorithms is permutation. The permutation is realized as follows : First, the weights are marked randomly based on inequality

$$rand() \leq prob(x_{ij}), \tag{2}$$

where rand() is a random number drawn from uniform distribution of [0,1]. Previous research suggested that prob(x_{ij}) should be defined as:

$$prob(x_{ij}) = 1 - \sqrt[n]{\frac{f(x_{ij}) - f_i^{min}}{f_i^{max} - f_i^{min}}} \tag{3}$$

or

$$prob(x_{ij}) = 1. \tag{4}$$

Then marked weights are swapped as shown in Fig.3.

However, when n is large in Eq.(3), the second term gets closer to 1 regardless of $f(x_{ij})$. In such case, permutation hardly occurs. In this paper, we use Eq.(3)

where n is equal to 2. The idea behind using Eq.(3) is that the individuals with high fitness have less chance of being permuted, and vice versa [6]. Since our preliminary experiments showed that by using Eq.(3), the results indicate better performance in comparison to by using Eq.(4), this work reports results obtained with Eq.(3).

The initial population is randomly generated from uniform distributions of $[-10, 10]$. In addition, we also tested two normal distribution $\mathcal{N}(0, 1)$ and $\mathcal{N}(0, 5)$ as in CMA-NeuroES to start from the same initial distribution.

We ran CoSyNE with 50 individuals in a population. For more detailed explanations of CoSyNE, you may refer to the previous publication by Gomez et al [6].

3.3 Nelder-Mead Simplex (NMS) [8]

The Nelder-Mead Simplex (NMS) is one of the most prominent algorithms used in function optimization. It is known to perform well on various function optimization problems. For detailed explanations of NMS, you may refer [8].

In this paper, we tested three initial distributions: uniform distribution of $[-10, 10]$, normal distribution $\mathcal{N}(0, 1)$, and $\mathcal{N}(0, 5)$ as used in CoSyNE.

3.4 Random Search (RS)

Random Search (RS) sets each weight as a random number in the beginning of each episode. We evaluated RS with the following three distributions:

1. Uniform distribution of $[-10, 10]$ ($UR[-10, 10]$)
2. Normal distribution of $\mathcal{N}(0, 1)$ ($RN(0, 1)$)
3. Normal distribution of $\mathcal{N}(0, 5)$ ($RN(0, 5)$)

to generate random numbers.

4 Experimental Setup

In this section, we explain the specifications of the simulated quadruped robot and the artificial neural network.

4.1 Quadruped Robot

The quadruped robot we used in this experiment is depicted in Fig.1.

Detailed geometry is illustrated in Fig.4. Main body of the robot is a rectangular box with a 25 cm × 10 cm base and of 6 cm height. The legs are also rectangular boxes with a 2 cm × 3 cm base and of 10 cm height. Each part is built from a material having density of 1 g/cm^3.

Each joint is equipped with a motor and angular spring. The maximum joint torques are set to 1 (kg · m) and the elasticity of springs are set to 6 (kg / radian).

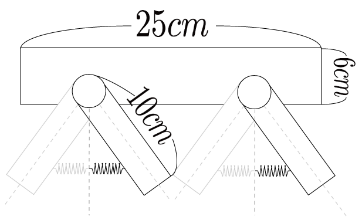


Fig. 4. Projection of the quadruped robot taken from a side. The legs are constrained by springs.

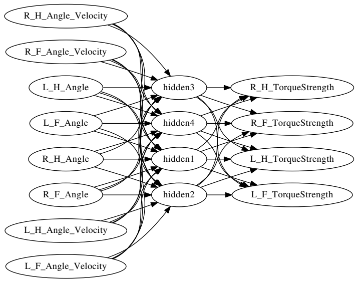


Fig. 5. Predefined structure of the neural network, which is used as the controller of the quadruped robot. The alphabet "L" means "Left" and the "R" means "Right". And the next alphabet "F" means "Front" leg and "H" means "Hind" leg.

4.2 Neural Network

The architecture of the neural network used to control the quadruped robot is shown in Fig.5. Each network is fully connected feed-forward network that has four hidden nodes. In total, the network has 48 connections, whose weights are to be optimized.

An ANN takes the angle and angular velocity of four legs as input signals and outputs the joint torques for four joint motors.

We use the following activation function on hidden and output nodes:

$$\text{hidden} = \frac{1}{1 + \exp^{-x}} \tag{5}$$

$$\text{torque} = \frac{2}{1 + \exp^{-x}} - 1 \tag{6}$$

According to Eq.6, torque output is constrained within [-1:1].

4.3 Evaluation

In the beginning of each episode, the quadruped erects at the origin. The robot is allowed to move for 5 seconds. We defined the fitness of the robot as its travel distance from the origin in straight forward direction. If the robot walks backward, negative fitness can be assigned. Since there is no concept of population and generation in NMS and RS, we evaluated each algorithm in terms of the number of episodes. We ran each algorithm for 50 runs and continued to run each algorithm for 10,000 episodes.

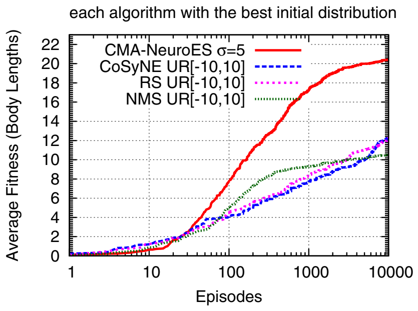


Fig. 6. The learning curves of compared algorithms initialized with the best distribution. The best fitness so far is recorded at each episode. The average of 50 runs over 10,000 episodes is shown. X-axis is log-scaled.

Table 1. Comparison of mean and SD of fitness at 10,000 episodes

	mean	SD
CMA-NeuroES	20.42	2.12
CoSyNE	12.22	3.31
NMS	10.620	3.89
RS	12.26	1.58

5 Results and Discussion

Fig.6 compares all four algorithms with the best performing random distribution or the step-size parameter. The results show their average learning curves over 10,000 episodes. Mean and standard deviation (SD) of walking speed at 10,000 episodes are listed in the Table 1. In order to compare the performance difference in the early stage, Fig.7 illustrates their learning curves until 200 episodes.

From Fig.6 and Fig.7, we can see that CMA-NeuroES consistently outperformed all other algorithms. Performance difference were statistically significant ($p < 0.05$) after 54, 36, 43 episodes over CoSyNE, NMS, and RS, respectively. Although NMS was inferior to CoSyNE and RS at 10,000 episodes, it performed better than them at the earlier stage (from 76 to 5746 episodes over CoSyNE and from 73 to 2949 over RS). In spite of the success of CoSyNE on PB benchmark, it performed poorly in quadruped gait control in terms of achieved walking speed.

In Section 5.1, we discuss the reason that caused the performance difference between CMA-NeuroES and CoSyNE. And in Section 5.2, we also discuss the robustness of CMA-NeuroES against the bad initial distribution.

5.1 The Reason of Performance Difference between CoSyNE and CMA-NeuroES

CoSyNE was not competitive against CMA-NeuroES, although it was on PB benchmark.

In CoSyNE, individuals with high fitness have less chance of being permuted and top quarter of population was used to produce offspring in each generation. These operations were considered to result in efficient optimization on PB benchmark. In terms of this characteristic, we regarded individuals used to produce

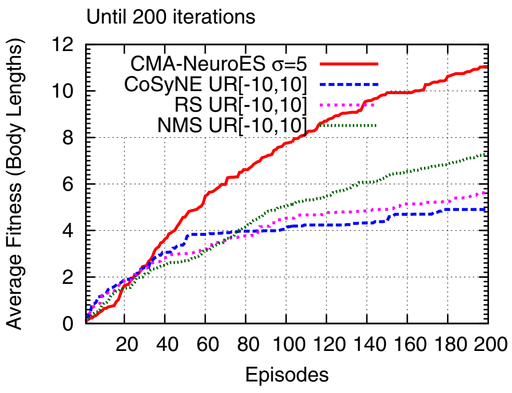


Fig. 7. Comparison of learning curves at the early stage. This figure is zooming from the figure 6 without the log scale on X-axis. In order to emphasize the early stage, the curves are drawn over 200 episodes. Each curve is average of 50 runs.

offsprings as elite individuals. From the viewpoint of these elite individuals, we discussed the reason why the performance of CoSyNE was worse than of the CMA-NeuroES.

In each generation, we calculate the mean fitness of the elite individuals for 50 runs. Elite individuals in CoSyNE correspond to top 25% population, and in CMA-NeuroES top 50% of the population [6,7]. Then, we calculated the mean and the standard deviation (SD) of these 50 mean fitnesses in each generation. The calculated mean and SD of the fitness were plotted at every generation, which corresponds to 50 episodes in CoSyNE and to 15 episodes in CMA-NeuroES, respectively. In addition, we showed the results until 500 episodes only in order to compare them clearly.

In Fig.8, while the mean fitness of CMA-NeuroES increased as the generation, bit CoSyNE has got 0 fitness in all generations. This results shows, that CoSyNE failed to preserve the elite individuals to the next generation. Due to this observation, the poor performance of CoSyNE appears to be caused by its failure in terms of elite preservation.

5.2 Robustness of CMA-NeuroES against the Bad Initial Distribution

In the experiment, CMA-NeuroES outperformed all other compared algorithms.

As for gait optimization on physical robots, NE algorithms should find sufficiently good gaits even with a small number of episodes. Fig.7 shows that it outperforms other algorithms with statically significant difference ($p < 0.05$) at the early stage.

Furthermore, in Fig.9 we compared all four algorithms with the worst performing random distribution and step-size parameter, which is RN(0,1) for all cases. CMA-NeuroES was shown to perform well even with badly tuned step-size

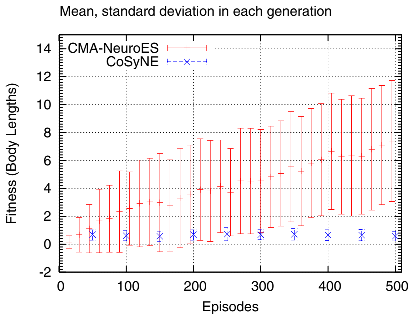


Fig. 8. The mean fitness and standard deviation of averaged elite individuals in each generation for 50 runs. We plot them every 50 episodes in CoSyNE and every 15 episodes in CMA-NeuroES respectively.

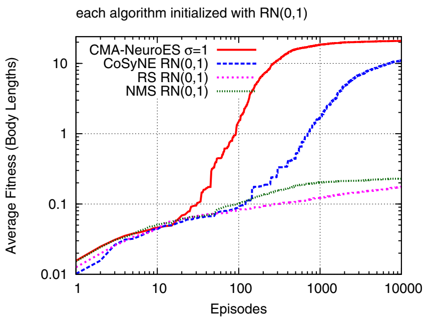


Fig. 9. The learning curves of four algorithms initialized with RN[0,1]

parameters (Fig.9) in comparison with CoSyNE. Since the appropriate parameters are not known in advance, such robustness against ill parameters is desirable characteristic of NE algorithms.

Fig.8 shows that elite individuals are successfully preserved, leading to achieve continuous fitness improvements. We consider that mutation-based evolution strategy could successfully allow elite preservation, while CoSyNE did not.

All these results combined, we consider that CMA-NeuroES is a promising approach to generating gaits of physical quadruped robots efficiently.

6 Conclusion and Future Work

In this research, we investigated two Neuroevolution algorithms, CMA-NeuroES and CoSyNE, on generating gaits of quadruped robots. We compared the walking speed of the quadruped robot gaits generated with each algorithm. In addition, we compared two NE algorithms with the Nelder-Mead Simplex and Random Search.

Experimental results show that CMA-NeuroES outperforms other algorithms. According to these results, we conclude the outstanding performance of CMA-NeuroES scales from PB benchmark to more practical problems, such as generating the gaits of quadruped.

Its performance was still higher than other compared algorithms even with ill-tuned strategy parameters. Moreover, it performed well in the early stage of optimization. These results indicate that CMA-NeuroES a promising method for the gait generation of real physical quadruped robots, in which appropriate initial parameter is unknown and small number of episodes can be repeated.

On the other hand, CoSyNE, which was proven competitive against CMA-NeuroES on Pole Balancing benchmark, did not perform comparably on quadruped gait generation. We discussed that CoSyNE has less scalability than

CMA-NeuroES would be due to failure in preserving elite individuals. According to this, we conclude that CoSyNE would not be suitable for a practical problem, in which a limited number of episodes can be executed. We conclude that CMA-NeuroES performs well in generating locomotion behavior of quadruped robots whereas CoSyNE is not scalable to the practical problem in this research.

In the near future, we will compare these two algorithms with other Neuroevolution algorithms like HyperNEAT. In addition, we plan to conduct the same experiment in a real-world setting to test the scalability of our conclusion.

Accknowledgement. We are deeply grateful to Sajid Hashmi and Yuki Inoue whose support and comments were invaluable to complete this research paper.

References

1. Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J., Lipson, H.: Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In: Proceedings of the 20th European Conference on Artificial Life, Paris, France, vol. 8-12, pp. 890–897. MIT Press (2011)
2. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 2764–2771. IEEE (2009)
3. Valsalam, V., Miikkulainen, R.: Modular neuroevolution for multilegged locomotion. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 265–272. ACM (2008)
4. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE 87(9), 1423–1447 (1999)
5. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evolutionary Intelligence 1(1), 47–62 (2008)
6. Gomez, F., Schmidhuber, J., Miikkulainen, R.: Accelerated neural evolution through cooperatively coevolved synapses. The Journal of Machine Learning Research 9, 937–965 (2008)
7. Heidrich-Meisner, V., Igel, C.: Neuroevolution strategies for episodic reinforcement learning. Journal of Algorithms in Cognition, Informatics and Logic-Algorithms 64(4), 152–168 (2009)
8. Nelder, J., Mead, R.: A simplex method for function minimization. The Computer Journal 7(4), 308 (1965)
9. Hornby, G., Takamura, S., Yamamoto, T., Fujita, M.: Autonomous evolution of dynamic gaits with two quadruped robots. IEEE Transactions on Robotics 21(3), 402–410 (2005)
10. Gauci, J., Stanley, K.: Generating large-scale neural networks through discovering geometric regularities. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 997–1004. ACM (2007)
11. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)

Learning and Generating Folk Melodies Using MPF-Inspired Hierarchical Self-Organising Maps

Edwin Hui-Hean Law[1] and Somnuk Phon-Amnuaisuk[1,2]

[1] Music Informatics Research Group
[2] Faculty of Creative Industries, Universiti Tunku Abdul Rahman, Petaling Jaya Campus, Selangor Darul Ehsan, Malaysia
elhh82@gmail.com, somnuk@utar.edu.my

Abstract. One of the elements in human music creativity results from certain features in the brain that allows it to make predictions of events based on information learnt from past music experiences. Inspired by the *Memory Prediction Framework* (MPF) theory, we propose a method to learn and generate new melodies based on the MPF concept. We first show how an MPF-inspired Hierarchical Self Organizing Map (MPF-HSOM) is used to capture these important features of the brain in the perspective of MPF. This MPF-HSOM is then trained with a selection of melodies taken from a corpus of folk melodies. We then show that by using a prediction algorithm, we are able to generate new melodies based on the trained MPF-HSOM of old melodies. The system proposed here is an abstraction of the features of the brain according to MPF. The results indicate that the system is able to learn and to produce novel melodies of reasonable quality.

Keywords: Hierarchical Self-Organising Map, Learning Folk Melodies, Folk Melody Generation, Memory Predictive Framework.

1 Background

Advances in neuroscience reveal many interesting facts about our brain and its functions. The columnar organization of the cerebral cortex was first characterised by Mountcastle in the 1950s, who, later on in 1978, proposed that the cortical column acted as the unit of computation [11]. Inspired by Mountcastle's work, Hawkins further investigated the concepts of cortical computation units and proposed the *Memory Prediction Framework* (MPF) [4]. The MPF attempts to theorise how the brain functions based on neurological observations. The central idea of the MPF is that the neocortex is a massive memory store. The neocortex processes sensory input which will be used to reinforce previously learned patterns or to form a new learned pattern in neocortical memory. Hawkins suggested four important features of neocortical memory: (i) *patterns are stored in sequences*; (ii) *patterns are recalled auto-associatively*; (iii) *patterns are stored in invariant form*, and (iv) *patterns are stored in a hierarchy.*

Inspired by the four important attributes highlighted by MPF [4], this work investigates the generation of folk melodies using Hierarchical Self-Organising Maps (HSOM). The HSOM is used as it is both an auto-associative memory and a hierarchical memory store. Sequential strorage is implemented through the representation scheme used with the HSOM. A higher layer of HSOM is trained using the patterns of the immediate lower layer, while a higher layer can be seen as abstracting information from the lower layers. The cluster in a higher layer is associated with many clusters in the lower layer and this allows us to emulate features such as sequential memory, associated memory and hierarchical abstraction.

In this research, we do not seek to prove or disprove the memory Prediction Framework, but we will attempt to apply these ideas to emulate the learning and generating of folk melodies using HSOM. The rest of the materials in the paper are organized into the following sections: Section 2: Learning Folk Melodies using MPF-HSOM; Section 3: Experimental Design & Results; and Section 4: Conclusion.

2 Learning Folk Melodies Using MPF-HSOM

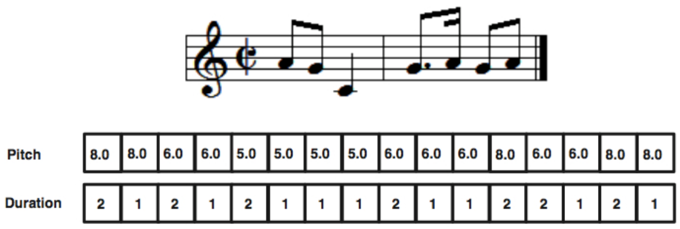
Kohonen's map model performs non-linear mapping. It is natural to apply SOM for clustering task [9]. However, it is not natural to apply SOM to sequential time series data. Time series data is common in problems where temporal information also characterises the data. Researchers have approached this temporal issue from different perspectives. Some examples found in the literature are Sliding window [6]; SardNet [5]; Temporal Organising Map (TOM) [14]; Recurrent SOM [8] and Hierarchical SOM [3]. In music domain, hierarchical structured SOM was used to analyse folk melodies based on symbolic events e.g., note events [10], [13]; to classify melodic sequences extracted from Bach's Fuge[1] [1]. SOM was used to capture a Tune-Map of popular melodies in [12].

There is no best approach in handling information storage and retrieval in SOMs and researchers usually extend their models further to handle different characteristics required in their applications. Because of the nature of work that our HSOM is required to do, the representation scheme used must be bidirectional. This means that not only must the scheme selected be representative of the original input data, but we must also be able to convert the represented data back into its original format. This is a departure from the usual representation schemes used in SOMs where statistical or other analysis is performed on the input data to produce multiple values that capture the characteristics of the data. In those processes, it is usually not required to revert to the original data.

2.1 Knowledge Representation

In our system, a melody is represented as a pair of vectors each representing the pitch **p** and duration **d** of a melody. Each melody is then broken up into elements

[1] This study investigates Fuge in G minor of the Well-Tempered Clavier (vol. I) by J.S. Bach



Pitch	8.0	8.0	6.0	6.0	5.0	5.0	5.0	5.0	6.0	6.0	6.0	8.0	6.0	6.0	8.0	8.0
Duration	2	1	2	1	2	1	1	1	2	1	1	2	2	1	2	1

Fig. 1. The melody representation used in the melody generation experiment. Only two bars is shown here.

that represent a fixed duration in time. In the melody prediction experiment, 8 bars of 2/4 time melodies are used for each vector and each element represents a semi-quaver beat. Therefore, each 8-bar melody is represented as 64 elements of pitch and duration respectively. The pitch values p_{il} (from song i with length l) are obtained from the position of the pitch based on its distance to the tonal center of the scale (e.g., C Major, C minor, etc.) These values are calculated based on Chew's spiral array indices [2] which was a 1-D approximation provided by Chew in her 2-D Spiral Array model. The tonal center of the Spiral Array Indices was denoted by a value of 0 while pitches that were increasingly further away from it would go further away from 0 (both positive and negative). In our work, we have chosen to shift the tonal center to 5.0 so as to eliminate the negative values, and the pitch values will range from 0 to 11. In the case of rests, the pitch values prior to the rest are repeated for the duration of the rest. The duration values $d \in \{0, 1, 2\}$ are one of the three values which represent the note onset (d = 2), note being held (d = 1) and a rest (d = 0). Hence, a melody line $\mathbf{M}_i$ can be represented as:

$$\mathbf{M}_i = [\mathbf{p}_i; \mathbf{d}_i] \tag{1}$$

where $\mathbf{p}_i = [p_{i1}, p_{i2}, ..., p_{il}]$ and $\mathbf{d}_i = [d_{i1}, d_{i2}, ..., d_{il}]$ respectively. Figure 1 shows an example of how the melody line is encoded in vectors of pitch and duration as described above.

2.2 MPF Inspired Hierarchical Self-Organising Maps (MPF-HSOM)

The Self-Organising Map (SOM) [7] is an unsupervised neural network commonly implemented as a two-dimensional array of neurons i.e., a 2D map. Each node i in the map is connected to an input vector $\mathbf{x} \in R^n$ through a weight vector $\mathbf{m}_i \in R^n$. Each node learns to associate its weight vector with input training vectors. The learning is competitive and it is a *winner takes all* paradigm. The learning process can be summarised in two steps. Firstly, a winning neuron c is the *best matching unit (BMU)* $\mathbf{m}_c$, which is defined as:

$$\|\mathbf{x}(t) - \mathbf{m}_c(t)\| \leq \|\mathbf{x}(t) - \mathbf{m}_i(t)\| \text{ for all } i. \tag{2}$$

After the best marching unit is chosen, it and its neighborhood nodes are updated using the following update rule:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \eta(t)h_{c,i}(t)(\mathbf{x}(t) - \mathbf{m}_i(t)), \tag{3}$$

where η is the learning rate and $h_{c,i}(t)$ is the neighborhood function for smoothing neighboring neurons of the winner.

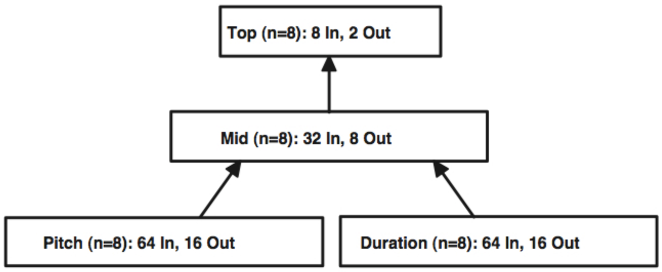


Fig. 2. Three-level HSOM: 8×8 inputs are captured at each of the bottom SOMs (the input layer). This produces 2×2×8 outputs, which are then captured by the middle SOM (the join layer). The top SOM (the melody layer) captures the 2×4 output of the middle SOM, and produces a single 2-element output.

In this implementation, the MPF-HSOM has three levels (see Figure 2), the *Input (bar) layer*[2], the *Join (2-bar) layer* and the *Melody (8-bar) layer*. The Input layer consists of two SOMs, the *pitch input SOM* and the *duration input SOM*, each responsible for capturing the pitch and duration input vectors respectively. The nodes of the input SOMs have a vector length of eight, so the original eight bar melody input vectors which are 64 elements long have to be broken up into eight equal length fragments of eight elements (representing one bar) each. When one melody is trained, eight BMUs will be activated on each of the input layer SOMs, thus producing 32 elements (i.e., 16 (x, y) coordinates of the BMUs chosen, 8 on the pitch input SOM and 8 on the duration input SOM) which will be used as input for the Join Layer.

The Join Layer is thus named becaused it joins the output of the pitch and duration SOMs into one SOM. Each node of the Join SOM also has a vector length of eight and each node accepts as input, the output of eight elements of the output of the input layers below. Similar to how the input for the Input layer are broken up into eight fragments, the input to the Join layer is also broken up into four equal-length fragments. These four inputs to the Join layer will activate four BMUs of the Join Layer and produce eight output corresponding to the eight (x, y) coordinates of the eight BMUs.

In the Melody layer, which sits at the top of this MPF-HSOM, each node has eight vector elements and one input, taken from the output of the Join Layer,

[2] The *bar* is the musical structure, a musical sentence commonly expresses in 2-4 bars.

will activate one BMU in the melody layer. No further manipulation of the Join Layer output is required as it's length is exactly the same as the vector length of the Melody Layer SOM. With this, one input melody which has been originally broken up into eight fragments will activate one node as the information is propagated up to the top of this three level MPF-HSOM hierarchy.

Finally, one more important thing to note is that the temporal relationship between the notes of our melody sequence is preserved even after propagating to the MPF-HSOM as the left to right ordering of the input is preserved from the Input Layer all the way up to the Melody Layer.

Training MPF-HSOM. The bottom level of each hierarchical SOM will be used to capture the input vectors while the SOMs at the higher levels of the hierarchy will store the output or connections to the bottom levels. In terms of training, the process is relatively similar to the training of a normal SOM.

The MPF-HSOM is trained one level at a time, beginning from the ones at the bottom of the hierarchy. As each SOM completes its training, its output are propagated to the SOM at the next level. The bottom level SOM is trained as a normal SOM while the higher level SOMs will be trained with the output of the SOM directly below it. An output, which consists of the X and Y coordinates of the BMU is produced for each vector that is trained onto the bottom SOM.

2.3 Predicting Missing Data Elements Using MPF-HSOM

The main feature of the MPF-HSOM design is that it allows us to form predictions of missing data elements in a data vector. This prediction is performed by traversing the different levels of the MPF-HSOM. Each level provides a different level of contextual information, with local information available at the lowest level and global information available at the top. At the start of the prediction, a partial BMU is obtained for an input string with missing elements. The partial BMU formulae is as follows:

$$\|\mathbf{x}'(t) - \mathbf{w}'_{bmu}(t)\| \leq \|\mathbf{x}'(t) - \mathbf{w}'_i(t)\| \tag{4}$$

$\mathbf{x}'$ and $\mathbf{w}'$ denote the input and weight vectors with missing elements. When making the BMU calculation, the partial BMU ignores the elements at the missing positions, and obtains a BMU which is the best estimate based on the available information. This partial BMU output is then propagated upwards on the hierarchy to obtain another prediction. If all elements are missing and no output (prediction) can be produced at the level below, then a partial BMU is also obtained at the current level. This upwards propagation is repeated until the top SOM produces its prediction. The process is then unwound to find the nodes at the bottom SOMs which are associated with the BMU of the top SOM. The values contained in these nodes are the predictions obtained for the incomplete vectors that have been used.

3 Experimental Design and Results

In this section, we discuss the experiment that was performed to generate new melodies by using the predictive abilities of the MPF-HSOM. Firstly, a pool of melodies were selected and trained onto the MPF-HSOM. These trained melodies served as the memory store in which the predictions would be based on. Once trained, the prediction process was performed by removing a significant portion of the elements from the original input. These input melodies with missing elements were then fed into the MPF-HSOM, and the MPF-HSOM would try to make predictions based on what information was available. However due to the large number of missing elements, the predictions produced would not be the same but dependent on the original context. With this, new melodies were generated.

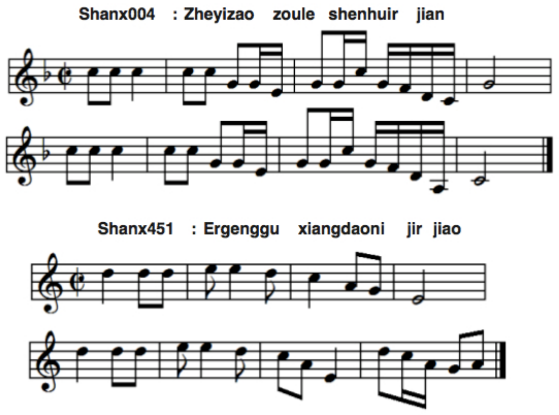

Shanx004 : Zheyizao zoule shenhuir jian
Shanx451 : Ergenggu xiangdaoni jir jiao

Fig. 3. Examples of training melodies from the Shanxi corpus of the Essen Folk Song Collection

3.1 Training data

The training melodies were obtained from the Shanxi corpus of the Essen Folk Song Collection in the Kern Scores Library[3]. The Shanxi folk tunes are characterised by their two four-bar phrase. The first four-bar is the call and the second four-bar answers the call. The melodies from the Shanxi corpus were parsed and filtered to fit within the limitations of the MPF-HSOM system. A total of 178 melodies of 8 bars each were obtained. Each of these melodies were then transposed to a single key (C-Major) and edited to remove irregularities because the representation scheme was not able to represent some triplets and some dotted notes. We modified the triplets into a combination of three notes with one full

[3] Kern Scores Virtual Music Library, `http://kern.ccarh.org`

duration note and two halves duration notes, while the pitches remained as the same three pitches in the triplet. For the dotted notes, the dot was simply removed when the half duration was too small to be represented. Since most of melodies used had a 2/4 time signature, for the sake of uniformity most melodies that contained 2/2 or 4/4 time signatures were converted into the 2/4 time signature by increasing or reducing the note durations respectively. Melodies that had time signatures like 3/4, 3/8 were simply removed from the training data set. Finally, melodies of unsuitable length were also cropped, repeated or broken into a few melodies so that each melody had exactly eight bars.

3.2 Melody Prediction

The operation of the melody generation system was broken up into two phases, the training phase and the prediction phase. In the training phase, melodies which had been converted into the appropriate vectors were trained onto the successive layers of the MPF-HSOM. The most bottom layer(s) accepted the converted melody vectors that corresponded to a single bar length fragment of the melody as input while each successive layer received as input, the coordinates of the Best Matching Units of the layers below it. With this, each SOM node in the bottom layer corresponded to just one bar of the melody, while each SOM node at the top layer would correspond to an entire melody line.

In the prediction phase, one incomplete melody line (with some vector elements removed from it) was fed as an input to the bottom SOM. A best matching unit was found on the top layer SOM for this incomplete fragment. Because of errors introduced by the missing data elements, the BMU units found in each of the layers would not likely fall on the exact BMU that had been selected during training, but rather onto the neighbouring nodes of the BMU which were most likely to resemble the melody before the elements were removed. After the BMU on the top layer was found, its contents (the coordinates of nodes in the SOM below it) would be extracted and this process would be repeated until nodes in the bottom most layer were selected. Finally these bottom SOM nodes would contain the vector elements that would be used to produce one predicted melody.

3.3 Discussion

The entire melody generation system presented here, from representation to training to prediction makes use of all the salient points of the Memory Prediction Framework and is therefore a good example of an experimental demonstration of the ideas presented in the MPF. The melody representation scheme used here ensures that the hierarchical memory system (the MPF-HSOM) stores the captured information *sequentially.* The left to right temporal ordering of the notes in the melody are preserved when conversion into our representation scheme is performed. This is an important characteristic of our chosen representation scheme as it also ensures the reversibility of the representation scheme, wherein melodies that exist in our representation can be used to recreate actual replayable melodies without any loss of information in the reversal process. Both

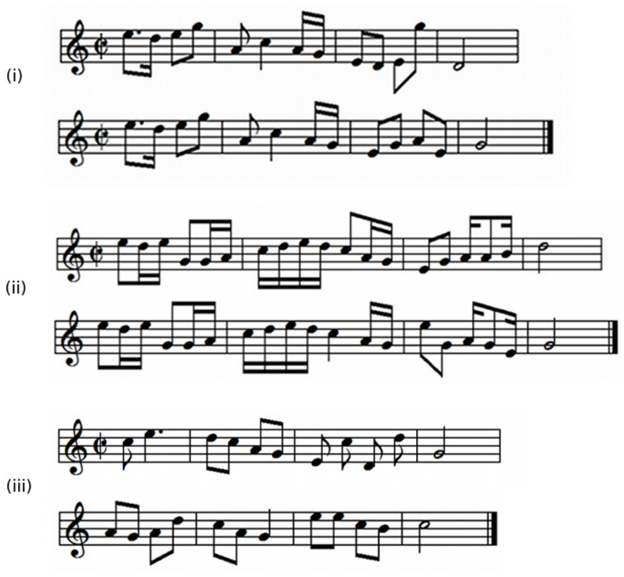


Fig. 4. Examples of Melody Generated using MPF-HSOM: (i) sample output obtained using only duration values; (ii) Sample output obtained using only the first bar values; and (iii) Sample output obtained using 32 randomly selected pitch & duration values.

these properties are a significant departure from many of the works in the music domain in which melodies are processed into a lossy representation which usually consists of various statistical measures of the original music.

For the training process, the MPF-HSOM incorporates two more key points of the MPF which for *hierarchical organization* and *representational invariance.* Both these points are seen in the design of the MPF-HSOM memory system: firstly, where the input data is stored in terms of a three level hierarchy; and secondly, as the data is propagated upwards in the hierarchy, the higher levels contain increasingly higher level of data abstraction which allows for representational invariance.

Finally, the prediction algorithm used in the actual melody generation portion of the experimental system is heavily reliant on the *auto associative* ability of the MPF-HSOM (predicting the elements that have been deliberately removed from an input vector). Unlike single layer SOMs where only information contained in the original SOM is used in making the auto-associative predictions, the MPF-HSOM refines the results of its prediction through the *bidirectional traversal* of the hierarchies. Each prediction is propagated upwards on the MPF-HSOM hierarchy so as to obtain a greater contextual reference for the prediction being made and the prediction is only made after the higher level SOM predictions are propagated downward into the original input level.

4 Conclusion

The MPF-HSOM system detailed in this paper is successful when used to learn and generate melodies. However various improvements can still be made to improve the quality and depth of the generated melodies. If the melodies were to be subjected to a Turing test using a subject with little or no musical training, many of the melodies generated would be able pass such a test. We also would like to highlight the fact that this system is very versatile as it is not tuned to any particular type of music. If used with a different corpus of music, one would be able to achieve comparable results as long as the melodies chosen are adjusted to fit within the constraints of the MPF-HSOM and the representation scheme used.

In conclusion, the design of this sytem has managed to capture all the critical dimensions of the Memory prediction framework. The dimensions of *hierarchical memory* is inherently captured in the design of the system through the use of the Hierarchical SOM while the melody representation scheme used here captures the information and stores it *Sequentially* in the MPF-HSOM. The prediction algorithm used here heavily incorporates the *auto-associative* dimension and also relies on the ability to perform *bi-directional traversal* of the hierarchy to produce results. Finally to a lesser extent, representational invariance is also achieved in the MPF-HSOM hierarchy where information becomes more abstract as it moves up the MPF-HSOM hierarchy. Therefore the system detailed here is an example of how the framework can be used as a reference to implement the MPF to solve other similar problems in different domains.

Acknowledgements. We wish to thank anonymous reviewers for their constructive comments and suggestions.

References

1. Carpinteiro, O.A.S.: A hierarchical self-organising map model for sequence recognition. Pattern Analysis & Applications (3), 279–287 (2000)
2. Chew, E., Chen, Y.: Realtime pitch spelling using the spiral array. Computer Music Journal 29(3), 61–76 (2005)
3. Dittenbach, M., Merkl, D., Rauber, A.: The Growing Hierarchical Self-Organizing Map. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2000), Como, Italy, pp. VI-15–VI-19. IEEE (2000)
4. Hawkins, J., Blakeslee, S.: On Intelligence. Henry Holt, New York (2004)
5. James, D.L., Miikkulainen, R.: SARDNET: a self-organising feature map for sequences. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) Proceedings of the Advances in Neural Information Processing Systems, vol. 7. Morgan Kaufmann (1995)
6. Kangas, J.: On the analysis of pattern sequences by self-organising maps. PhD thesis, Laboratory of Computer and Information Science, Helsinki University of Technology, Rakentajanaukio 2C, SF-02150, Finland (1994)
7. Kohonen, T.: Self-organising Maps, 2nd edn. Springer, Berlin (1997)

8. Koskela, T., Varsta, M., Heikkonen, J., Kaski, K.: Recurrent SOM with local linear models in time series prediction. In: Proceedings of the 6th European Symposium on Artificial Neural Networks, pp. 167–172 (1998)
9. Lampinen, J., Oja, E.: Clustering properties of hierarchical self-organizing maps. Journal of Mathematical Imaging and Vision 2, 261–272 (1992)
10. Law, E.H.H., Phon-Amnuaisuk, S.: Towards Music Fitness Evaluation with the Hierarchical SOM. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., McCormack, J., O'Neill, M., Romero, J., Rothlauf, F., Squillero, G., Uyar, A.Ş., Yang, S. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 443–452. Springer, Heidelberg (2008)
11. Mountcastle, V.B.: Organizing principle for cerebral function: The unit model and the distributed system. In: Eldelman, G.M., Mountcastle, V.B. (eds.) The Mindful Brain. MIT Press (1978)
12. Skovenborg, E., Arnspang, J.: Extraction of Structural Patterns in Popular Melodies. In: Wiil, U.K. (ed.) CMMR 2003. LNCS, vol. 2771, pp. 98–113. Springer, Heidelberg (2004)
13. Toivianen, P., Eerola, T.: A method for comparative analysis of folk music based on musical feature extraction and neural networks. In: Proceedings of the Seventh International Symposium on Systematic and Comparative Musicology; the Third International Conference on Cognitive Musicology, Jyväskylä, Finland, pp. 41–45 (2001)
14. Wiemer, J.: The Time-Organized Map algorithm: Extending the self-organizing map to spatiotemporal signals. Neural Computation 15, 1143–1171 (2003)

Multi Objective Learning Classifier Systems Based Hyperheuristics for Modularised Fleet Mix Problem

Kamran Shafi, Axel Bender*, and Hussein A. Abbass

School of Engineering & Information Technology
University of New South Wales,
Canberra, Australia
{k.shafi,hussein.abbass}@adfa.edu.au,
axel.bender@defence.gov.au
http://seit.unsw.adfa.edu.au/index.php

Abstract. This paper presents an offline multi-objective hyperheuristic for the Modularised Fleet Mix Problem (MFMP) using Learning Classifier Systems (LCS). The LCS based hyperheuristic is built from multi-objective low-level heuristics that are derived from an existing MFMP solver. While the low-level heuristics use multi-objective evolutionary algorithms to search non-dominated solutions, the LCS based hyperheuristic applies the non-dominance concept at the primitive heuristic level. Two LCS, namely the eXtended Classifier System (XCS) and the sUpervised Classifier System (UCS) are augmented by multi-objective reward and accuracy functions, respectively. The results show that UCS performs better than XCS: the hyperheuristic learned by the UCS is able to select low-level heuristics which create MFMP solutions that, in terms of a distance-based convergence metric, are closer to the derived global Pareto curves on a large set of MFMP test scenarios than the solutions created by heuristics that are selected by the XCS hyperheuristic.

Keywords: fleet optimisation, hyperheuristic, learning classifier system, multi-objective optimisation.

1 Introduction

Determining Fleet Size and Mix (FSM) is an important medium to long term decision problem in logistics that involves estimating the optimal composition of a transportation fleet's future demands. In this paper we focus our attention on an FSM variant in the land-based defence logistic domain, referred to as MFMP. In the MFMP, heterogeneous fleets of modularised, military vehicles are constructed such that a range of anticipated future missions can be fulfilled in a cost-effective and efficient manner [2]. Several features make the MFMP stand out from the standard FSM problem including domain-specific constraints

* Axel Bender is with the Defence Science and Technology Organisation, Australia.

relating, for instance, to material handling and access to roads in the logistics network, and a set of unique objectives such as the fleet's ability to sustain a diverse range of operations.

In a previous study [7] Shafi et al. used XCS, a leading learning classifier system, to learn patterns in the MFMP scenarios and relate them to the best performing heuristics using its simple if-then rules base. It used a multi-objective reward function to deal with multiple non-dominated actions available for any particular scenario or its matching conditions. However, the set of low-level heuristics used in [7] was essentially single objective in nature as each heuristic in the set provided a single best solution based on a given criterion. This can become a serious limitation for a problem like MFMP, where the tradeoff between objectives might change over time [1]. In the current work we relax this assumption by introducing a set of low-level heuristics which take multiple objectives into consideration and produce a set of non-dominated solutions instead of a single best solution. An LCS based hyper-heuristic is then used to search this multi-objective heuristic space and choose best performing heuristics for specific MFMP problem instances. We first extend XCS to handle this new requirement and refer to this system as *XCSHH*. We also apply UCS, a supervised classifier system, and develop a UCS based offline multi-objective hyperheuristic (*UCSHH*) using a multi-objective accuracy function.

The rest of this paper is organised as follows: Section 2 sets the context of the paper by providing background information including a formal description of MFMP. The methodology to use two LCS as hyperheuristics for MFMP is detailed in Section 3. The application of the two LCS based hyperheuristics to the MFMP and their effectiveness is demonstrated in Section 4 through experiments and the analysis of their results. The paper is concluded in Section 5.

2 Background

2.1 Modularised Fleet Mix Problem (MFMP)

MFMP involves selecting trucks, trailer and module assets under given constraints to provide the best fleet mix outcomes that fulfil several efficiency and effectiveness criteria [2]. Given:

- T_{di}^{w} mobility tasks, where w is the time window in which a task needs to be fulfilled, d is the duration of the task and i is a unique task ID,
- V^k vehicles, where k represents the vehicle type which includes the combination of modules that the base vehicle can carry,
- M^r modules, where r represents the module type including the set of materials that can go with each module,
- C^k the cost of a vehicle of type k and
- C^r the cost of a module of type r,

the problem is to identify a mixture of vehicles and modules to fulfil the tasks such that the total fleet acquisition cost is minimised,

$$\min\left(\sum_k C^k V^k + \sum_r C^r M^r\right),$$

and the mixture is balanced:

$$\min \frac{\sum_k (V^k - \overline{V})^2}{|k|}.$$

The problem may involve optimisation of more objectives if other dimensions, such as vehicle routing, are considered. For instance, the optimisation heuristics used in this study also try to minimise the total time required to complete all tasks in an MFMP scenario ($\min \sum_k T^w_{di}$).

2.2 Existing Fleet Optimisation System

The fleet optimisation system that meets the complex MFMP requirements consists of two main components [2]. The scenario generation component models complex future operational scenarios. It also performs sensitivity analysis by parameterising the scenario space and generating a large number of scenarios to be evaluated by the second component, the heuristic based solver. This solver evolves pseudo-optimal solutions to the problem instances created by the scenario generation module. Since the low-level heuristics that form the basis of our proposed hyperheuristics are derived from the solver, we discuss it in greater detail below.

2.3 Hyperheuristics

Hyperheuristics in search and optimisation [4][6] are recent techniques that automate the development of optimisation heuristics. Instead of searching in a solution space their distinctive characteristic is that they search in a heuristic space.

2.4 Learning Classifier Systems

LCS are genetic based machine learning systems that evolve a population of classifiers. Each classifier consists of a rule, which includes a condition (essentially a conjunction of predicates) and an action or the predicted class, and a set of parameters that keep different statistics about the classifiers. Stewart Wilson's XCS [9] is considered the current state-of-the art LCS. XCS incorporates temporal-difference learning in its framework and thus has obvious application to reinforcement learning problems. UCS [3] is an LCS derived from XCS. UCS, in contrast to XCS, is specifically designed for supervised learning tasks and benefits directly from known labels during training instead of receiving an immediate or delayed reward as in the case of XCS.

3 LCS Based Hyperheuristic for MFMP

The LCS based hyperheuristic applied to MFMP consists of a set of rules each of which can match the characteristics of one or more MFMP instances. Each rule advocates a single low-level heuristic that should be used when it gets activated. The approach can be categorised as offline selective. This is because the set of rules are learnt offline in a training phase and are used to select multi-objective low-level heuristics to be applied to the future MFMP instances. Notice, however, that the hyperheuristic is not static and has the ability to adapt to changes in the problem space due to the online and incremental learning nature of LCS.

The hyperheuristic for MFMP is built around two representative LCS; XCS and UCS. In both systems, the training (explore) and test (exploit) phases are executed as usual except the parameter update and action selection mechanisms are modified in both systems to suit the requirements of multi-objective heuristic selection. Furthermore, to avoid modifications to different system operators, data input is supplied in the closest match to the standard format. Thus, an input here corresponds to a feature vector representation of an MFMP instance and a set of ranks that are obtained by evaluating each low-level heuristic in the MFMP solver and then using a convergence metric to score the heuristics (see Section 4.1 for details on the generation of training and test data).

3.1 XCS Based Hyperheuristic (XCSHH) for MFMP

During exploration XCSHH builds a *matchset* $[M]$ as usual. Upon receiving an input the whole population is scanned and the classifiers whose conditions match the incoming scenario features form the matchset. An action is then chosen uniform-randomly and an *actionset* $[A]$ is formed of all classifiers in $[M]$ that advocate the selected action. A reward is received for the selected action and the parameters of the classifiers participating in $[A]$ are updated accordingly.

The reward function is explicitly designed to consider multiple objectives. For every input scenario (aka MFMP instance), each low-level heuristic produces a set of non-dominated solutions. The heuristics are ranked relative to their solutions' convergence distance from a global non-dominated set obtained over the whole solution pool. A reward R is computed for each classifier in $[M]$ based on the rank of its predicted action (i.e., the solicited low-level heuristic) using

$$R = \begin{cases} P - \frac{Pr}{M-1} & \mid r \in [0, M-1], M \in [2, N] \\ P & \mid M = 1. \end{cases} . \tag{1}$$

Here P represents a constant payoff, r is the rank of the predicted action for a scenario (0 representing the highest rank), M is the total number of fronts realised by all heuristics in that scenario and N is the total number of heuristics.

The rest of the parameters are updated as usual for each classifier in order. A conventional GA is then applied to the population of classifiers in $[A]$. The offspring are added to the population if they are not subsumed by their parents. The maximum population size is fixed and a weighted, fitness proportional deletion operator removes excess classifiers from the population.

The above procedure is repeated for the whole data set over a given number of training cycles. In the test mode, XCSHH is to predict a heuristic that will perform best in a given test scenario using the hyperheuristic built during training. A fitness weighted system prediction is computed for all matching classifiers for every test scenario and the heuristic with best prediction is chosen.

3.2 UCS Based Hyperheuristic (UCSHH) for MFMP

UCSHH operates in a similar manner to XCSHH. However, instead of a random exploration, it explicitly uses the correct label of the input and updates accuracy of all classifiers participating in a matchset as the ratio of correctly classified instances to total matched instances. Accuracy and accuracy based fitness are the main parameters in UCS. In single-objective classification problems, the accuracy update mechanism relies on a binary answer to each input. However, in multi-objective optimisation, there is no right and wrong: in every scenario each low-level heuristic produces solutions which can be mapped to a multi-dimensional objective space. Hence, in the MFMP hyperheuristics construction through UCSHH a modified accuracy function is used to update classifier accuracy during the training phase:

$$A_{n(i+1)} = \frac{1}{i+1}\left(S_{n(i)} + \begin{cases} 1 - \frac{r}{M-1} & \mid {\scriptstyle r\in[0,M-1],M\in[2,N]} \\ 1 & \mid {\scriptstyle M=1.} \end{cases}\right). \tag{2}$$

Here, $S_{n(i)}$ refers to the accumulated score of the n^{th} classifier in $[M]$ after the i^{th} appearance in $[M]$. Unlike in XCSHH, in UCSHH classifier parameters, including accuracy, are updated in the matchset. However a correctset $[C]$ (equivalent to the actionset in XCS) is formed using the correct label of the input. In the MFMP application, $[C]$ contains all classifiers predicting any of the low-level heuristics whose solutions fall into the non-dominated set for the current MFMP instance. Finally, a GA is applied to $[C]$ similar to XCSHH. The UCSHH gets trained iteratively over the whole training dataset and the post-training ruleset acts as the hyperheuristic for predicting low-level heuristics for future MFMP cases.

3.3 Low-Level MFMP Heuristics

The set of multi-objective low-level heuristics used by LCS based hyperheuristics in this work is derived from the MFMP solver of [2]. The heuristics consist of a set of sorting criteria that are specific to each of the solver's four optimisation modules (see Figure 1). To find MFMP solutions during the evolutionary search in [2] the sorting criteria are chosen at random and iteratively from each module. However, in the work presented here the sets of sorting criteria are fixed inputs into the solver and used to evaluate the solutions evolved by the multi-objective GA. In other words, each low-level heuristic can be considered a

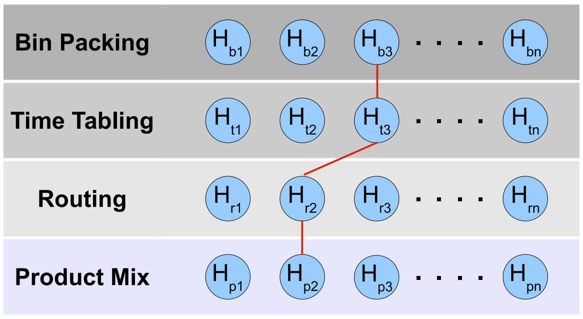


Fig. 1. Graphical representation of low-level heuristics derivation. A low-level heuristic H_x consists of a fixed sequence of heuristics selected from the heuristics pool of each MFMP solver module. In the shown example: $H_x = H_{b3}, H_{t3}, H_{r2}, H_{p2}$.

composite of several domain specific heuristics (here: sorting criteria), as illustrated in Figure 1. The number of low-level heuristics can be very large due to the combinatorics. For instance, for four sub-problems (modules) with ten heuristics each 10^4 low-level heuristics can be constructed, which, considering that the LCS has to match these to n-dimensional MFMP scenario feature vectors, means that the size of the search space for a hyperheuristic is huge. In this paper we therefore only use the Time-Tabling and Routing modules of the MFMP solver to construct the low-level heuristics: six sorting criteria for Time-Tabling and two for Routing (Table 1), leading to a total of 12 low-level heuristics. Notice, that in [7] sorting on performance objectives was included when generating the low-level heuristics, thus turning each low-level heuristic into a single-objective optimisation heuristic. This resulted in each low-level heuristic producing one unique solution for every problem instance only. In the current work, sorting on performance objectives is not included; hence each low-level heuristic used in the MFMP solver produces Pareto sets of MFMP solutions.

Table 1. Sorting criteria used to derive low-level heuristics from the MFMP solver. ↑ and ↓ correspond to respectively ascending and descending sorting.

Time Tabling	Routing
Earliest start time (↑)	Capacity (↑), Speed (↓)
Earliest start time (↓)	Capacity (↓), Speed (↑)
Latest finish time (↓)	
Completion priority (↑)	
Weight to be delivered (↑)	
Weight to be delivered (↓)	

4 Experimental Setup

This section details our experimental methodology used to evaluate the performance of XCSHH and UCSHH for MFMP.

4.1 Data Sourcing

The input data used to train and test the hyperheuristics are generated using the fleet optimisation system described in Section 3.3. The data essentially consist of a set of MFMP scenarios and the sets of non-dominated solutions obtained by each low-level heuristic in each of these scenarios. To suit the input representation of XCSHH and UCSHH, the scenario data are transformed to feature vectors through a feature extraction module extension in the system. The performance data is obtained by applying each of the 12 low-level heuristics in sequence to every input scenario. The low-level heuristics optimise three objective values including the total cost of the fleet, diversity of the fleet mix and the total time taken by the fleet to complete all tasks in a scenario. Since the low-level heuristics use multi-objective evaluation they provide a set of non-dominated solutions to choose from and therefore there is no direct mechanism to rank heuristic performance. Thus we use a normalised convergence metric as in [5]. For each MFMP scenario, the convergence metric C_m for the m^{th} low-level heuristic is computed by measuring the distance between the set of non-dominated solutions ρ_m obtained by this heuristic and an approximated global Pareto Front P. In the traditional multi-objective optimisation studies where the nature of the test functions are often known, the optimal Pareto front is generally used to grade the performance of multiple multi-objective optimisation algorithms [5]. However, in our problem the optimal values of the objective functions are unknown. Therefore a pseudo-optimal Pareto front is generated by first combining all sets of non-dominated solutions obtained by the low-level heuristic suite into a single solution pool. Then a non-dominated sort is performed over the whole solution space to generate the approximated global Pareto front P.

The convergence metric for each heuristic is obtained by averaging the normalised distances over the size of its corresponding ρ:

$$C_m = \frac{\sum_{i=1}^{|\rho_m|} d_i^m}{|\rho_m|} \tag{3}$$

where d_i^m is computed as:

$$d_i^m = \min_{j=1}^{|P|} \sqrt{\sum_{k=1}^{m} \left(\frac{O_k^m(i) - O_k(j)}{O_k^{max} - O_k^{min}} \right)^2}. \tag{4}$$

Here $O_k^m(i)$ and $O_k(j)$ are the function values of the k^{th} objective achieved by the i^{th} solution in ρ_m and j^{th} solution in P respectively; and O_k^{max} and O_k^{min} correspond to the maximum and minimum objective values in P for the k^{th} objective.

Finally, a relative grading is applied to rank all the heuristics according to C. Given n features to characterise an MFMP scenario and l low-level heuristics, the final input into XCSHH and UCSHH then takes the following feature vector form, where r_i is the rank given to the i^{th} heuristic:

$$x_t = f_1, f_2, \cdots, f_n, r_1, r_2 ... r_l.$$

The final training and test datasets are built by sampling 1000 scenarios (MFMP instances), obtaining their solutions by employing the modified MFMP solver, and using 900 instances for training and another non-overlapping 100 instances for the evaluation of the LCS based hyperheuristics.

4.2 Parameter Setting

We used our own implementation of XCS and UCS in C++ with modifications to handle the particular input representation of the MFMP. We used the standard interval based representation to cover the numeric features.The parameters common to XCS and UCS were set to the same value. The maximum population size in XCS was set to twice the population size in UCS due to its complete action map representation. The parameter settings were as follows:

$\alpha = 0.1$, $\beta = 0.2$, $\delta = 0.1$, $\upsilon = 10$, $\chi = 0.8$, $\mu = 0.04$, $m_0 = 0.2$, $r_0 = 0.4$, $\theta_{GA} = 50$, θ_{sub}=20, $\theta_{del} = 20$, XCS population size=4000, UCS population size=2000, $ACC_0 = 0.99$, GASubsumption=YES, ASSubsumption=NO.

4.3 Experiments

In the experiments each system was trained using 100 iterations through the training set. After every training iteration, system performance was recorded in terms of an average score obtained by the system over all test MFMP scenarios. For each test scenario the system receives a score S between 0 and 1 which depends on the rank r of the predicted heuristic and the total number of unique ranks R obtained in a scenario:

$$S = \begin{cases} 1 - \frac{r}{R-1} & \text{for } r \in [0, R-1], R \in [2, N] \\ 1 & \text{for } R = 1. \end{cases} \tag{5}$$

The experiments were repeated for ten different seeds.

Table 2 shows different statistics relating to system performance achieved by XCSHH and UCSHH. The average best scores are obtained by averaging (over 100 test scenarios) the ten best testing cycle scores corresponding to ten independent runs of the experiments. A score of 1 means that, on average, the system was able to choose a low-level heuristic that is closest to the derived global Pareto front for every test scenario in each run of the experiment. The best scores are reported because a system can record a better performance before reaching the maximum number of training epochs. Since XCSHH and UCSHH are offline hyperheuristics, these best models can then be used to select the low-level heuristics in future problem instances. Similarly, the worst scores are averages of the ten worst test runs. The learning iterations correspond to the average number of training cycles after which the best scores were obtained. Finally, the average population size correspond to the number of macros[1] in the final rule sets after the last training cycle, averaged over ten experimental runs. This measure gives an indication of the systems' generalisation capability. Generally, the greater the difference between the macros and the maximum population size, the better the generalisation ability.

Table 2. Performance measures achieved by the two systems over 100 test MFMP instances and averaged over ten independent runs

Measure	XCSHH	UCSHH
Avg. Best Score (%)	94.25±0.008	**99.78±0.003**
Avg. Worst Score (%)	80.6±0.012	77.8±0.078
Learning Iterations	57.9±34.63	58±36.15
Avg. Population Size	3979.4±5.62	**1758±11.47**

Based on a paired *t*-test at more than 99% confidence interval, UCSHH significantly outperforms XCSHH in terms of the best score. It achieves a score of almost 100% with very small deviation, which means that, through training, UCSHH almost always finds a model able to choose the best low-level heuristics. The better performance of UCSHH is further supported by a much compacter rule set than XCSHH. In a sense the poorer performance of XCSHH is expected as its learning is based on a complete action map [3] and hence it subsequently needs to work on a much larger search space. However, this is only one aspect of hyperheuristic performance. In [7] it was shown that XCS can help incorporate a decision maker's bias through modified reward functions. From a hyperheuristic perspective, such a control means that low-level heuristics can be selected that perform at par in terms of convergence distances but contribute to diverse areas of the global Pareto operating curve [1]. Furthermore, XCS can be easily adapted to learn a best action map instead of a complete action map [8], which can improve its ability to select the right heuristics for test problems. UCSHH, on the other hand, is preferable over XCSHH in cases where a decision bias is not available. Finally, it can be seen that both systems learn their best models in about the same time ($\sim$ 58 learning iterations), albeit with a very high variance.

5 Conclusions

The main contribution of this paper is a methodology that can be considered a generic approach to use LCS as a heuristic selection technique in a

[1] XCS and UCS represent multiple copies of a rule in the population as a single rule, called a macro classifier, with a numerosity parameter that keeps a record of the number of copies.

multi-objective optimisation problem where multiple low-level heuristics produce competitive locally non-dominated solutions across a number of problem instances. The methodology was demonstrated using two representative Michigan style LCS, XCS and UCS, that employ modified reward and accuracy functions to guide their learning bias towards selecting Pareto optimal low-level heuristics across multiple problem instances. A military fleet mix problem was used as a test bench to study the performance of the two hyperheuristics. The results show that UCS, due to its more direct representation, produces better results than XCS in terms of selecting low-level heuristics which dominate the other heuristics in the selection pool across a number of test problems.

The use of LCS based hyperheuristics has important implications for multi-objective optimisation problems. The approach can be used for the identification of problem characteristics that lead to specific performance bottlenecks when searching for efficient and effective solutions in important problems. They can also be used to adapt the learning bias according to decision maker preferences towards different objectives.

References

1. Abbass, H., Bender, A.: The Pareto operating curve for risk minimization. Artificial Life and Robotics 14(4), 449–452 (2009)
2. Baker, S., Bender, A., Abbass, H., Sarker, R.: A scenario-based evolutionary scheduling approach for assessing future supply chain fleet capabilities. In: Dahal, K., Tan, K., Cowling, P. (eds.) Evolutionary Scheduling. SCI, vol. 49, pp. 485–511. Springer, Heidelberg (2007)
3. Bernadó-Mansilla, E., Garrell-Guiu, J.: Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. Evolutionary Computation 11(3), 209–238 (2003)
4. Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology. In: Handbook of Metaheuristics, pp. 457–474 (2003)
5. Khare, V.R., Yao, X., Deb, K.: Performance Scaling of Multi-objective Evolutionary Algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 376–390. Springer, Heidelberg (2003)
6. Ross, P.: Hyper-heuristics. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies, pp. 529–556. Springer, US (2005)
7. Shafi, K., Bender, A., Abbass, H.: Fleet estimation for defence logistics using a multi-objective learning classifier system. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 1195–1202. ACM (2011)
8. Shafi, K., Kovacs, T., Abbass, H., Zhu, W.: Intrusion detection with evolutionary learning classifier systems. Natural Computing 8(1), 3–27 (2009)
9. Wilson, S.: Classifier fitness based on accuracy. Evolutionary Computation 3(2), 149–175 (1995)

Where Should We Stop? An Investigation on Early Stopping for GP Learning

Thi Hien Nguyen[1], Xuan Hoai Nguyen[2], Bob McKay[3], and Quang Uy Nguyen[1]

[1] Le Quy Don University, Vietnam
[2] Hanoi University, Vietnam
[3] Seoul National University, Korea

Abstract. We investigate the impact of early stopping on the speed and accuracy of Genetic Programming (GP) learning from noisy data. Early stopping, using a popular stopping criterion, maintains the generalisation capacity of GP while significantly reducing its training time.

1 Introduction

Genetic Programming (GP) [1] describes a class of evolutionary algorithms that solve problems by finding solutions of non-predefined complexity. GP has often been viewed as a form of machine learning, as it aims to induce relations between input and output data in the form of a functional expression or program. Among the successful real-world applications, learning tasks have been common [2]. Early GP research seldom attended to the generalisation capacity of GP. The focus was on how GP could fit the data by finding an exact solution/relation While learning exact solutions may be important in some discovery tasks, machine learning (ML) [3] has emphasised generalisation over unseen data as the most important aspect. A learning machine should avoid over-fitting the training data. Recently, GP generalisation has caught more attention, with an increasing number of related publications [4, 5, 6, 7, 8, 9, 10, 11]. In particular, GP overfitting has been repeatedly demonstrated: while the errors on the training data may improve over the generations, it may deteriorate on unseen test data.

There are at least two ways to combat over-fitting for learning machines [12] – reducing machine complexity (or regularisation) and early stopping. In the first approach, based on Occam's razor [3], the learning process avoids over-fitting by preferring simple hypotheses. In the second, a learner does not eliminate over-fitting but rather tries detect it and stops training once it does so. Early stopping is widely used for learning processes because it is simple, easy to implement, and, in many cases, superior to regularisation [13]. In GP, there have been a number of attempts to improve GP generalisation by regularisation, through including the complexity of an individual as part of its fitness [14, 15, 16, 17, 6]. However, as reported in [18], reducing complexity of individuals may not lead to better generalisation. To date, the only published work on early stopping for GP has been two preliminary works, [19] and [20]. In [19]. Tuite et al. adapted three stopping criteria adopted from [12] to Grammatical Evolution (GE). These criteria helped GE to detect when to stop during training. However the experiments only covered two simple problems, and no detail of the impact of early stopping was provided. In [20], we investigated the impact of early stopping on GP learning. Some results suggested that early stopping could trade off generalisation error and run time complexity, but the results were mixed. However we can now explain them as due to ineffective stopping criteria; better criteria lead to better results.

In this paper, we re-investigate the impact of early stopping on GP learning based on Prechelt's stopping criterion [12]. In Section 2, we briefly review related work on GP generalisation, and on the role of early stopping in learning machines. Section 3 introduces our implementation of early stopping. Sections 4 details the experimental settings. The results are presented and discussed in Section 5. The paper concludes in Section 6 by highlighting possible future work.

2 Background

2.1 Over-fitting and Generalisation in GP

Although achieving high generalisation capability is the main objective any learning machine [3], it was neglected in the early work on GP. Before Kushchu published his seminal paper on generalisation in GP [7], there was little in the literature dealing with GP generalisation. In [21], Francone et al. proposed a system called Compiling GP (CGP) and compared its generalisation with that of other machine learning techniques, demonstrating comparable results. Extending the use of the mutation operator was shown to improve generalisation. Zhang [14] proposed avoiding over-fitting through Minimum Description Length (MDL) methods, providing an adaptive mechanism for balancing between accuracy and complexity preferences. He obtained robust results for tasks with noisy or incomplete data. Hooper et al. [15] argued that expression simplification may help GP to avoid over-fitting and obtain better generalisation. In [22], Iba incorporated Bagging and Boosting into GP (BagGP and BoostGP), showing improved generalisation in discovering trigonometric identities, chaotic time series prediction, and 6 bit multiplexer.

Recently, generalisation in GP has gained more attention. In [9], Panait and Luke investigated the impact of using six common sampling methods on the robustness of GP solutions. None dominated on all problems, showing that the impact of sampling method is problem domain dependent. Paris et al. [23] used GP as the core learning algorithm in a boosting framework to trigger over-fitting on two problems, demonstrating much better performance with boosting. Becker and Seshadri [16] compared two techniques to evolve more comprehensible trading rules. One applied expert-level knowledge of useful arithmetic operators for technical trading, while the other used a complexity penalty in the fitness. The second evolved more comprehensible, better-generalising rules. Mahler et al. [24] tried Tarpeian control on symbolic regression problems and tested for generalisation accuracy, finding mixed results: the effects of Tarpeian control are problem-dependent. In [6], Gagné et al. investigated two methods to improve GP generalisation: the selection of the best-of-run individuals through three separate data sets (training, validation, and test), and the application of parsimony pressure. The validation set results showed somewhat improved stability than parsimony pressure.

More recently, Costelloe and Ryan [4] investigated the role of generalisation in GP. They showed that linear scaling [25] improves GP training performance, but not test performance. They proposed combining Linear Scaling and the No Same Mate strategy [26] for better performance. Vanneschi and Gustafson [11] improved GP generalisation through a crossover based similarity measure. They keep a list of over-fitted individuals, and eliminate individuals that are too similar (based on structural distance or a subtree crossover metric) to individuals in that list. The method was tested on a real-life drug discovery regression problem and showed improvements in GP generalisation. Nguyen Quang Uy et al. [8] showed that semantic information could guide GP crossover to reducing code bloat and improve generalisation on real-valued symbolic regression problems. In [27], Vanneschi at el. proposed a method to quantify/detect over-fitting during GP learning.

2.2 Early Stopping for Learning Machines

The preceding work has focused on avoiding over-fitting to improve GP generalisation, generally through reducing individual complexity. This resembles the common machine learning technique of regularisation. While regularisation often works in GP, recent work has shown that reducing individual complexity does not guarantee better generalisation in GP [18]. Over-fitting is sometimes inevitable. Machine learning also uses another approach: is stopping training when over-fitting is detected [12]. Early stopping has been widely used in neural networks (NN) because of its simplicity and effectiveness. In [12], Prechelt considered three criteria for stopping training. The first criterion stops as soon as the generalisation loss (on an independent validation set) exceeds a predetermined threshold. The second criterion uses the quotient of generalisation loss and progress, while the third stops when generalisation error first increases over s successive training strips. None of the criteria dominated the others in terms of average generalisation performance. However ”slower” criteria, stopping later than others, on average improve generalisation, but at the cost of greater training time [12].

In other word, early stopping embodies a trade-off between training time and generalisation. In [28], Shafi and Abbass investigated the effects of early stopping in learning classifier system (LCS); as with NN, they found that early stopping improves generalisation. The preliminary work of Tuite et al. [19] i and of ourselves [20] in applying early stopping to GP was described in the preceding section.

3 Methods

Our method is inspired by Prechelt's work [12] on early stopping criteria for NNs and that of Tuite et al. [19] on early stopping for GE. We use three data sets: training, testing and validation. The validation set is used to estimate the generalisation error of individuals during evolution. Our stopping criterion is the second from [12].[1]

[1] We have investigated 5 different stopping criteria, but only report the best due to space limitations. Tuite et al. [19] reported the third criterion from [12] as best-performing, but we found the second (with a different parameter setting) better.

To specify the stopping criterion, we first define generalisation loss during the evolutionary process in equation 1:

$$GL(g) = 100.(\frac{E_{va}(g)}{E_{opt}(g)} - 1) \tag{1}$$

where $E_{va}(g)$ is the validation error of the best individual at generation g. $E_{opt}(g)$ is the lowest validation error up to generation g:

$$E_{opt}(g) = \min_{g' \leq g} E_{va}(g') \tag{2}$$

Sharp generalisation loss is an indication of ineffective learning. However, if the training error is still decreasing rapidly, generalisation loss might recover, since we assumed that over-fitting only begins when the error decreases slows [12]. Training progress is defined over a training strip of length k: a sequence of k successive generations $n+1, .., n+k$ with $k|n$. It measures by how much the average training error exceeds the minimum training error within the strip:

$$P_k(t) = 1000.(\frac{\sum_{t'=g-k+1}^{g} E_{tr}(g)}{k \min_{t'=g-k+1}^{g} E_{tr}(g)} - 1) \tag{3}$$

where $E_{tr}(g)$ is the training error of the best individual at generation g. We use the stopping criterion from [12] defined as:
PQ_α: stop after the first end-of-strip generation g satisfying $\frac{GL(g)}{P_k(t)} > \alpha$
(In this criterion, α is a tuning parameter, permitting small excursions in the validation error.)

4 Experiments

We conducted experiments on fifteen regression problems, including both synthetic and real-world data sets. The ten synthetic data sets are given in Table 1. These test functions have been extensively used in the GP and Machine Learning literature.

Table 1. The Synthetic Test Functions

1	$F_1(x) = x^4 + x^3 + x^2 + x$
2	$F_2(x) = \cos(3x)$
3	$F_3(x) = \sqrt{x}$
4	$F_4(x) = x_1x_2 + \sin((x_1-1)(x_2-1))$
5	$F_5(x) = x_1^4 - x_1^3 + \frac{x_2^2}{2} - x_2$
Friedman1	$F_6(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$
Friedman2	$F_7(x) = \sqrt{x_1^2 + (x_2x_3 - \frac{1}{x_2x_4})^2}$
Gabor	$F_8(x) = \frac{\pi}{2}e^{-2(x_1^2+x_2^2)}\cos[2\pi(x_1+x_2)]$
Multi	$F_9(x) = 0.79 + 1.27x_1x_2 + 1.56x_1x_4 + 3.42x_2x_5 + 2.06x_3x_4x_5$
3-D Mexican Hat	$F_10(x) = \frac{sin(\sqrt{x_1^1+x_2^2})}{\sqrt{x_1^2+x_2^2}}$

Table 2. The real-world data sets

Data sets	Abbreviation	Features	Size	Source
Concrete Slump Test	$Slum$	10	103	UCI
Concrete Compressive Strength	$Conc$	9	1030	UCI
Pollen	$Poll$	5	3848	StatLib
Chscase.census6	$Cens$	7	400	StatLib
No2	$No2$	8	500	StatLib

Table 3. Data Sets for Problems. For the synthetic problems, the notation [min, max] defines the range from which the data points are sampled.

Problem	Function	Attribute	Sample size	Training size	Validation size	Test size
1	F_1	$x \in [-1, 1]$	1500	750	375	375
2	F_2	$x \in [0, 2]$	1200	600	300	300
1	F_3	$x \in [0, 4]$	1200	600	300	300
4	F_4	$x_1, x_2 \in [-3, 3]$	1000	500	250	250
5	F_5	$x_1, x_2 \in [-3, 3]$	1000	500	250	250
6	F_6	$x_1, x_2, x_3, x_4, x_5 \in [0, 1]$	1000	500	250	250
7	F_7	$x_1 \in [0, 100]$, $x_2 \in [40\pi, 560\pi]$, $x_3 \in [0, 1]$, $x_4 \in [1, 11]$	1000	500	250	250
8	F_8	$x_1, x_2 \in [0, 1]$	1200	600	300	300
9	F_9	$x_1, x_2, x_3, x_4, x_5 \in [0, 1]$	1000	500	250	250
10	F_{10}	$x_1, x_2 \in [-4\pi, 4\pi]$	1000	500	250	250
11	$Slum$		103	52	25	26
12	$Conc$		1030	515	257	258
13	$Poll$		3848	1924	962	962
14	$Cens$		400	200	100	100
15	$No2$		500	250	125	125

The five real-world data sets were chosen from the UCI machine learning repository [29] and StatLib [30], and are shown in Table 2.

Table 3 shows the ranges from which the inputs for the synthetic problems were drawn, together with (for all problems) the sizes of the data sets into which the training instances were divided.

Since these experiments are about generalisation ability, we corrupted the output of the synthetic test functions by adding Gaussian noise with $\sigma = 0.01$. The parameter settings for the GP systems are given in Table 4. Standard GP is denoted by GPM, while

Table 4. Parameter settings for the GP systems

Population Size	500
Number of generations	150 (for GPM)
Tournament size	3
Crossover probability	0.9
Mutation probability	0.05
Initial Max depth	6
Max depth	15
Non-terminals	+, -, *, / (protected) , sin, cos, exp, log (protected)
Standardized fitness	mean absolute error
Number of runs	50

GPV is a variant which checks the stopping criterion at each generation, terminating if it is satisfied. Otherwise GPM and GPV are identical, up to using the same random seed in corresponding runs.

All runs were conducted on a Compaq Presario CQ3414L computer with Intel Core i3-550 Processor (4M Cache, 3.20 GHz) running Ubuntu Linux operating system.

5 Results and Discussions

For each run, we recorded the generalisation error (GE – measured on the test data) of the best individual of the run, the size of the best individual, the first generation where the best individual of the run was discovered, and the last generation of the run (for GPV). Table 5 presents the results, averaged over 100 runs (with standard deviations). Three different values for α (7, 29, 51) were tested, resulting in three different versions

Table 5. Best Generalisation Errors, Run Times, and p-values and percentage differences, GP with Stopping Criterion (various α) vs GP

	Generalisation Error				p-value of GE			run time				p-value of time		
Functions	PQ_7	PQ_{29}	PQ_{51}	GP	PQ_7	PQ_{29}	PQ_{51}	PQ_7	PQ_{29}	PQ_{51}	GP	PQ_7	PQ_{29}	PQ_{51}
F_1	0.0097±	0.0085±	0.0090±	0.0097±	0.9509	0.2608	0.4975	348.9897±	377.8947±	376.5979±	528.9278±	**0.0000**	**0.0000**	**0.0000**
	0.0080	0.0071	0.0077	0.0081	98.8%	88.4%	92.6%	208.3706	203.1782	201.5762	172.4390	66.6%	73.7%	72.5%
F_2	0.0126±	0.0126±	0.0125±	0.0222±	**0.0005**	**0.0004**	**0.0004**	348.9897±	377.8947±	376.5979±	528.9278±	**0.0000**	**0.0000**	**0.0000**
	0.0138	0.0137	0.0138	0.0223	59.0%	58.4%	58.3%	208.3706	203.1782	201.5762	172.4390	62.7%	66.3%	67.2%
F_3	0.0050±	0.0050±	0.0050±	0.0056±	0.4741	0.3156	0.3113	236.0652±	259.8370±	262.2935±	461.9588±	**0.0000**	**0.0000**	**0.0000**
	0.0044	0.0044	0.0044	0.0046	90.5%	87.1%	90.9%	156.7395	166.9716	166.6070	132.1351	52.1%	56.9%	59.2%
F_4	0.5084±	0.5036±	0.5036±	0.4958±	***0.0368***	0.1222	0.1313	110.4432±	128.4778±	128.5000±	223.3333	**0.0000**	**0.0020**	**0.0020**
	0.0430	0.0485	0.0485	0.0559	100.6%	103.0%	101.9%	162.9480	189.7861	189.5296	221.7522	52.5%	61.7%	60.7%
F_5	1.2100±	1.1854±	1.1688±	1.2342±	0.7429	0.5000	0.3628	400.0000±	421.8750±	429.8105±	563.0632±	**0.0000**	**0.0000**	**0.0000**
	0.5089	0.4907	0.4804	0.5073	99.8%	97.4%	97.1%	202.5774	202.9539	205.2737	166.2303	72.2%	75.6%	76.9%
F_6	1.5875±	1.5708±	1.5381±	1.4677±	***0.0000***	***0.0094***	0.0541	836.1277±	896.7660±	947.1522±	1225.4896±	**0.0000**	**0.0000**	**0.0000**
	0.2981	0.3036	0.2641	0.2317	107.7%	106.6%	103.8%	389.1948	417.8879	417.6109	242.9004	68.6%	73.0%	76.9%
F_7	3.9381±	3.9689±	3.9311±	4.0648±	0.9580	0.7531	0.6594	549.4792±	601.7500±	625.3750±	797.1954±	**0.0000**	**0.0000**	**0.0000**
	2.1808	2.1971	2.1762	1.9215	100%	96.8%	98.0%	322.8258	314.5690	325.4105	237.6541	68.9%	77.4%	81.1%
F_8	0.1456±	0.1414±	0.1412±	0.1312±	0.0656	0.1841	0.1953	529.9000±	564.9300±	568.4600±	747.0100 ±	**0.0000**	**0.0000**	**0.0000**
	0.0603	0.0589	0.0590	0.0483	110.9%	107.7%	107.6%	281.5569	271.7877	270.1209	169.6208	70.9%	75.6%	76.1%
F_9	0.1595±	0.1581±	0.1581±	0.1619±	0.7081	0.5547	0.5547	519.7340±	541.4894±	539.2660±	712.4457±	**0.0000**	**0.0000**	**0.0000**
	0.0470	0.0471	0.0471	0.0399	98.9%	98.4%	99.3%	283.8214	285.9565	285.0949	175.5821	74.7%	76.4%	77.1%
F_{10}	0.0824±	0.0813±	0.0815±	0.0802±	***0.0076***	0.1773	0.1119	188.6186±	261.1667±	271.9896±	565.7113±	**0.0000**	**0.0000**	**0.0000**
	0.0050	0.0057	0.0055	0.0054	102.7%	101.4%	102.4%	208.0677	238.6885	244.9955	212.6491	33.8%	46.6%	49.2%
$Slum$	5.9632±	5.7866±	5.7869±	5.5959±	0.1521	0.4462	0.4489	62.4396±	90.9111±	97.2584±	422.4045±	**0.0000**	**0.0000**	**0.0000**
	1.8028	1.7203	1.7347	1.6211	105.6%	103.1%	102.8%	53.4050	79.0853	80.7470	187.8031	14.7%	21.8%	23.9%
$Conc$	7.8401±	6.9023±	6.9508±	6.8906±	***0.0000***	0.9353	0.6893	1036.8400±	1518.4045±	1541.4286±	1708.7677±	**0.0000**	**0.0443**	0.0735
	1.9861±	0.9700±	1.0593±	1.0114	112.9%	100.4%	101.2%	777.4485	712.0448	708.1432	556.4130	60.8%	85.6%	87.4%
$Poll$	1.7451±	1.7426±	1.7403±	1.7206±	0.5995	0.6366	0.6747	1768.7000±	1995.0500±	2064.0700±	3113.6200±	**0.0000**	**0.0000**	**0.0000**
	0.3320	0.3301	0.3338	0.3275	101.4%	101.3%	101.1%	1105.8256	1199.1555	1190.9865	1090.1410	56.8%	64.1%	66.3%
$Cens$	1.2417±	1.2469±	1.2501±	1.3475±	**0.0000**	**0.0000**	**0.0000**	87.0488±	138.2530±	156.6867±	427.2708±	**0.0000**	**0.0000**	**0.0000**
	0.0509	0.0489	0.0490	0.2042	93.4%	96.0%	97.3%	87.8209	145.4627	167.0125	185.6668	20.8%	33.8%	38.2%
$No2$	0.4846±	0.4822±	0.4813±	0.4740±	0.0663	0.1457	0.1888	164.6364±	235.6566±	254.1100±	567.1600 ±	**0.0000**	**0.0000**	**0.0000**
	0.0421	0.0406	0.0402	0.0387	102.2%	101.7%	101.6%	146.8023	194.6069	201.4310	225.5351	29.0%	41.6%	44.8%

of GPV (PQ_7, PQ_{29}, PQ_{51} in the table)[2]. We tested the significance of the differences in generalisation error between GPM and GPV, using a two-tailed pairwise t-test with confidence level 0.95 ($\alpha = 0.05$). The p-values are shown.[3] Our null and alternative hypotheses were:

- $H_0 =$ "the average GE of GPM and GPV are the same".
- $H_1 =$ "GPM and GPV have different average GE".

In Table 5, if H_0 is rejected the printed p-value is bolded (if GPV is better than GP) or italicised and bolded (if GPV is worse than GPM). When α is small, early stopping can degrade the generalisation capacity of GP. For $\alpha = 7$ (column PQ7), four functions (F_4, F_6, F_{10}, $Conc$) show worse generalisation from GPV solutions than from GPM (H_0 is rejected). This reduces to one function for $\alpha = 29$ (column PQ29) and none for $\alpha = 51$ (i.e. H_0 is accepted in most cases); on the contrary, in two cases for the latter, GPB has strictly better generalisation. Table 5 also shows the percentage ratio between GE of solutions found by each GPV system with that found by GP (averaged over all runs). The average run time (with standard deviations) of each system is given in the right haft of Table 5, with p-values for a two-tailed t-test with null and alternative hypotheses as follows:

- $H_0 =$ "the average run times of GPM and GPV are the same".
- $H_1 =$ "GPM and GPV have different average runtime".

For all settings of $alpha$, GPV has significantly better run time than GPM, on all problems. As α increases, so does the relative runtime of GPV – but generally rather slowly. Overall, the better overall performance of larger α values is probably worth the increased runtime.

6 Conclusions

We have presented a study of the impact of early stopping on GP learning, focusing on its learning efficiency (generalisation error and runtime complexity). The results from 10 synthetic regression and 5 real-world problems show that early stopping improves GP learning efficiency by significantly reducing training time while retaining, or even slightly improving, the quality of the solutions it learns. It also confirms the value of Prechelt's second stopping criterion [12] with different settings of parameter $alpha$ than were used by Prechelt. The results somewhat contradict those reported in [19], where this stopping criterion is found to be less effective for GE. We conjecture that this results from the the different settings of α, and that Tuite et al. might see better results from the second criterion with increased values of α.

In future, we plan to test early stopping criteria on more problems, and to compare with regularisation via Tarpeian Control [24] and similar methods. Since early stopping

[2] [12] and [19] tested α values of 2.5, 5, and 7.5. For our problem domains, these values were too small, and we recalibrated α for the best performance of GPV.

[3] Values shown as 0.0000 were truncated from the actual value so as to fit the table on the page.

uses only the validation and training errors of individuals, and does not interfere with the fitness function or individual complexity metrics as does regularisation, the two (regularisation and early stopping) could potentially be used in combination. Studying such a combination is in our intermediate future plans.

Acknowledgments. The ICT at Seoul National University provided research facilities for this study. This work was funded by The Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2011.08. The first author would like to thank the R@FIT funding of school of Information Technology, Le Quy Don University for providing financial support for her to present this paper.

References

[1] Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
[2] Poli, R., McPhee, W.L.N.: A Field Guide to Genetic Programming (2008), http://lulu.com
[3] Mitchell, T.M.: Machine Learning. McGraw Hill (1997)
[4] Costelloe, D., Ryan, C.: On Improving Generalisation in Genetic Programming. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 61–72. Springer, Heidelberg (2009)
[5] Foreman, N., Evett, M.: Preventing overfitting in GP with canary functions. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO 2005), pp. 1779–1780. ACM (2005)
[6] Gagné, C., Schoenauer, M., Parizeau, M., Tomassini, M.: Genetic Programming, Validation Sets, and Parsimony Pressure. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 109–120. Springer, Heidelberg (2006)
[7] Kushchu, I.: Genetic programming and evolutionary generalization. IEEE Transactions on Evolutionary Computation 6, 431–442 (2002)
[8] Uy, N.Q., Hien, N.T., Hoai, N.X., O'Neill, M.: Improving the Generalisation Ability of Genetic Programming with Semantic Similarity based Crossover. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 184–195. Springer, Heidelberg (2010)
[9] Panait, L., Luke, S.: Methods for Evolving Robust Programs. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1740–1751. Springer, Heidelberg (2003)
[10] Paris, G., Robilliard, D., Fonlupt, C.: Exploring Overfitting in Genetic Programming. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. (eds.) EA 2003. LNCS, vol. 2936, pp. 267–277. Springer, Heidelberg (2004)
[11] Vanneschi, L., Gustafson, S.: Using crossover based similarity measure to improve genetic programming generalization ability. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009), pp. 1139–1146. ACM (2009)
[12] Prechelt, L.: Early Stopping - But When? In: Orr, G.B., Müller, K.-R. (eds.) NIPS-WS 1996. LNCS, vol. 1524, pp. 55–69. Springer, Heidelberg (1998)

[13] Finno, W., Hergert, F., Zimmermann, H.: Improving model selection by nonconvergent methods. Neural Networks 6, 771–783 (1993)

[14] Zhang, B.T., Muhlenbein, H.: Balancing accuracy and parsimony in genetic programming. Evolutionary Computation 3, 17–38 (1995)

[15] Hooper, D., Flann, N.: Improving the accuracy and robustness of genetic programming through expression simplification. In: Proceedings of the First Annual Conference on Genetic Programming 1996, vol. 428. MIT Press (1996)

[16] Becker, L., Seshadri, M.: Comprehensibility and overfitting avoidance in genetic programming for technical trading rules. Technical report, Worcester Polytechnic Institute (2003)

[17] Liu, Y., Khoshgoftaar, T.: Reducing overfitting in genetic programming models for software quality classification. In: Proceedings of the Eighth IEEE Symposium on International High Assurance Systems Engineering, pp. 56–65 (2004)

[18] Silva, S., Vanneschi, L.: Operator equalisation, bloat and overfitting: a study on human oral bioavailability prediction. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009), pp. 1115–1122 (2009)

[19] Tuite, C., Agapitos, A., O'Neill, M., Brabazon, A.: Early stopping criteria to counteract overfitting in genetic programming. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO 2011, pp. 203–204. ACM, New York (2011)

[20] Hien, N.T., Hoai, N.X., Uy, N.Q., McKay, R.: Where should we stop? - an investigation on early stopping for gp learning. Technical Report TRSNUSC:2011:001, Strutural Complexity Laboratory, Seoul National University, Seoul, Korea (February 2011)

[21] Francone, F., Nordin, P., Banzhaf, W.: Benchmarking the generalization capabilities of a compiling genetic programming system using sparse data sets. In: Proceedings of the First Annual Conference on Genetic Programming 1996, pp. 72–80. MIT Press (1996)

[22] Iba, H.: Bagging, boosting, and bloating in genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1053–1060. Morgan Kaufmann (1999)

[23] Paris, G., Robilliard, D., Fonlupt, C.: Exploring Overfitting in Genetic Programming. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. (eds.) EA 2003. LNCS, vol. 2936, pp. 267–277. Springer, Heidelberg (2004)

[24] Mahler, S., Robilliard, D., Fonlupt, C.: Tarpeian Bloat Control and Generalization Accuracy. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 203–214. Springer, Heidelberg (2005)

[25] Keijzer, M.: Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 70–82. Springer, Heidelberg (2003)

[26] Gustafson, S., Burke, E.K., Krasnogor, N.: On improving genetic programming for symbolic regression. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 912–919. IEEE Press, Edinburgh (2005)

[27] Vanneschi, L., Castelli, M., Silva, S.: Measuring bloat, overfitting and functional complexity in genetic programming. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010), pp. 877–884. ACM (2010)

[28] Shafi, K., Abbass, H.A., Zhu, W.: The Role of Early Stopping and Population Size in XCS for Intrusion Detection. In: Wang, T.-D., Li, X., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) SEAL 2006. LNCS, vol. 4247, pp. 50–57. Springer, Heidelberg (2006)

[29] Blake, C., Keogh, E., Merz, C.J.: UCI machine learning repository (1998)

[30] Vlachos, P.: Statlib project repository (2000)

From Subjective to Objective Metrics for Evolutionary Story Narration Using Event Permutations

Kun Wang[1], Vinh Bui[1], Eleni Petraki[2], and Hussein A. Abbass[1]

[1] School of Engineering and IT, UNSW-Canberra, Australia
Kun.Wang@student.adfa.edu.au, vinhqb@gmail.com, H.Abbass@adfa.edu.au
[2] Faculty of Arts and Design, University of Canberra, Australia
Eleni.Petraki@canberra.edu.au

Abstract. The use of evolutionary computation to automatically narrate a story in a natural language, such as English, is a very daunting task. Two main challenges are addressed in this paper. First, how to represent a story in a form that is simple for evolution to work on? Second, how to evaluate stories using proper objective metrics? We address the first challenge by introducing a permutation-based linear representation that relies on capturing the events in a story in a genome, and on transforming any sequence represented by this genome into a valid story using a genotype-phenotype mapping. This mapping uses causal relationships in a story as constraints. The second challenge is addressed by conducting human-based experiments to collect subjective measurements of two categories of familiar and unfamiliar stories to the participants. The data collected from this exercise are then correlated with objective metrics that we designed to capture the quality of a story. Results reveal interesting relationships that are discussed in details in the paper.

Keywords: automatic story narration, genotype-phenotype mapping.

1 Introduction

Three major automatic story generation approaches are observed in the literature: case-based reasoning [1–3], simulation-based [4–6] and grammar-based approach [7, 8]. In the first approach, case-based reasoning is used as a prime reasoning tool for retrieving and combining parts of the existing stories that can match a given user query. In the simulation-based approach, a planning algorithm is applied to let every character act according to the changing world state to achieve their preassigned goals in this virtual story world so that stories can be generated by simply recording the events and world states that happened during the simulation. In the grammar-based approach, story generation becomes a process of grammar derivation and evolution, in which a story grammar is designed by representing the underlying story structure in formal grammar.

If a story can be represented as a genome, existing evolutionary algorithms can be applied to evolve the generated stories. However, to make the story evolutionary process work, an appropriate fitness function must be defined first. This raises one of the biggest challenges in automatic story generation—a computational story evaluation where quantitative story measures such as coherence, novelty and interestingness are required.

Existing evolutionary-based story generation [7, 8] relies on a human-in-the-loop story evaluation. However, interactive evolution makes it difficult to evaluate a large number of stories. As a result, the ultimate solution to an automatic story evaluation and evolution is still to define quantitative measures of a story.

The contributions of this paper are threefold: Firstly, we propose to narrate stories with flashback from a story dependence network and represent the story as a permutation problem that is capable of incorporating heterogeneous story information into one genome. A flashback is a story played backward. It can be a full flashback, where the whole story is played backward, or a partial flashback, where a subset of events are played backward, with the rest of the events played forward. Secondly, a genotype-phenotype mapping is proposed to ensure that any genome can map to a valid story. Third, we present objective metrics for evaluating the narration of a story then correlate these metrics with subjective ones based on human evaluation. This will assist in the future in designing quantitative story measures for automatic story evaluation and evolution.

The rest of the paper is organized as follows. In Section 2 a dependence network that reflects the causal relationships between events in a story is extracted. Subsequently, in Section 3, a permutation-based linear representation is introduced to capture the events in a story in a genome. The data collected from humans are correlated with objective metrics that we designed in Section 4 to capture the quality of a story. The details of the experiment design and result analysis are presented in Section 5. Finally, some concluding remarks are made and future work is discussed in Section 6.

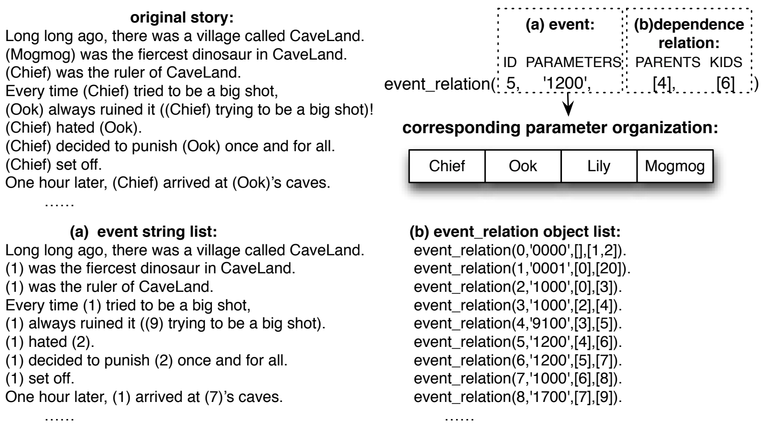
2 Story Dependence Network

In Oxford dictionary [10], event is defined as a "thing that happens, especially something important". This definition has been further enhanced by TimeML guideline—an international cross-language ISO standard for annotating events from text [11]. According to the TimeML guideline, an event is defined as "a cover term for situations that happen, occur, hold, or take place" which "can be punctual or last for a period of time" and also includes "those predicates describing states or circumstances in which something obtains or holds true".

However, the subtlety of the original TimeML event definition makes it difficult to give a computational and unambiguous event definition for our story parsing use because some of the difference between an event and non-event are so delicate that even humans fail to reach a consensus [12]. As a result, the event definition applied in this paper is as follows: Event is a predicate that denotes an action, state, or occurrence in a story, which is bound by a position in the temporal dimension, possesses a spatial situation in the story world and has participants as parameters.

Compared to the TimeML event definition, we regard all predicates as events and exclude those nouns that do not belong to any predicate such as “rain”. Instead, those nouns will be extracted as parameters of the events they are involved in. We also group similar events, e.g. events that possess common parameters.

The dependence relation between two story building blocks (say two events) can be identified from a counterfactual test [14] which has this form: “If A had not happened in the circumstances of the story, then B would not have happened”. The event-level dependence network is represented by a list of event strings (see Figure 1 (a) for illustration) and a list of event_relation objects (see Figure 1 (b) for illustration). Every event_relation object contains the information of a specific event including its “ID” and “PARAMETERS”, and the “ID” information of all its dependently related events. Besides, we also group similar events that possess the same non-zero parameters and obtain a new dependence network.



original story:
Long long ago, there was a village called CaveLand.
(Mogmog) was the fiercest dinosaur in CaveLand.
(Chief) was the ruler of CaveLand.
Every time (Chief) tried to be a big shot,
(Ook) always ruined it ((Chief) trying to be a big shot)!
(Chief) hated (Ook).
(Chief) decided to punish (Ook) once and for all.
(Chief) set off.
One hour later, (Chief) arrived at (Ook)’s caves.
......

(a) event string list:
Long long ago, there was a village called CaveLand.
(1) was the fiercest dinosaur in CaveLand.
(1) was the ruler of CaveLand.
Every time (1) tried to be a big shot,
(1) always ruined it ((9) trying to be a big shot).
(1) hated (2).
(1) decided to punish (2) once and for all.
(1) set off.
One hour later, (1) arrived at (7)’s caves.
......

(b) event_relation object list:
```
event_relation(0,'0000',[],[1,2]).
event_relation(1,'0001',[0],[20]).
event_relation(2,'1000',[0],[3]).
event_relation(3,'1000',[2],[4]).
event_relation(4,'9100',[3],[5]).
event_relation(5,'1200',[4],[6]).
event_relation(6,'1200',[5],[7]).
event_relation(7,'1000',[6],[8]).
event_relation(8,'1700',[7],[9]).
......
```

Fig. 1. An example of event-level story dependence network

3 Story Generation as Permutation Problem

We represent stories with flashback as a permutation; that is, finding an appropriate permutation of the events and parameters in the story dependence network so that diversified stories with different narration orders and parameter arrangements from the original story can be generated. We first encode the new story generated from the story dependence network into a genome. This genome is a valid permutation (*ie* a permutation under predefined constraints) of integers that represent heterogeneous story information including the order of events, the parameter arrangement (such as characters) and the layer threshold that serves as the meeting point of forward narration and flashback in the story dependence network. We then generate different permutations in the genotype to change the phenotype—the corresponding text-form story.

3.1 Encoding Narration into Genome

Technically, we can narrate the events in the story dependence network in a random order. However, it will be difficult to control the coherence of the generated stories and too much noise will exist in the generated stories for humans to give a comparatively objective evaluation. In order to control the coherence in the generated stories to some degree while generating different stories with flashback, the following constraints have been imposed—a story is told by combining both a forward narration and flashback of the events in the story dependence network.

The above-mentioned constraints can be realized in the following steps: Firstly, randomly choose a layer in the story dependence network as a threshold layer. Then all the events with a smaller layer value will appear in the generated story in the sequence of forward narration, *ie* the events with a smaller layer will always occur before those with a bigger layer, while the events whose layer value is bigger than the threshold layer will be narrated in a flashback way, *ie* the events with a bigger layer will always occur before those with a smaller layer in the generated story. Finally, the story will end at the events with the threshold layer value which serve as the meeting point in the narration of the story dependence network.

Under these constraints, the diversity of the generated stories can still be maintained. For one thing, different layer thresholds will generate stories that end at diversified points in the event-level dependence network. For another, the combination of the alternation of forward events and flashback events can be big, not to mention the options when parallel paths are encountered during the narration of the events in the dependence network.

A new story with flashback generated from the story dependence network is encoded into a genotype. The genotype incorporates heterogeneous story information including the order of events, the parameter arrangement and the layer threshold that serves as the meeting point of forward narration and flashback.

In this paper, the incorporation of heterogeneous story information in a genotype is achieved by assigning unique value ranges to different types of information: suppose there are M events and N layers in the story dependence network, and the overall number of different parameters in the story is P. Then in the genotype, genes whose values are between 0 and M-1 denotes events, genes with the values between M and M+N-1 denotes layers and the threshold layer is the first layer gene that occurs in the genotype which makes all the following layer genes redundant, and genes with the values between M+N and M+N+P-1 denotes parameters of the story and their occurrence order in the genotype determines the parameter arrangement of the phenotype (*ie* the text-form story).

Then the genotype can be generated in two steps. First, we randomly generate a permutation of M+N+P integers from 0 to M+N+P-1 which may not meet the permutation constraints mentioned above. Second, we transform the permutation into a corresponding permutation that conforms to the constraints elaborated above. During the transformation process, the dependence relation information represented in the story dependence network's event_relation object list (*ie* the PARENTS and KIDS members) serves as the reference for constraints and the obtained valid permutation is the generated genotype.

3.2 Obtaining Text-Form Story from Genotype-phenotype Mapping

Then the text-form story (*ie* the phenotype) can be obtained from a genotype-phenotype mapping by scanning the genome from left to right to extract an event list (*ie* a list of event genes) and a parameter list (*ie* a list of parameter genes). See figure 2 (b), (c) for illustration.

We, then, extract the parameter arrangement from the parameter list. Figure 2 (c) illustrates this process where the extracted parameter list is mapped into a parameter arrangement which may be a rearrangement of the parameter reference (*ie* the parameter arrangement in the original story).

Finally, the text-form story can be obtained by enumerating the text representation of each of the events one after another in the order of their positions in the event list. The text representation of an event is obtained by replacing the bracketed numbers in the "event string" (see Figure 1 for illustration) with corresponding parameters in the parameter organization.

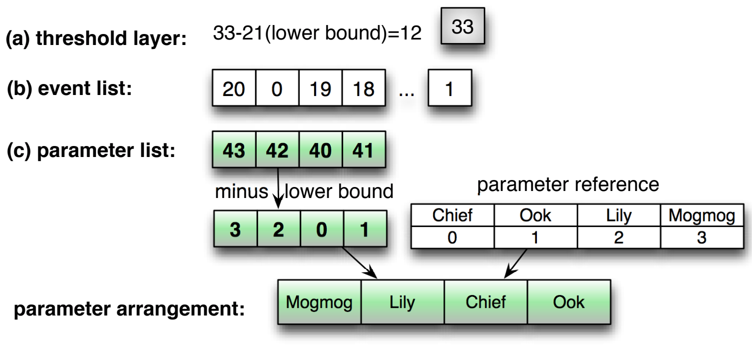
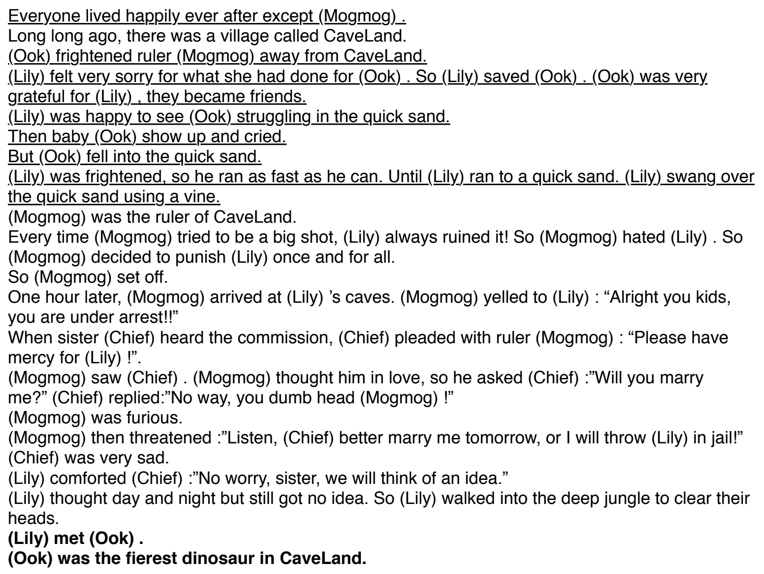


Fig. 2. Extracted story information from the genotype

Figure 3 illustrates the corresponding text-form story (*ie* the phenotype) of the genotype. In this figure, the underlined text corresponds to the flashback part of the story dependence network; the regular text corresponds to the forward narration part while the forward narration and flashback meet at the event whose layer equals to 12 which is denoted as bold face text. It is worth mentioning that although the layer value of the last event "(Ook) was the fiercest dinosaur in CaveLand" is 1, the layer value of its only KIDS node in the story dependence network is 13, so it has an equivalent layer value ranging from 1 to 12, this means that it can occur at the end of this story.



Everyone lived happily ever after except (Mogmog) .
Long long ago, there was a village called CaveLand.
(Ook) frightened ruler (Mogmog) away from CaveLand.
(Lily) felt very sorry for what she had done for (Ook) . So (Lily) saved (Ook) . (Ook) was very grateful for (Lily) , they became friends.
(Lily) was happy to see (Ook) struggling in the quick sand.
Then baby (Ook) show up and cried.
But (Ook) fell into the quick sand.
(Lily) was frightened, so he ran as fast as he can. Until (Lily) ran to a quick sand. (Lily) swang over the quick sand using a vine.
(Mogmog) was the ruler of CaveLand.
Every time (Mogmog) tried to be a big shot, (Lily) always ruined it! So (Mogmog) hated (Lily) . So (Mogmog) decided to punish (Lily) once and for all.
So (Mogmog) set off.
One hour later, (Mogmog) arrived at (Lily) 's caves. (Mogmog) yelled to (Lily) : "Alright you kids, you are under arrest!!"
When sister (Chief) heard the commission, (Chief) pleaded with ruler (Mogmog) : "Please have mercy for (Lily) !".
(Mogmog) saw (Chief) . (Mogmog) thought him in love, so he asked (Chief) :"Will you marry me?" (Chief) replied:"No way, you dumb head (Mogmog) !"
(Mogmog) was furious.
(Mogmog) then threatened :"Listen, (Chief) better marry me tomorrow, or I will throw (Lily) in jail!"
(Chief) was very sad.
(Lily) comforted (Chief) :"No worry, sister, we will think of an idea."
(Lily) thought day and night but still got no idea. So (Lily) walked into the deep jungle to clear their heads.
(Lily) met (Ook) .
(Ook) was the fierest dinosaur in CaveLand.

Fig. 3. The text-form story obtained from the genotype-phenotype mapping

4 Story Measures Selection

This section is concerned with the second challenge of this research: how to evaluate stories using proper objective metrics. This challenge is addressed by conducting human-based experiments to collect subjective story measurements to the participants. The data collected from this exercise are then correlated with objective features that we have designed to capture the quality of a story.

4.1 Subjective Story Measures

In this investigation, coherence, novelty and interestingness and the overall quality of a story have been selected as the subjective measures of human story evaluation. The coherence of a story reflects "a global representation of story meaning and connectedness, which is the temporal and causal structure of a story" [15]. Coherence makes a story understandable to the reader [16].

Novelty reflects the unexpectedness and rule-breaking degree of a story. It serves as a supplement to the coherence measure. An objective of automatic story generation is to help discover stories or structures that exceed our imagination so as to achieve some degree of creativity: a fundamental characteristic of human intelligence and an inescapable challenge for any forms of AI.

Generating interesting stories is another general objective of storytelling. If we may say the above two measures reflect a readers's global impression of a story after understanding is achieved, then interestingness may indicate the dynamics of human's appreciation of a story in the sense that "the increases in cognitive interest were observed before full comprehension was achieved" [17].

In the investigation, all the data of the above story measures are collected as scores given by humans after reading each of the stories. Overall score serves as a indicator of human's overall evaluation of a story's quality.

4.2 Objective Story Features

In this investigation, the definition of some quantitative story features are facilitated by our proposed story representation. The story features defined in this paper are listed below:

– **disOfFlashback** means the distance of flashback feature of a generated story, which is determined by the threshold layer in the genotype that has been chosen as the meeting point of forward narration and flashback in the story dependence network;
– **consistEventOrders** denotes the consistency of the event order of a generated story with the event order in the reference story, *ie* the original "Little Red Riding Hood" or "CaveLand" story;
– **consistPars** denotes the consistency of the parameter arrangement with the parameter arrangement in the reference story. It is noteworthy that in this section, the term "participant" refers to the participant of the investigation instead of that of an event in the story dependence network;
– **consistParRoles** denotes the consistency of the arrangement of parameters' roles (*ie* characters and objects' roles in the story) with those in the reference story. The parameter roles we have differentiated in this paper include victims, villains, heros, and event triggers.

The reasons for choosing the previous quantitative story features are manifold. First, different degrees of flashback and deviation from the smooth flow of dependency in the story dependence network may have diversified effects on reader's understanding and evaluation of the story. Second, different ways of parameter shuffling may change the reader's mental picture of the story [9] thus manipulating his or her understanding of the story to a certain degree. Third, shuffling parameters while totally destroying their roles may have different effects compared to shuffling parameters while maintaining their roles to some degree.

5 Experimental Study

In this section, we present the details of the experiment carried out to study the correlation between objective story features and different subjective measures of human story evaluation such as coherence, novelty and interestingness. This correlation may help us to design quantitative story measures from quantitative story features in the future so as to facilitate semiautomatic and even automatic story evaluation and evolution, which is one of the biggest challenges in automatic story generation field. Besides, this investigation may also provide insights into human-in-the-loop evolution.

5.1 Experiment Design

Two existing children's stories with different familiarity are selected for the investigation in order to draw common conclusions regardless of the effect of familiarity and also to uncover the effect of familiarity on human evaluation of a story. The two stories include the classic "Little Red Riding Hood" story which is probably familiar to most of the participants and a story referred to as "Cave-Land" revised from a recent comic book "Ook and Gluk" which is supposed to be unfamiliar to most of the participants.

For each of the two selected children's stories, fifteen different versions with different quantitative features are automatically generated. Eight participants are involved in this experiment where every participant is given the same fifteen versions of a story in a continuous time slot of a day which fluctuates between half an hour to half past an hour depending on the evaluation difficulty of that story and individual diversity in story understanding.

In order to reduce the impact of tiredness and context influence on the investigation result, for different participants, the reading order of the same fifteen versions of a story has been shuffled randomly, so the cognitive state of different participants when evaluating the same version of that story may vary.

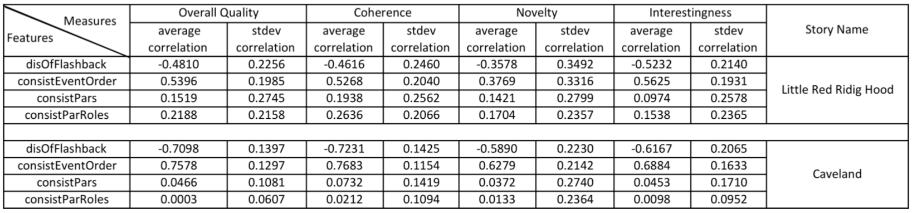
After reading every version of a story, each participant is required to give scores on story measures which include its "overall score", "coherence", "novelty" and "interestingness". Finally, the scores of 240 stories (*ie* 2 stories *15 versions *8 participants) are collected and analyzed.

5.2 Results and Analysis

In this paper, the general correlations between different story features and measures are obtained. For each of the two stories, every participant's individual correlation coefficients between any combinations of story features and story measures are calculated. For instance, for the story "Little Red Riding Hood", participant No.1's correlation coefficient between "disOfFlashback" and "coherence" is calculated using Pearson's product-moment coefficient equation where n is the number of different story versions he or she has read which equals to 15, and all the average is calculated across different story versions. Then the average and standard deviation of all the individual's correlation between any story feature and measure is calculated to reflect the general correlation between any story feature and measure across individuals.

Figure 4 shows the statistics of the correlation coefficients between each story feature and measure across different individuals. The denotation used in this table is explained as follows: "average correlation" denotes the average of the correlation coefficients between a story feature and a story measure across individuals; "stdev correlation" denotes the standard deviation of the correlation coefficients between a story feature and a story measure among different individuals.

A comparison between the two stories in terms of the average correlation coefficient figures offers a number of interesting insights into the effect of familiarity on human evaluation of a story:



Measures / Features	Overall Quality		Coherence		Novelty		Interestingness		Story Name
	average correlation	stdev correlation	average correlation	stdev correlation	average correlation	stdev correlation	average correlation	stdev correlation	
disOfFlashback	-0.4810	0.2256	-0.4616	0.2460	-0.3578	0.3492	-0.5232	0.2140	Little Red Ridig Hood
consistEventOrder	0.5396	0.1985	0.5268	0.2040	0.3769	0.3316	0.5625	0.1931	
consistPars	0.1519	0.2745	0.1938	0.2562	0.1421	0.2799	0.0974	0.2578	
consistParRoles	0.2188	0.2158	0.2636	0.2066	0.1704	0.2357	0.1538	0.2365	
disOfFlashback	-0.7098	0.1397	-0.7231	0.1425	-0.5890	0.2230	-0.6167	0.2065	Caveland
consistEventOrder	0.7578	0.1297	0.7683	0.1154	0.6279	0.2142	0.6884	0.1633	
consistPars	0.0466	0.1081	0.0732	0.1419	0.0372	0.2740	0.0453	0.1710	
consistParRoles	0.0003	0.0607	0.0212	0.1094	0.0133	0.2364	0.0098	0.0952	

Fig. 4. Statistics of the correlation coefficients between each story feature and measure

On the one hand, familiarity with the story "Little Red Riding Hood" made the average correlation coefficients between disOfFlashback or consistEventOrder and all story measures smaller than those of the unfamiliar story "CaveLand". This indicates that flashback in a story plays a comparatively less important role in human evaluation of a story when they are familiar with that story. When reading a familiar story, people can easily situate every event in the whole plot disregarding their occurrence order in the story narration; while reading an unfamiliar story, people will put more effort in understanding the whole context of the story. Flashback destroys the flow of causality thus makes the story difficult to understand.

On the other hand, there is a notable gap between the two stories in terms of the average correlation coefficients between each of the story measures and two story features: consistPars (*ie* consistency with the parameters in the original story) and consistParRoles (*ie* consistency with the parameter roles in the original story). For the familiar story "Little Red Riding Hood", a weak average correlation, around 0.2 can be observed, while for the unfamiliar story "Caveland", the corresponding figures are close to zero, which possibly indicates that people prefer less change of parameters to the familiar stories than the unfamiliar ones. In another word, people prefer alignment of characters (and objects) to the character (and object) stereotypes built in their mind while they are more tolerant to the change of characters (and objects) to unfamiliar stories.

From the standard deviation of the correlation coefficients between each story feature and story measure, it can be perceived that the effect of different story features on story measures vary among individuals, which implies that people hold diversified opinions on what makes a good story. This occurs even though, the individual deviation does not change the overall trend of the correlation between story features and measures. Moreover, for all the story features, it can be observed that the biggest standard deviation of the correlation coefficients is found in the novelty measure, which possibly indicates that people have the most distinct understanding and appreciation of the novelty of a story.

6 Conclusion and Future Work

At this stage of our research in automatic evolutionary story generation, different stories can be randomly generated by changing the narration order of a specific story's dependence network. Future work includes the implementation of the story evolutionary process with an approximated human story evaluation model as the fitness function. Moreover, the content of all the generated story narrations is quite similar. Future work also includes generating stories that can further change the content and underlying structure of the dependence network, that is, generating stories by manipulating the plot.

References

1. Turner, S.R.: MINSTREL: A computer model of creativity and storytelling. Ph.d. thesis, University of California at Los Angeles, Los Angeles, CA, USA (1992)
2. Díaz-Agudo, B., Gervás, P., Peinado, F.: A Case Based Reasoning Approach to Story Plot Generation. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 142–156. Springer, Heidelberg (2004)
3. Pérez, R., Sharples, M.: Mexica: A computer model of a cognitive account of creative writing. Journal of Experimental & Theoretical Artificial Intelligence 13(2), 119–139 (2001)
4. Hsueh-Min, C., Von-Wun, S.: Planning-based narrative generation in simulated game universes. IEEE Transactions on Computational Intelligence and AI in Games 1(3), 200–213 (2009)
5. Cheong, Y., Young, R.: A computational model of narrative generation for suspense. AAAI Press (2006)
6. Peinado, F.: Interactive digital storytelling: Automatic direction of virtual environments. Upgrade. Monograph: Virtual Environments 7(2), 42–46 (2006)
7. Bui, V., Abbbass, H., Bender, A.: Evolving stories: Grammar evolution for automatic plot generation. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2010)
8. Wang, K., Bui, V.Q., Abbass, H.A.: Evolving Stories: Tree Adjoining Grammar Guided Genetic Programming for Complex Plot Generation. In: Deb, K., Bhattacharya, A., Chakraborti, N., Chakroborty, P., Das, S., Dutta, J., Gupta, S.K., Jain, A., Aggarwal, V., Branke, J., Louis, S.J., Tan, K.C. (eds.) SEAL 2010. LNCS, vol. 6457, pp. 135–145. Springer, Heidelberg (2010)
9. Herman, D.: The Cambridge companion to narrative. Cambridge University Press, Cambridge (2007)
10. `http://oxforddictionaries.com/:` Oxford dictionaries online
11. Saurı, R., Goldberg, L., Verhagen, M., Pustejovsky, J.: Annotating events in english timeml annotation guidelines (2009)
12. Puscasu, G., Mititelu, V.: Annotation of wordnet verbs with timeml event classes. In: Proceedings of the Sixth International Language Resources and Evaluation (LREC 2008), European Language Resources Association (2008)
13. Bal, M.: Narratology: Introduction to the theory of narrative. Univ. of Toronto Pr., Toronto (1997)
14. Trabasso, T., Van den Broek, P., Suh, S.: Logical necessity and transitivity of causal relations in stories. Discourse Processes 12(1), 1–25 (1989)
15. Karmiloff-Smith, A.: Language and cognitive processes from a developmental perspective. Language and Cognitive Processes 1(1), 61–85 (1985)
16. Young, R.M.: Computational creativity in narrative generation: Utility and novelty based on models of story comprehension. In: 2008 AAAI Spring Symposium, Stanford, CA, vol. SS-08-03, pp. 149–155 (2008)
17. Campion, N., Martins, D., Wilhelm, A.: Contradictions and predictions: Two sources of uncertainty that raise the cognitive interest of readers. Discourse Processes 46(4), 341–368 (2009)

GPU Accelerated Genetic Clustering

Pavel Krömer[1,2], Jan Platoš[1,2], and Václav Snášel[1,2]

[1] Department of Computer Science, VŠB-Technical University of Ostrava, 17.listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic
[2] IT4Innovations, 17.listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic
`{pavel.kromer,jan.platos,vaclav.snasel}@vsb.cz`

Abstract. Genetic and evolutionary algorithms have been used to find clusters in data with success. Unfortunately, evolutionary clustering suffers from the high computational costs when it comes to fitness function evaluation. The GPU computing is a recent programming and development paradigm introducing high performance parallel computing to general audience. This study presents a design, implementation, and evaluation of a genetic algorithm for density based clustering for the nVidia CUDA platform.

Keywords: genetic algorithms, clustering, GPU, CUDA, acceleration.

1 Introduction

The search for optimal partitioning of a set of objects is known to be an NP-hard problem [1]. Most frequently used clustering algorithms such as the k-means clustering or fuzzy c-means clustering minimize selected cost function in order to find clusters in the data. Their disadvantage is that they require prior knowledge about the data structure, most importantly setting the number of clusters beforehand. Evolutionary algorithms have shown good ability to find quality clusters in the past [1,2,3,12] but the high computational complexity of the artificial evolution for clustering caused by repeated cluster formation and clustering quality evaluation made them impractical for partitioning of real world data sets.

This study presents design, implementation, and evaluation of a genetic algorithm for clustering accelerated by the GPU using the nVidia CUDA platform. It aims to exploit the massively parallel architecture of the GPU to speed up cluster formation and cluster validity evaluation that are the most important parts of fitness function evaluation. The study provides the comparison of two variants of the algorithm on CPU and GPU: the first version uses a pre-computed matrix with distances between points while the second version re-evaluates the distances every time.

2 Evolutionary Clustering

Clustering represents a fundamental data analysis task of separation of objects to meaningful clusters. It is a problem with many applications and a variety of algorithms. The most often used clustering algorithms include hierarchical clustering, centroid (medoid) based clustering, and density based clustering [14]. In this study, we design, implement, and evaluate a simple genetic algorithm for density-based clustering.

Density based clustering was chosen due to its ability to discover clusters with arbitrary shapes. Informally, a density based cluster C is a set of points in the problem space that are *density connected*, i.e. for each pair of points in C there is a chain of points with distance between two consecutive points smaller than a constant ϵ. A popular algorithm for density-based clustering is the DBSCAN algorithm [14].

A parallel approach to density based clustering has been previously presented in [5]. The authors have designed a parallel version of the DBSCAN in the environment of a network of workstations. Their approach combined the hierarchical clustering algorithm OPTICS for data partitioning and an extended version of DBSCAN for clustering of partitions of the data set. The first GPU based approach to density based clustering was presented in [4]. The CUDA-DClust and CUDA-DClust* algorithms were tailored to fit the data parallel architecture of the GPUs and resulted in a significant performance increase of the clustering process.

Various evolutionary algorithms (EAs) have been used to find meaningful clusters in the data. The design of an evolutionary algorithm for clustering involves, among others, the definition of candidate solution representation (chromosome encoding) and the selection of fitness function.

2.1 Clustering Representation

Most often, three main types of clustering representation are used [12]. The *binary encoding* represents each clustering by a binary matrix with the dimensions $m \times n$ where m represents maximum number of clusters and n the number of objects in the data set. The ith row encodes the content of ith cluster: 0 at jth position means that the jth object does not belong to the ith cluster and 1 means that the objects belongs to the cluster. Binary encoding requires very large chromosomes and it is not used very often.

There are more variants of the *integer encoding* for clustering EAs [12]. The label-based integer encoding assigns a cluster label (i.e. an integer cluster id) to every object in the data set. The size of the chromosome equals to the size of the data set which makes it rather impractical for even small data sets (e.g. with several hundreds of objects). On the other hand, the label-based encoding eliminates the need to form clusters during fitness evaluation. Another type of the integer encoding uses the notion of medoids. The representation consists of a sequence of medoids, i.e. objects that represent individual clusters in encoded partitioning [12]. the representation length is equivalent to the number (or maximum number) of clusters in the data set.

The last widely used clustering representation is the *real encoding* that uses real numbers to encode arbitrary points in the problem domain. The points provide a representation of clusters in the encoded partitioning. The chromosomes can also encode additional parameters and control variables as shown e.g. in [6].

2.2 Clustering Evaluation

Proper cluster validity measures are important to evaluate and compare different partitionings of data. Clustering validity measures can be divided into two main categories [7,10,18]. The *external* clustering validity measures use external information about the correctness of the clustering while the *internal* clustering validity measures utilize exclusively the information contained in the data set. Internal clustering validity measures aim to identify good data partitioning by maximizing cohesion (objects in the cluster should be similar) and separation (clusters should be well separated) [7]. Internal cluster validity measures are the basis for unsupervised learning of partitions of data sets. In this study, we use the well known Dunn index [9] defined by:

$$D = \min_{1 \leq i \leq n} \left\{ \min_{1 \leq j \leq n, i \neq j} \left\{ \frac{d(i,j)}{\max_{i \leq k \leq n} d'(k)} \right\} \right\} \tag{1}$$

where $d(i,j)$ is the distance between clusters i and j and $d'(k)$ is the diameter of cluster k. Different Dunn-like indexes use various definitions of $d(i,j)$ and $d'(k)$. We have used the traditional $d(i,j)$ representing the shortest distance between any two objects in i and j and $d'(k)$ expressing longest distance between any two points in cluster k:

$$d(i,j) = \min_{a \in i, b \in j} \{dist(a,b)\}, \qquad d'(k) = \max_{a,b \in k} \{dist(a,b)\} \tag{2}$$

$$dist(a,b) = \sqrt{\Sigma_{i=1}^{n} (a_i - b_i)^2} \tag{3}$$

The disadvantages of Dunn-like indexes include the time complexity of its calculation and higher sensitivity to noise. The advantage of Dunn-like indexes is the ability to validate partitions of arbitrary shapes. In this work we accelerate the Dunn index computation on the GPU and present the performance evaluation and results of a complete genetic algorithm for clustering implemented on CUDA.

3 Genetic Algorithm for Clustering on CUDA

Modern graphics hardware has gained an important role in the area of parallel computing. Graphic cards have been used to power gaming and 3D graphics applications, but recently, they have been used to accelerate general computations as well. The complex architecture of the GPUs is suitable for vector and matrix algebra operations, which leads to the wide use of GPUs in the area of scientific computing with applications in information retrieval, data mining, image processing, data compression and so on. Nowadays, the developer does not have to be an expert in graphics hardware because of the availability of various Application Programming Interfaces (APIs) that help to implement parallel applications rapidly. Nevertheless, it is still crucial to follow the elementary rules of data parallel programming to write efficient GPU code. The Compute Unified Device Architecture (CUDA) is an GPGPU API developed by nVidia.

The GPGPU programming has established a new platform for machine learning and evolutionary computation [8]. Majority of the evolutionary algorithms including the genetic algorithms [20], the genetic programming [21,17], and the differential evolution [22,23,16,15] were implemented on the GPU.

The nVidia CUDA-C language is an extension to the C programming language that allows development of GPU routines called kernels. Each kernel defines a sequence of instructions that are executed on the GPU by many threads at the same time following the data parallel SIMD model. The threads can be organized into so called thread groups (or thread blocks) that can benefit from the GPU features including fast shared memory, atomic data manipulation, and synchronization. The CUDA runtime takes care of the scheduling and execution of the thread groups on available hardware. The set of thread groups requested to execute a kernel is called in the CUDA terminology a grid. A kernel program can use several types of memory: fast local and shared memory, large but slow global memory, and fast read-only constant memory and texture memory.

3.1 The Design of GA for Clustering on CUDA

The genetic algorithm for clustering presented in this study uses real encoded chromosomes with variable length as the representation of the set of clusters, a parallel density-based clustering approach, and the Dunn index as cluster validity measure. The choice of encoding was motivated by the intention to process large data sets, i.e. the integer label-based encoding would not be practical due to the size of the chromosomes and the medoid-based encoding would restrict the choice of cluster representatives to objects that already exist in the collection. The density based clustering was chosen because of its ability to form arbitrary shaped clusters and the Dunn index was selected because it can evaluate the validity of such arbitrary shaped (e.g. non-spherical) clusters. Euclidean distance (3) was used to express the distance between objects in the data set.

The fitness function evaluation consists of three main steps: first, each cluster representative encoded in the chromosome (hereafter referred to as *pin*) is mapped to the closest object in the data set. The closest objects serve as seeds for the formation of density-based clusters. Second, the clusters are formed, i.e. all non-labeled objects that are density-reachable from already formed portions of the clusters are iteratively included to the clusters. With all the clusters fully expanded, the remaining points are gathered into a *remainder* cluster. Finally, the quality of the clustering is evaluated using the Dunn index.

The crossover was in the GA for clustering implemented as mutual exchange of parts of parent chromosomes. The mutation operator was implemented as a random change in the value of a gene, or chromosome expansion allowing an increase in the number of clusters represented by the chromosome. During the expansion, a chromosome encoding n clusters grows by adding $m \in [1, n]$ new randomly generated points potentially representing new clusters.

In the GA for clustering, two pins can expand to the same density-based cluster. This causes little problems in the sequential implementation when the clusters are formed after each other. The redundant pins are simply removed from the chromosome.

3.2 CUDA-C Kernels for Geneti Clustering

To accelerate the GA for clustering, we have implemented the key steps of the fitness function evaluation as CUDA-C kernels that can be executed on nVidia GPUs in parallel. The kernels were *cudaPlacePins*, *cudaFormClusters*, *cudaDunnIndex*, and auxiliary kernels for memory setup and cleanup. The kernels are described in detail in the following sections.

cudaPlacePins. The *cudaPlacePins* kernel maps all pins encoded in the chromosome to the closest object in the data set. It calculates the distance between the pin and every object in the data set because the distance matrix D cannot be used as the location of the pins changes during the evolution. The CUDA-C parallel implementation launches a thread block for all pins in the chromosome and uses each thread in the block to compute the distance between the pin and $no\_of\_objects/block\_dimension$ objects in the data set.

cudaFormClusters. The *cudaFormClusters* implements the formation of the density-based clusters and it is the most complex and time consuming kernel in the program. Each clusters is expanded using a stack breadth-first search (BFS) in the data set. The expansion starts from the seed found in the previous step and it iteratively appends to the cluster all objects that are directly density connected to the cluster. The stack breadth-first search itself is inherently sequential algorithm hard to implement in parallel. Despite that, GPU implementations of BFS were recently presented by [11], and [19]. Current BFS implementations on the GPUs, however, do expect a single BFS instance running at the same time. To find clusters corresponding to each of the k pins in the chromosome, we run k BFS instances in parallel. Similar approach was on the GPU recently implemented by Bohm et al. [4]. The DClust algorithm was based on the idea of chains of density-connected points forming the actual clusters. We a use similar simplified method that aims to form the clusters by a data parallel BFS. Each thread block forms a cluster starting in a pin and expands the cluster. Our implementation uses only a simple collision detection and avoids locking and atomic operations to maximize the performance. In the case of a collision, i.e. when a point is density reachable from more of the forming clusters, it is assigned to one of the colliding clusters at random (due to race conditions). Such a situation is a sign of poor clustering with some clusters too close to each other. The clustering will be assigned a low fitness value and will not survive in the artificial evolution.

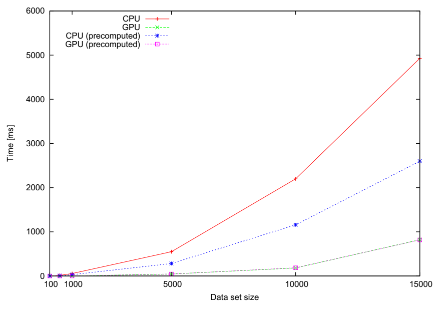
cudaDunnIndex. The last important kernel implements the Dunn index evaluation. It finds the minimum distance between any two clusters and maximum distance between any two points in the same cluster at the same time by a single scan of the distance matrix computing minimum $d(i, j)$ and maximum $d'(k)$ in parallel. The kernel was implemented with minimum branching to optimize the performance.

4 Experimental Evaluation

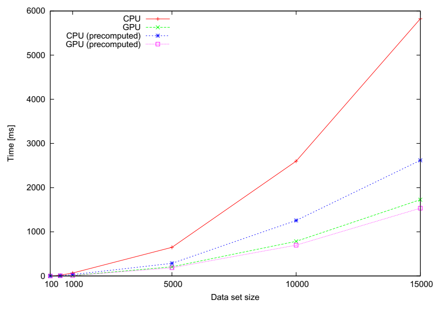
We have evaluated the GA for density-based clustering on the GPU for both, performance and correctness of the results. The experiments were conducted on a server with 2 dual core AMD Opteron processors at 2.6 GHz and an nVidia Tesla C2050 device with 448 cores at 1.15 GHz. To test the performance of the kernels, we have generated a data set containing 100, 500, 1000, 5000, 10000, and 15000 points. The test data set was based on the *data_3_2* data set that was used to evaluate evolutionary clustering algorithms e.g. in [3]. The data set was extended by generating random additional points within the shape of the original clusters. We have measured the time needed to compute the Dunn index (i.e. execution time of the kernel cudaDunnIndex) and to form clusters (i.e. execution time of the kernel cudaFormClusters). The kernel cudaDunnIndex is not data bound and it was executed with the maximum possible block size (1024 threads for compute capability 2.0 on the C2050). The kernel cudaFormClusters was executed with different number of threads per block because the performance of the kernel is data bound. Two variants of fitness function were considered. The first one utilized a precomputed $N \times N$ distance matrix D defined by $D_{ij} = dist(i, j)$ to avoid repeated computation of distances between the same points and the second one computed the distances every time when they were needed. The latter variant represents the situation in which the distances cannot be precomputed e.g. due to memory limits.

The execution time of Dunn index computation and cluster formation on the CPU and GPU is shown in fig. 1(a) and fig. 1(b) respectively. The block size for cluster formation was set upon initial profiling of the kernel. Clearly, the Dunn index evaluation on the GPU is faster than the sequential implementation. The speedup obtained for particular test data sizes is shown in table 1. The improvement in the Dunn index computation time ranges from 1.6 for the smallest data set to 15.12 for the data set of the size 1000 with precomputed distances. When evaluating the distances every time, the speedup ranged from 3.07 to 28.87 because the serial code was significantly slower than in the former case while the parallel code performed similarly. The drop in GPU speedup for the three largest data sets is due to increased number of accesses to the global memory where the collision matrix was stored. The speedup in cluster formation achieved by the GPU is also shown in table 1. The CPU was faster for smallest data sets when using precomputed distances because it can benefit from CPU's superior clock speed, fast cache memories and so on. The GPU implementation was able to speedup the cluster formation process for 1000 objects data set more than 1.76 times, for 5000 objects data set 3.43 times, for 10000 objects data set 5.08 times, and for the largest data set more than 6 times. The speedup in cluster formation was different when the distances were not precomputed. It was approximately twice as large as in the former case for the data sets with 5000 and more objects but it was smaller (e.g. the GPU code was slower) for the two smallest data sets.

The performance evaluation is rather illustrative because the performance of the cluster formation process is data bound, i.e. the speedup factor will be



(a) Dunn index computation time on the CPU and on the GPU



(b) Cluster formation time on the CPU and on the GPU with block size 64

Fig. 1. Execution time of fitness evaluation steps

Table 1. Dunn index computation and cluster formation speedup on the GPU

Dataset	Dunn index		Cluster formation	
size	Precomputed	Regular	Precomputed	Regular
100	1.59505	3.06729	0.254054	0.197507
500	11.7554	22.9229	0.365051	0.778277
1000	15.1233	28.8716	1.75798	1.59244
5000	6.267	12.223	3.42677	7.35243
10000	6.25585	12.0584	5.08202	10.0821
15000	3.18195	6.00526	6.08223	12.0933

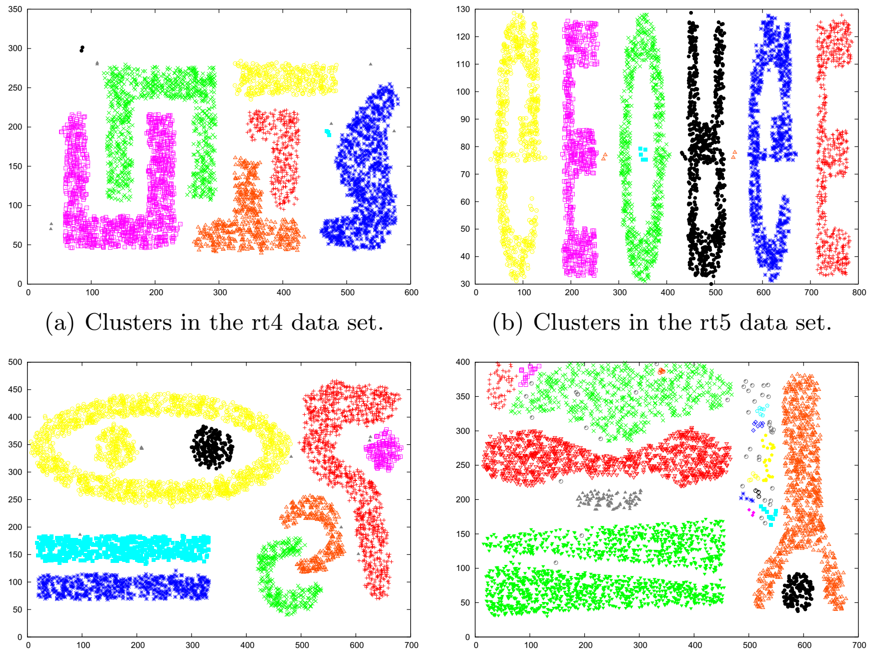
different for different data sets. The performance of the *cudaFormClusters* kernel was also affected by the intensive work with global memory because the data structures needed to perform the stack BFS did not fit to the shared memory.

Next, we have tested the ability of the CPU based and the GPU based implementation of the GA for clustering to find good partitioning of different data sets with irregular partitions and different density. We have downloaded the modified Chameleon data set collection [1] introduced in [13]. Most of the noise was removed from each data set within the collection to test the ability of the GA for clustering to find correct clusters. After the modifications, four data sets were created. The data set rt4 with 4231 objects, rt5 with 4407 objects, rt7 with 5305, and rt8 with 4877 objects. The GA for clustering was used to find clusters in the data sets. The GA used a population of 100 candidate solutions, neighborhood size ϵ equal to 10, crossover probability 0.8, and mutation probability 0.6 (the parameters were set after an initial testing of the algorithm.). The artificial evolution was executed for 200, 500, and 1000 generations with thread block sizes 90 and 256.

The GA was in most cases able to identify correct partitioning of the data before reaching 500 generations. Both, the CPU and GPU implementations

[1] `http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/chameleon-data.tar.gz`

delivered correct results. The clusters found by the GA accelerated by the GPU are shown in fig. 2(a), fig. 2(b), fig. 2(c), and fig. 2(d) respectively. The largest clusters were identified and the remaining outlying points were gathered in the remainder cluster. Let us note that the left circle inside the ellipse in the data set rt7 is density connected to the ellipse. Also, the two triangles in the data set rt8 are density connected and the upper left cluster and the sparse vertical clusters in the rt8 are composed of more separated clusters with $\epsilon = 10$.



(a) Clusters in the rt4 data set.

(b) Clusters in the rt5 data set.

(c) Clusters in the rt7 data set.

(d) Clusters in the rt8 data set.

Fig. 2. Clusters discovered in Chameleon data sets

5 Conclusions

This work presents a design, initial implementation, and evaluation of a genetic algorithm for clustering accelerated by the GPU. A simple density-based clustering and the Dunn index were used as the GPU powered building blocks of a genetic algorithm for clustering that were shown to outperform for some data sets its CPU-based counterparts more than 28 times. This is an encouraging result and further optimizations can lead to additional performance increase.

This work will continue in multiple directions. The cudaComputeDunn kernel can be improved so that it will use shared memory for the collision matrix whenever it fits into it and the texture memory will be considered in other cases (the trade off between additional memory transfers and performance gain will be investigated). Also, a more comprehensive implementation of the clustering algorithm able to handle noise will be implemented on the GPU.

Acknowledgment. This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic. It was also partially supported by SGS, VŠB-Technical University of Ostrava, under the grant no. SP2012/58, and by the Ministry of Industry and Trade of the Czech Republic, under the grant no. FR-TI1/420.

References

1. Alves, V., Campello, R., Hruschka, E.: Towards a fast evolutionary algorithm for clustering. In: Yen, G.G., Lucas, S.M., Fogel, G., Kendall, G., Salomon, R., Zhang, B.T., Coello, C.A.C., Runarsson, T.P. (eds.) Proc. of the 2006 IEEE Congress on Evolutionary Computation, July 16-21, pp. 1776–1783. IEEE Press, Vancouver (2006)
2. Bandyopadhyay, S.: Genetic algorithms for clustering and fuzzy clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1(6), 524–531 (2011)
3. Bandyopadhyay, S., Maulik, U.: Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Rec. 35(6), 1197–1208 (2002)
4. Böhm, C., Noll, R., Plant, C., Wackersreuther, B.: Density-based clustering using graphics processors. In: Proc. of the 18th ACM Conf. on Information and Knowledge Management, CIKM 2009, pp. 661–670. ACM, New York (2009)
5. Brecheisen, S., Kriegel, H.-P., Pfeifle, M.: Parallel Density-Based Clustering of Complex Objects. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 179–188. Springer, Heidelberg (2006)
6. Das, S., Abraham, A., Konar, A.: Automatic Hard Clustering Using Improved Differential Evolution Algorithm. In: Das, S., Abraham, A., Konar, A. (eds.) Metaheuristic Clustering. SCI, vol. 178, pp. 137–174. Springer, Heidelberg (2009)
7. Das, S., Abraham, A., Konar, A.: Metaheuristic Pattern Clustering – An Overview. In: Das, S., Abraham, A., Konar, A. (eds.) Metaheuristic Clustering. SCI, vol. 178, pp. 1–62. Springer, Heidelberg (2009)
8. Desell, T.J., Anderson, D.P., Magdon-Ismail, M., Newberg, H.J., Szymanski, B.K., Varela, C.A.: An analysis of massively distributed evolutionary algorithms. In: IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2010)
9. Dunn, J.C.: Well separated clusters and optimal fuzzy-partitions. Journal of Cybernetics 4, 95–104 (1974)
10. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. J. Intell. Inf. Syst. 17, 107–145 (2001)

11. Harish, P., Narayanan, P.J.: Accelerating Large Graph Algorithms on the GPU Using CUDA. In: Aluru, S., Parashar, M., Badrinath, R., Prasanna, V.K. (eds.) HiPC 2007. LNCS, vol. 4873, pp. 197–208. Springer, Heidelberg (2007)
12. Hruschka, E.R., Campello, R.J.G.B., Freitas, A.A., De Carvalho, A.C.P.L.F.: A survey of evolutionary algorithms for clustering. Trans. Sys. Man Cyber. Part C 39, 133–155 (2009)
13. Karypis, G., Han, E.H., Kumar, V.: Chameleon: hierarchical clustering using dynamic modeling. Computer 32(8), 68–75 (1999)
14. Kriegel, H.P., Kröger, P., Sander, J., Zimek, A.: Density-based clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1(3), 231–240 (2011)
15. Krömer, P., Platos, J., Snasel, V.: Differential evolution for the linear ordering problem implemented on cuda. In: Smith, A.E. (ed.) Proceedings of the 2011 IEEE Congress on Evolutionary Computation, June 5-8. IEEE Computational Intelligence Society, pp. 790–796. IEEE Press, New Orleans (2011)
16. Krömer, P., Snásel, V., Platos, J., Abraham, A.: Many-threaded implementation of differential evolution for the cuda platform. In: Krasnogor, N., Lanzi, P.L. (eds.) GECCO, pp. 1595–1602. ACM (2011)
17. Langdon, W.B., Banzhaf, W.: A SIMD Interpreter for Genetic Programming on GPU Graphics Cards. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 73–85. Springer, Heidelberg (2008)
18. Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J.: Understanding of internal clustering validation measures. In: 2010 IEEE 10th Int. Conf. on Data Mining (ICDM), pp. 911–916 (December 2010)
19. Luo, L., Wong, M., Hwu, W.M.: An effective gpu implementation of breadth-first search. In: Proc. of the 47th Design Automation Conf., DAC 2010, pp. 52–55. ACM, New York (2010)
20. Pospichal, P., Jaros, J., Schwarz, J.: Parallel Genetic Algorithm on the CUDA Architecture. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcazar, A.I., Goh, C.-K., Merelo, J.J., Neri, F., Preuß, M., Togelius, J., Yannakakis, G.N. (eds.) EvoApplicatons 2010, Part I. LNCS, vol. 6024, pp. 442–451. Springer, Heidelberg (2010)
21. Robilliard, D., Marion, V., Fonlupt, C.: High performance genetic programming on gpu. In: Proc. of the 2009 Workshop on Bio-inspired Algorithms for Distributed Systems, BADS 2009, pp. 85–94. ACM, New York (2009)
22. de Veronese, L., Krohling, R.: Differential evolution algorithm on the gpu with c-cuda. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–7 (July 2010)
23. Zhu, W., Li, Y.: Gpu-accelerated differential evolutionary markov chain monte carlo method for multi-objective optimization over continuous space. In: Proceeding of the 2nd Workshop on Bio-inspired Algorithms for Distributed Systems, BADS 2010, pp. 1–8. ACM, New York (2010)

Memetic Input Variable Selection in Neuro-Genetic Prediction System

Jacek Mańdziuk[1] and Marcin Jaruszewicz[2]

[1] Warsaw University of Technology, Faculty of Mathematics and Information Science, Koszykowa 75, 00-662 Warsaw, Poland
mandziuk@mini.pw.edu.pl, http://www.mini.pw.edu.pl/~mandziuk/
[2] Outbox Ltd., ul. Grójecka 5, 02-019 Warsaw, Poland
jaruszewicz@data.pl

Abstract. This paper describes a hybrid neuro-genetic system applied to short-term prediction of a stock index. Prediction is made by an ensemble of three simple neural networks, each of which processes data selected by the evolutionary algorithm. The input data comes from the three stock markets and additionally includes two exchange rates. Besides the review of results, which have been published in detail elsewhere, some aspects of the system which appeared to be vital for accomplishing high efficiency are thoroughly examined and discussed. These include: autonomous mechanism for extraction of technical analysis patterns, memetic-type local improvement phase in the evolutionary process, and incorporation of domain knowledge into memetic-evolutionary variable pre-selection.

1 Introduction

The prediction system considered in this paper takes advantage of a synergy between the computational power of neural networks and optimization capabilities and flexibility of genetic algorithms (GAs). The latter is further enhanced by the use of several memetic-type local optimizations. The system relies solely on the data extracted with the use of Technical Analysis (TA). Its prediction target is the next day's closing price of the German Stock Exchange (GSE) index DAX. Genetic algorithm chooses variables from a large pool of 370 candidates, which includes mainly those related to the target market with some support of the data coming from two other international markets: New York Stock Exchange (NYSE) and Tokyo Stock Exchange (TSE) with DJIA and NIKKEI 225 (NIKKEI) index, respectively. Finally, there are two exchange rates (€/\$) and (\$/¥) in the pool of available variables. The variables represent past raw index values, transaction volumes, patterns of TA, moving averages and oscillators. All these data is widely used by human brokers relying on TA (see [1] for an introduction to stock market prediction).

The usefulness of input variables selected by the GA is verified by a quick procedure involving approximate neural network training and testing. The input set chosen by the GA is used for 5 trading days only and over the week-end a new set is selected for the next trading week, and so on. This limited usage of the chosen variables stems from high volatility of mutual correlations among input (market) variables.

The system works autonomously with no need for human intervention. On a general note, it can be observed that variables are selected with noticeable sense. Characteristic, repeatable patterns of variables existing in consecutive 5-day prediction periods can be pointed out.

The underlying idea of the system and the main results have been published in [2], and therefore will only be sketched here. The system features also a few interesting modifications to "generic" neuro-evolutionary implementation, which seem to be crucial for its overall high performance. These are:

1. extraction of graphical patterns used in TA,
2. existence of dead/alive chromosomes and the ways of dealing with this phenomenon,
3. "memetic-like" incorporation of domain knowledge into the system and the use of local improvement techniques.

The second issue has been analyzed in detail in [3]. In this paper the focus is on points 1 and 3, which were only briefly mentioned in previous papers.

The rest of this paper is organized as follows: in the next section an overview of the prediction system and the summary of results are presented. In section 3 the algorithm for autonomous extraction and assessment of patterns is presented and section 4 introduces and discusses the memetic improvements to the base neuro-evolutionary system design. The last section concludes the paper.

2 System Overview

The system starts off with pre-selection of the initial pool of data. Part of the data are raw index values (opening, highest, lowest or closing) of the considered stock markets in subsequent days. The other part includes various transformations of these basic indicators, e.g. the relative change of closing value over a certain period of time (5, 10 or 20 trading days), or the averages of the opening values over the last 5, 10 or 20 trading days, etc. The above aggregates provide concise estimation of raw indicators and are known to be useful for financial predictions. In particular, the moving averages are helpful in smoothing local changes and looking for more general trends [4]. A similar reasoning was applied to oscillators - the well known tools in TA. Based on the literature advice [1] eight oscillators, together with the buy/sell signals they generate, were selected for inclusion in the pool of available variables. These were MACD, Williams, Two Averages, IMPET, RSI, ROC, SO and FSO.

All the above-mentioned variables are available for the GA, which is used as a selection mechanism. Each chromosome represents a particular set (list) of variables. Once the GA procedure is completed the best fitted chromosome is extracted and variables it is coding are used for final neural network training. These variables are presumed to represent a close-to-optimal selection for the next week's predictions (a 5-day trading window).

The actual index prediction is performed by the feed-forward neural networks with one hidden layer (of the size equal to the *floor* of the half of the input layer's size) and with one output unit. For example, in the case of 11 input variables (input units) the size of the hidden layer equals 5. This type of architecture was selected based on some number of initial tests. In effect, the network's structure used in the GA operation phase or for subsequent final prediction is completely defined by the number of inputs. The training procedure starts with random weights initialization. A standard back-propagation learning method with momentum is used.

As stated above the chromosome defines the list of variables for a neural network training and prediction. Additionally each chromosome defines a concrete network's architecture (see above). For instance, if a chromosome codes the following four variables: the opening value of DJIA, the RSI oscillator, the average change of DAX opening values in 10 days, and the last day's €/$ exchange rate, then this chromosome defines a $4 - 2 - 1$ network (4 input, 2 hidden and 1 output units). For each input neuron the value of the respective variable is provided. Chromosome's fitness is inverse-proportional to the error generated by trained network. More precisely, for each chromosome several (in the current implementation 3) neural networks with different initial weights, but the same architecture are considered. The ultimate fitness value is calculated with respect to the errors output by all these (3) networks.

GA employs two main operators: mutation and crossover. Mutation operator brings random changes into chromosomes, hence extends *exploration* capabilities of the system. Crossover operator mates two parent chromosomes in the process of creating two new (hopefully better fitted) offspring chromosomes. Since it relies on information coded by the parents is enhances the *exploitation* aspect of the system. A combined crossover method used by the system depends on the common part of the parent chromosomes and random selection of the remaining variables. It is schematically presented in fig. 1. This type of crossover promotes variables which are shared within the population. A rank method is used for selection of the chromosomes for crossover operation (see [2] for details).

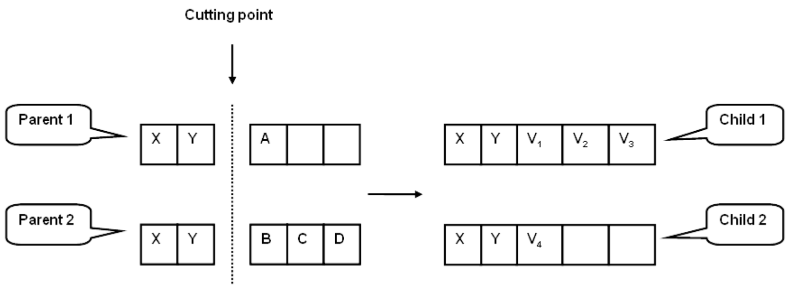


Fig. 1. Combined crossover scheme. Variables V_1, V_2, V_3, V_4 replacing A, B, C, D are randomly selected from the remaining pool of variables (i.e. all except X and Y) on condition that they are pairwise different, i.e. $V_2 \neq V_3, V_3 \neq V_4$ and $V_2 \neq V_4$.

2.1 Summary of Results

The proposed model was compared with four other signal-based prediction models. Each of the models generates *buy* or *sell* signal whenever certain conditions are fulfilled.

Model 1 - *"buy and hold"* strategy: all stocks are bought in the first day and sold in the last one. These are the only two transactions performed by the system. This model is used as a "neutral" point of reference, since its relative profit equals the relative change of index value in the testing period.

Model 2 - assumes that the next day's direction of a change of an index value will be the same as the last (today's) direction.

Model 3 - signals are generated based on the next day's predictions calculated by *our model*.

Model 4 - signals are generated with the use of MACD oscillator [1], which is commonly used in TA.

Model 5 - signals are generated using the actual knowledge of the future (next day's) values. The result of this omnipotent model defines an upper theoretical bound for the prediction score for a one-day prediction horizon.

Each experiment was composed of twenty steps and in each of them some number of neural networks were trained, validated and tested on 290, 5 and 5 samples, respectively. In subsequent steps the samples were shifted forward (in time) by 5 records, i.e. validation samples become a tail of the training set and test samples are used for validation. This way significant data overlap exists between consecutive steps. 290 training samples represent the trading data of one year, approximately.

Thanks to the above-described shifting of the training/testing data, in each experiment there are 100 test samples which represent the period between April 7th, 2004 and August 26th, 2004. Each step starts with creation of a pool of variables initially considered by the system. Afterward the GA finds the best chromosome and then the neural networks coded by that chromosome are trained and used for prediction for the next 5 trading days.

The system's performance depends on the choice of some number of key parameters. In the current implementation they were chosen by hand based on the results of initial simulations. The size of the population was set to 48 and the initial sizes of the chromosomes were restricted to the range between 4 and 11. The number of GA's generations and the number of epochs used for training the nets (in order to calculate fitness of the chromosomes) were equal to 500 and 200, respectively.

The results of one particular experiment are presented in Table 1. Please recall, that the result of Model 1, by its definition, reflects the change of index value in the testing period. Also note, that model 5 takes advantage of the future information (unavailable in real situations) and its role is to define the highest possible benchmark result. Our system visibly outperformed the remaining three models (except for the prothetic one). The experiment was repeated 5 more times under the same settings, except for the initial neural networks' weights, and for the same period of time. The results proved the upper-hand of our system in each case. The average return of our model in these 6 experiments was equal to 5.44%.

Table 1. Return of different tested models

Model	**Return [%]**	**Model**	**Return [%]**	**Model**	**Return [%]**
Model 1	-5.46	Model 2	-7.54	Model 3	9.31
Model 4	-3.59	Model 5	39.56		

3 Autonomous Extraction of Technical Analysis Patterns

One of the most interesting facets of the system is the mechanism for autonomous extraction of patterns from the historical data. These charts represent *formations* used in TA. Generally speaking, these patterns can be divided into two categories depending on whether they indicate holding or changing of the current trend of the predicted variable. Examples of some of the most common patterns are presented in fig. 2. The pattern extraction algorithm comprises three major steps.

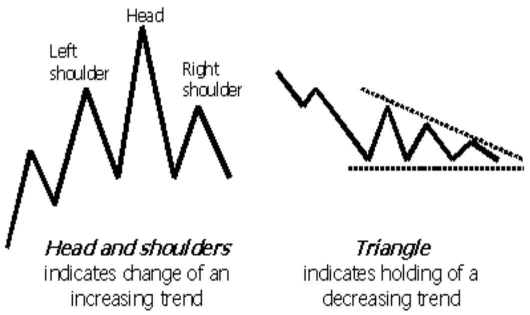

Head
Left shoulder
Right shoulder
Head and shoulders
indicates change of an increasing trend
Triangle
indicates holding of a decreasing trend

Fig. 2. Examples of popular patterns: *Head and shoulders* (left) - indication of changing an increasing trend, and *Triangle* (right) - indication of holding a decreasing trend

First, for a given period of time all minima and maxima in the index (in general: predicted variable) chart are identified. The number of extrema found this way is usually too large for making reasonable classification of potential pattern. Hence, in the second step all minima and maxima which are not "indicative enough" are discarded. More precisely, all vertices for which the absolute value of a difference between them and their closest *preceding* neighbor *of the opposite type* is smaller than a predefined value ϵ are discarded. For example, for a given vertex which represents the minimum, its value is confronted with the preceding (latest) maximum. Finally, in the third step the formations are identified and extracted based on the pre-defined templates. In this phase, two vertices are considered *approximately equal* ($x_1 \approx x_2$) if and only if they fulfill the following condition:

$$\mid x_1 - x_2 \mid \leq \; \epsilon_{eq} \tag{1}$$

and are considered *different* ($x_1 \neq x_2$) if and only if

$$\mid x_1 - x_2 \mid > \; \epsilon_{diff} \tag{2}$$

On the basis of definition (2) two inequality operators are defined:

$$x_1 >' x_2 \qquad \Longleftrightarrow \qquad x_1 > x_2 \ \text{ and } \ x_1 \neq x_2 \tag{3}$$

and

$$x_1 <' x_2 \qquad \Longleftrightarrow \qquad x_1 < x_2 \ \text{ and } \ x_1 \neq x_2 \tag{4}$$

Note, that depending on the choice of ϵ_{eq} and ϵ_{diff} there may exist pairs of vertices which are neither approximately equal nor different or those which are at the same time approximately equal and different. In our experiments the following choices concerning the above confidence coefficients were made:

$$\epsilon = 0.01, \qquad \epsilon_{eq} = 100, \qquad \epsilon_{diff} = 10 \tag{5}$$

based on some preliminary simulations verifying their effectiveness.

The three-step scheme of pattern extraction is presented in figure 3. The set of operators $\approx, >', <'$ defined by equations (1), (3) and (4) is directly applied to define particular formations. Each pattern template is defined based on five consecutive extrema. For example, the one specifying the *Head and shoulders* formation presented in fig. 2 is of the following form:

$$(S_1 >' D_2) \wedge (S_3 >' S_1) \wedge (D_4 <' S_5) \wedge (S_3 >' S_5) \wedge (S_1 \approx S_5) \wedge (D_2 \approx D_4) \tag{6}$$

where $D_i, S_i, i = 1, \ldots, 5$ denote the i-th extremum (minimum or maximum, respectively) and $\wedge$ is a logical conjunction.

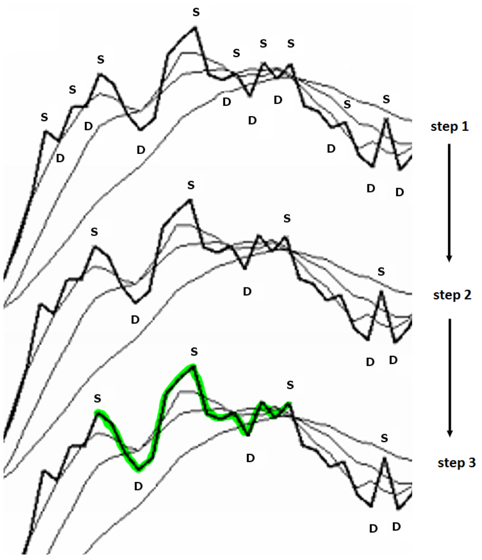


Fig. 3. A three-step pattern extraction procedure. D's and S's denote minima and maxima, respectively.

In our implementation 12 templates were defined for identifying 12 most popular patterns, namely: single/double/triple top/bottom (C), ascending/descending triangle (H), two types of flag charts (H), and head and shoulders top/bottom (C). The letters in parentheses denote indication of C - changing or H - holding the trend by a given formation. In the period of 1995/01/30 - 2004/09/01 our system extracted 1289 formations of the above types, which gives the average of 11.2 per month.

Once extracted, the charts form input variables available to the system in the following way. Each prognosis of changing or holding the trend is assigned a value of -1 and 1, respectively. For each day all indications stemming from the extracted patterns (due to overlaps, there may be more than one pattern valid for a given trading day) are summed up and the resulting value, after appropriate normalization, is placed as the single input value (in the same way as, for example, the DAX opening value or the RSI oscillator, etc.). Similarly to the case of oscillators the impact of a given formation is valid for the consecutive 5 days *after* its identification. This way, the system is not biased by pre-mature application of a given formation before its actual completion (which is indispensable for a formation identification in our system).

3.1 The Relevance of the Formation-Based Data

The assessment of the proposed pattern extraction mechanism can be generally made in two ways. On the one hand, one can perform comparative prediction experiments with the extraction mechanism switched off. On the other hand, one may take advantage of the statistical data related to the frequencies of choosing various types of input variables by the system. For several reasons we have adopted the latter approach. The main motivation was the possibility of having a comprehensive view on the issue of what kind of variables seem to be the most prominent, and whether the choices made by the system are in line with human experience and intuition.

The percentage of variables of various types picked by our system is presented in Table 2. From the table it is clear that the closing value of NYSE seems to be much more important than the respective value of TSE. Moreover, both the exchange rates are also over-represented compared to their frequency of occurrence in the traing/test data. The frequency of choosing the pattern-representing variable was around 2.70% during the tests. This result places this data between the currency exchange rates on the frequency scale (please note that since the pattern-based data is represented by a single variable, its *a priori* probability of occurrence is the same as those of the currency exchange rates).

Table 2. Percentage use of variables

Source of variable	GSE	NYSE	TSE	\$/¥	€/\$
Frequency of occurrence in results [%]	84.11	6.62	3.97	3.31	1.99
Availability for the algorithm [%]	94.60	1.35	1.35	1.35	1.35

4 Memetic-Like Improvements to the Prediction System

Richard Dawkins in his seminal book entitled “The selfish Gene” [5] refers to memes as “basic units of cultural transmission via imitation”. Memes are therefore capable of transmitting information between society members (entities) *also within the same generation* through direct interaction. This in-generation ability to convey cultural heritage makes memetic computing an interesting complementary method to genetic/evolutionary algorithms. In terms of computational models “meme has been typically perceived as individual learning procedures, adaptive improvement procedures or local search operators that enhance the capability of population based search algorithms” [6]. Due to the space limitation we are unable to delve more into memetic computing paradigm and its applications. The interested reader will find a comprehensive overview and a thorough discussion on memetic computing in [7,8].

As we have mentioned in the Introduction the pure neuro-evolutionary schemata of our system was enhanced by introducing several memetic-like improvements, which - together with originally developed pattern extraction algorithm - strongly contributed to the overall high efficiency of the system. The memetic improvements are threefold and all of them concentrate on the input variable selection process, which is vital for the system's performance.

First of all, based on our experience and general knowledge on prediction (and stock market prediction in particular) we arbitrarily forced the presence of the closing DAX value of the previous day. This value has the highest correlation with the predicted variable (next day's closing value of DAX) and for obvious reasons its presence in the prediction process is crucial.

The second improvement consists in application of a priori defined, local search algorithm after completion of the main system's procedure. More precisely, if we denote by W the set of all variables available to the system, by $C = \{c_1, \ldots, c_k\}$ the best chromosome found by the system, and by $R = \{r_1, \ldots, r_m\}$ the set of all remaining variables, i.e. $R = W - C$, then the following $k \cdot m$ chromosomes $C_i^j = \{c_1, \ldots, c_{i-1}, r_j, c_{i+1}, \ldots, c_k\}$ are assessed by neural networks in exactly the same way as the chromosomes generated during the experiment. The best-fitted chromosome among $\{C\} \cup \{C_i^j \mid i = 1, \ldots, k; j = 1, \ldots, m\}$ is chosen as the final system's output.

Approximately in 60% of the prediction steps the system benefited from the above-described local improvement scheme. In the remaining cases the best chromosome (the solution) was found during the main part od the system's performance and was not further improved in the local search phase.

It is worth to note, that all variables present in the chromosome *including the closing value of DAX* (the forced variable) were exposed to the above procedure. However, there was not even a single case in which this specific variable was exchanged with another one. Such an observation further justifies forcing its presence in each chromosome. It is worth noting that except for the closing value of DAX there are three other variables which were never exchanged in this local improvement phase. These are three oscillators related to the GSE market:

Stochastic, IMPET10 and IMPET20, where IMPETn denotes IMPET oscillator based on n days.

The third memetic improvement is an adaptive, self-tuning procedure of missing variables selection in the crossover operation. Please recall that, as stated in section 2, in the crossover operation used in our implementation, only the common parts of the chromosomes are unconditionally promoted to the next generation. The other variables (denoted $V_1, \ldots, V_4$ in figure 1) are randomly chosen from the pool of all the remaining variables. This selection, however, is not uniform. For the sake of exploitation of knowledge already possessed by the system concerning the potential usefulness of particular variables, they are divided into the following three categories:

– *Best Chromosomes Variables* - this set is composed of all variables which were present in the best chromosomes (in all iterations, one chromosome per iteration) found in the current prediction step;
– *Often Selected Variables* - this set is composed of all variables which are not best chromosome ones and whose frequency of occurrence in the current generation (in all chromosomes) is above-average;
– *Remaining Variables* - this set is composed of all variables which do not belong to any of the above two sets.

The probabilities of choosing a variable from a particular set were equal to $\frac{4}{9}$, $\frac{1}{3}$, and $\frac{2}{9}$, respectively for *best chromosomes*, *often selected*, and the *remaining* variables. The above categorization took place after some number of initial iterations when differences between the frequencies of choices became indicative. During the initial phase the preferences for variable selections were uniform.

The memetic mechanism for crossover variable selection works very efficiently and its inclusion in the purely neuro-genetic process leads to overall improvement of the system. Certainly one needs to be careful with assigning selection probabilities to the variable-sets. If the preference for the first one were too high a harmful tendency for premature convergence might appear. The choice made in our experiments works very well, especially when aligned with the procedure of gradual elimination of "dead" (i.e. useless) chromosomes described in [3].

5 Conclusions

A neuro-genetic system applied to prediction of the DAX index closing value, is presented and examined in this paper. Due to changing in time dependencies between financial variables and indicators (raw market values, oscillators, moving averages, etc.) and based on the underlying assumption that the usefulness of any indicator is limited in time, our system relies on frequent GA-based input data re-selection.

A standard GA procedure is enhanced by adding a new type of crossover operator (figure 1), the autonomous procedure for extraction of relevant patterns from the historical plot of DAX index and by introduction of several memetic improvements.

The results are repeatable and encouraging. The choices of input variables made by the GA are reasonable and explainable by the experts (brokers). Variables which are no longer useful for prediction are discarded from the current input set. On the other hand some of them remain in the input set for a long time proving their usefulness and importance. The system outperforms other trading models used for comparison.

Both, pattern extraction algorithm and memetic enhancements play a vital role in accomplishing high quality prediction. In particular the system benefits from a one-step local search procedure after completion of the main neuro-genetic algorithm. Categorization of variables into three types (best chromosomes, often selected, other) and applying different probability of selection to each type of them leads to deeper exploration of the most promising directions in the input variables space.

References

1. Murphy, J.: Technical Analysis of the Financial Markets. New York Instituite of Finance (1999)
2. Mańdziuk, J., Jaruszewicz, M.: Neuro-genetic system for stock index prediction. Journal of Intelligent & Fuzzy Systems (2), 93–123 (2011)
3. Mańdziuk, J., Jaruszewicz, M.: "Dead" Chromosomes and Their Elimination in the Neuro-Genetic Stock Index Prediction System. In: Leung, C.S., Lee, M., Chan, J.H. (eds.) ICONIP 2009, Part II. LNCS, vol. 5864, pp. 601–610. Springer, Heidelberg (2009)
4. Schwager, J.D.: Getting started in Technical Analysis. John Wiley & Sons (1999)
5. Dawkins, R.: The selfish gene. Oxford University Press, Oxford (1976)
6. Feng, L., Ong, Y.S., Tan, A.H., Chen, X.S.: Towards human-like social multi-agents with memetic automaton. In: IEEE CEC, pp. 1092–1099 (2011)
7. Chen, X.S., Ong, Y.S., Lim, M.H., Tan, K.C.: A multi-facet survey on memetic computation. IEEE Transactions on Evolutionary Computation 15(5), 591–607 (2011)
8. Ong, Y.S., Lim, M.H., Chen, X.S.: Research frontier: Memetic computation - past, present & future. IEEE Computational Intelligence Magazine 5(2), 24–36 (2010)

Learning Rule for TSK Fuzzy Logic Systems Using Interval Type-2 Fuzzy Subtractive Clustering

Binh Huy Pham, Hai Trung Ha, and Long Thanh Ngo

Department of Information Systems, Faculty of Information Technology,
Le Quy Don Technical University, No 100, Hoang Quoc Viet, Hanoi, Vietnam
{huybinhhvhc,hatrunghai1982,ngotlong}@gmail.com

Abstract. The paper deals with an approach to model TSK fuzzy logic systems (FLS), especially interval type-2 TSK FLS, using interval type-2 fuzzy subtractive clustering (IT2-SC). The IT2-SC algorithm is combined with least square estimation (LSE) algorithms to pre-identify a type-1 FLS form from input/output data. Then, an interval type-2 TSK FLS can be obtained by considering the membership functions of its existed type-1 counterpart as primary membership functions and assigning uncertainty to cluster centroids, standard deviation of Gaussian membership functions and consequence parameters. Results is shown in comparison with the approach based on type-1 subtractive clustering algorithm.

Keywords: subtractive clustering, type-2 fuzzy sets, fuzzy logic system, TSK model.

1 Introduction

TSK fuzzy logic systems (TSK FLSs) have widely been deployed in various real applications especially in model-based control and model-based fault diagnosis. TSK qualitative modelling, as known as TSK modelling, was proposed in an effort to develop a systematic approach to generating fuzzy rules from a given input-output data set [6,7]. When, the identification of a TSK FLS using clustering involves formation of clusters in the data space and translation of these clusters into TSK rules such that the model obtained is closer to the system to be identified [4,5]. However, in most real data exists uncertainty and vagueness which cannot be appropriately managed by type-1 fuzzy sets. Meanwhile, type-2 fuzzy sets allow us to obtain desirable results in designing and managing uncertainty. Mendel et al [1,2,3] extended previous studies and established a complete type-2 fuzzy logic theory with the handling of uncertainties. On the basis, type-2 TSK FLS was presented [16].

One of the important tasks to design a fuzzy system is how to determine the number of rules (structure identification). There are two approaches to generate initial fuzzy rules: manually and automatically. In the automatically approaches, the basic idea is to estimate fuzzy rules through learning process from input-output sample data. An automatic data-driven based method for generating the initial fuzzy rules is Chius subtractive clustering algorithm (SC) [6]. When, subtractive clustering algorithm is combined with least squares estimation algorithm to design TSK FLSs [8]. Then, an interval type-2 TSK FLS can be obtained by considering the membership functions of its existed type-1 counterpart as primary membership functions and assigning uncertainty to cluster centroids and consequence parameters [9]. In this way, clustering results of SC decides the structure of fuzzy systems. Interval type-2 fuzzy subtractive clustering (IT2-FSC) [15] is extension of SC algorithms to handle uncertainty.

In subtractive clustering algorithms, setting subtractive clustering parameters are very influential to the results of clustering. This paper deals with an approach to model type-2 TSK FLS from input/output dataset. Interval type-2 fuzzy subtractive clustering is used to determine the number of rules and to learn rule-base from dataset. IT2-FSC is also combined with LSE algorithm to estimate parameters for designing interval type-2 TSK FLS. Results on function approximation is shown that the proposed approach to obtain accuracy and simple TSK models.

The remainder of this paper is organized as follows. In Section 2 introduces briefly type-2 fuzzy sets, interval type-2 fuzzy subtractive clustering. In section 3, we discuss how to using interval type-2 fuzzy subtractive clustering algorithm to design TSK FLS and extend interval type-2 TSK FLS from type-1 TSK FLS. In section 4, we provide several experiments to show the validity of our proposed method. Finally, section 5 gives the summaries and conclusions.

2 Interval Type-2 Fuzzy Logic Systems

2.1 Type-2 Fuzzy Sets

A type-2 fuzzy set in X is denoted $\tilde{A}$, and its membership grade of $x \in X$ is $\mu_{\tilde{A}}(x,u)$, $u \in J_x \subseteq [0,1]$, which is a type-1 fuzzy set in [0, 1]. The elements of domain of $\mu_{\tilde{A}}(x,u)$ are called primary memberships of x in $\tilde{A}$ and memberships of primary memberships in $\mu_{\tilde{A}}(x,u)$ are called secondary memberships of x in $\tilde{A}$.

Definition 1. *A $type-2$ $fuzzy$ set, denoted $\tilde{A}$, is characterized by a type-2 membership function $\mu_{\tilde{A}}(x,u)$ where $x \in X$ and $u \in J_x \subseteq [0,1]$, i.e.,*

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \tag{1}$$

in which $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$.

At each value of x, say $x = x'$, the 2-D plane whose axes are u and $\mu_{\tilde{A}}(x',u)$ is called a *vertical slice* of $\mu_{\tilde{A}}(x,u)$. A *secondary membership function* is a vertical slice of $\mu_{\tilde{A}}(x,u)$. It is $\mu_{\tilde{A}}(x = x',u)$ for $x \in X$ and $\forall u \in J_{x'} \subseteq [0,1]$, i.e.

$$\mu_{\tilde{A}}(x = x', u) = \int_{u \in J_{x'}} f_{x'}(u)/u, J_{x'} \subseteq [0,1] \tag{2}$$

in which $0 \leq f_{x'}(u) \leq 1$.

Type-2 fuzzy sets are called an interval type-2 fuzzy sets if the secondary membership function $f_{x'}(u) = 1 \; \forall u \in J_x$ that are defined as follows:

Definition 2. *An interval type-2 fuzzy set $\tilde{A}$ is characterized by an interval type-2 membership function $\mu_{\tilde{A}}(x,u) = 1$ where $x \in X$ and $u \in J_x \subseteq [0,1]$, i.e.,*

$$\tilde{A} = \{((x,u),1) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \tag{3}$$

Uncertainty of $\tilde{A}$, denoted FOU, is union of primary functions i.e. $FOU(\tilde{A}) = \bigcup_{x \in X} J_x$. Upper/lower bounds of membership function (UMF/LMF), denoted $\overline{\mu}_{\tilde{A}}(x)$ and $\underline{\mu}_{\tilde{A}}(x)$, of $\tilde{A}$ are two type-1 membership function and bounds of FOU.

2.2 Type-1 TSK Fuzzy Logic Systems

A generalized type-1 TSK model is described by fuzzy IF-THEN rules which represent input-output relations of a system. For a multi-input-single-output (MISO) first order type-1 TSK model, its l^{th} rule can be expressed as follows:

R^l : IF x_1 is F_1^l AND x_2 is F_2^l AND ... AND x_n is F_n^l THEN

$$w^l = c_o^l + c_1^l x_1 + c_2^l x_2 + ... + c_n^l x_n \tag{4}$$

in which $x_i (i = 1, ..., n)$ are linguistic variables, $F_i^l (i = 1, ..., n)$ are type-1 fuzzy sets, w^l is output from the l^{th} IF-THEN rule, $c_i^l (i = 0, 1, ..., n)$ are consequent parameters.

The output of a TSK FLS is computed as following steps:

- Calculating degree of firing of l^{th} rule as:

$$f^l = \mu_1^l(x_1) \wedge \mu_2^l(x_2) \wedge \ldots \wedge \mu_n^l(x_n) \tag{5}$$

where $\wedge$ is a conjunction operator and a t-norm, can be minimum or product.

- Calculating the output from the l^{th} IF-THEN rule of M rules FLS:

$$w^l = c_o^l + c_1^l x_1 + c_2^l x_2 + ... + c_n^l x_n \tag{6}$$

- Calculating the output of FLS by weighted averaging:

$$\mathrm{W} = \frac{\sum_{i=1}^{k} f^i \mathrm{w}^i}{\sum_{i=1}^{k} f^i} \tag{7}$$

2.3 Interval Type-2 TSK Fuzzy Logic Systems

An interval type-2 TSK model includes M-rules, n-inputs, its l^{th} fuzzy IF-THEN rule can be expressed as bellow:

R^l : IF x_1 is $\tilde{F}_1^l$ AND IF x_2 is $\tilde{F}_2^l$ AND ... AND IF x_k is $\tilde{F}_k^l$ THEN

$$\tilde{w}l = \tilde{C}_0^l + \tilde{C}_1^l x_1 + \tilde{C}_2^l x_2 + ... + \tilde{C}_n^l x_n \tag{8}$$

in which $x_i(i = 1, ..., n)$ are linguistic variables, w^l is output from the l^{th} IF-THEN rule; $\tilde{C}_i^l(i = 0, 1, ..., n)$ type-1 fuzzy sets are consequent parameters and $\tilde{C}_i^l = [c_i^l - s_i^l, c_i^l + s_i^l]$ with c_i^l denotes the centroid of $\tilde{C}_i^l$ and s_i^l denotes the spread of $\tilde{C}_i^l$; $\tilde{F}_1^l(i = 0, 1, ..., n)$ are interval type-2 fuzzy sets and $\tilde{\mu}_i^l = [\underline{\mu}_{\mathrm{i}}^{\mathrm{l}}, \overline{\mu}_{\mathrm{i}}^{\mathrm{l}}]$.

Interval type-2 TSK FLS is computed as the following steps:

- Degree of firing of l^{th} rule $f^l = [\underline{f}^l, \overline{f}^{\mathrm{l}}]$ with

$$\begin{aligned} \underline{f}^l &= \underline{\mu}_1^l(x_1) \wedge \underline{\mu}_2^l(x_2) \wedge \ldots \wedge \underline{\mu}_n^l(x_n) \\ \overline{f}^l &= \overline{\mu}_1^l(x_1) \wedge \overline{\mu}_2^l(x_2) \wedge \ldots \wedge \overline{\mu}_n^l(x_n) \end{aligned} \tag{9}$$

- The output from the l^{th} IF-THEN rule of M rules: $\tilde{\mathrm{w}}^l = [\mathrm{w}_{\mathrm{L}}^{\mathrm{l}}, \mathrm{w}_{\mathrm{R}}^{\mathrm{l}}]$ with

$$\mathrm{w}_{\mathrm{L}}^{\mathrm{l}} = \sum_{j=1}^{n} c_j^l x_j + c_0^l - \sum_{j=1}^{n} s_j^l x_j - s_0^l \tag{10}$$

$$\mathrm{w}_{\mathrm{R}}^{\mathrm{l}} = \sum_{j=1}^{n} c_j^l x_j + c_0^l + \sum_{j=1}^{n} s_j^l x_j + s_0^l \tag{11}$$

- Calculating output of FLS by weighted averaging of individual rules contributions:

$$\mathrm{w}_{\mathrm{L}} = \frac{\sum_{j=1}^{k} \underline{f}^j * \mathrm{w}_{\mathrm{L}}^{\mathrm{j}}}{\sum_{j=1}^{k} \underline{f}^j} \; and \; \mathrm{w}_{\mathrm{R}} = \frac{\sum_{j=1}^{k} \overline{f}^j * \mathrm{w}_{\mathrm{R}}^{\mathrm{j}}}{\sum_{j=1}^{k} \overline{f}^j} \tag{12}$$

3 Rule Extraction for Interval Type-2 TSK FLS

The problem of identification of TSK model is divided into two sub-tasks: Learning the antecedent part of the model, which consists on the determination of centroids and spreads of membership functions by using IT2-FSC; and Learning the parameters of the linear subsystems of the consequent by using LSE algorithm.

3.1 Learning Rule Antecedents

Subtractive clustering estimated the potential of a data point as a cluster centroid based on the density of surrounding data points, which is actually based on the distance between the data point with the remaining data points. In addition, we must set four parameters: accept ratio $\overline{\varepsilon}$, reflect ratio $\underline{\varepsilon}$, cluster radius r_a and squash factor η (or r_b) [4,5]. The choice of parameters have greatly influences to results of clustering. SC includes various types of uncertainty as distance measure, initialization parameters... So we consider a fuzziness parameters that control the distribution of data points into clusters by making the parameter m in the density function to calculate the potential of a data point [15]. Membership degree of a point in the k^{th} cluster centroid is defined as following formula:

$$\mu_{ik} = e^{-\frac{4}{r_a^2}(x_i - x_k)^{\frac{2}{m-1}}} \tag{13}$$

where x_k is the k^{th} cluster centroid.

According to the formula (13), membership value of a data point in the k^{th} cluster centroid depends on the position of the k^{th} cluster and the fuzziness parameter m. Thus, the fuzziness parameter m is the most uncertainty element in the expanded subtractive clustering algorithm. Therefore, to design and manage the uncertainty for fuzziness parameter m, pattern set to interval type-2 fuzzy sets is extended using two fuzzifiers m_1 and m_2, which creates a footprint of uncertainty (FOU) for the fuzziness parameter m. Then the degree of membership of the k^{th} cluster centroid is defined as the following formula:

$$\begin{cases} \overline{\mu}_{ik} = e^{-\frac{4}{r_a^2}(x_i - x_k)^{\frac{2}{m_1-1}}} \\ \underline{\mu}_{ik} = e^{-\frac{4}{r_a^2}(x_i - x_k)^{\frac{2}{m_2-1}}} \end{cases} \tag{14}$$

Two density functions are computed the potential of each data point as follows:

$$\begin{cases} \overline{P}_i = \sum_{j=1}^{n} e^{-\frac{4}{r_a^2}(x_j - x_i)^{\frac{2}{m_1-1}}} \\ \underline{P}_i = \sum_{j=1}^{n} e^{-\frac{4}{r_a^2}(x_j - x_i)^{\frac{2}{m_2-1}}} \end{cases} \tag{15}$$

The centroids are identified by the formula (15) and type-reduction for centroids is done as bellows:

$$P_i = \frac{\overline{P}_i * m_1 + \underline{P}_i * m_2}{m_1 + m_2} \tag{16}$$

When the k^{th} cluster centroid is identified, the density of all data points is revised by using the following formula:

$$\begin{cases} \underline{P}_i^{sub} = P_k^* \sum_{j=1}^{n} e^{-\frac{4}{r_b^2} d_{\text{ij}}^{\frac{2}{m_1-1}}} \\ \overline{P}_i^{sub} = P_k^* \sum_{j=1}^{n} e^{-\frac{4}{r_b^2} d_{\text{ij}}^{\frac{2}{m_2-1}}} \\ P_i^{sub} = \frac{\underline{P}_i^{sub} * m_1 + \overline{P}_i^{sub} * m_2}{m_1 + m_2} \\ P_i = P_i - P_i^{sub} \end{cases} \tag{17}$$

Because, each cluster centroid is representative of a characteristic behaviour of the system, the resulting cluster centroids are used as parameters of the antecedent parts defining the focal points of the rules of the model. Then clustering results of IT2-SC decides the structure of fuzzy systems.

3.2 Learning Rule Consequent Using LSE Algorithm

The output of type-1 TSK model is determined by the formula (7).

Suppose that:

$$\delta^i = \frac{f^i}{\sum_{i=1}^{k} f^i} \tag{18}$$

When:

$$\mathrm{W} = \sum_{i=1}^{k} \delta^i w^i \tag{19}$$

For a given set of m input-output data points. The equations can be obtained as:

$$\begin{array}{l} W^1 = \sum_{i=1}^{k} \delta^i c_0^i + \sum_{i=1}^{k} x_i \delta^i c_1^i + ... + \sum_{i=1}^{k} x_i \delta^i c_n^i \\ W^2 = \sum_{i=1}^{k} \delta^i c_0^i + \sum_{i=1}^{k} x_i \delta^i c_1^i + ... + \sum_{i=1}^{k} x_i \delta^i c_n^i \\ ... \\ W^m = \sum_{i=1}^{k} \delta^i c_0^i + \sum_{i=1}^{k} x_i \delta^i c_1^i + ... + \sum_{i=1}^{k} x_i \delta^i c_n^i \end{array} \tag{20}$$

The formula (20) can be taken a standard form: $AP = W$, where A is a constant matrix (known), W is a matrix of the output and P is a matrix of parameters to be estimated. We use least square estimation problem to determine P as:

$$P = (A^T A)^{-1} A^T W \tag{21}$$

3.3 Building for Interval Type-2 TSK FLS

An interval type-2 TSK FLS can be obtained by considering the membership functions (MFs) of its existed type-1 counterpart as primary MFs and assigning uncertainty to cluster centroids, standard deviation of Gaussian MF and consequence parameters with membership functions of type-1 FLS is defined by

$$F_j^l = N(x_j, x_l^*, \sigma) = \exp\left[-\frac{1}{2}\left(\frac{x_j - x_l^*}{\sigma}\right)^2\right] \tag{22}$$

in which $\sigma = \frac{r_a}{2\sqrt{2}}$.

By doing that, cluster centroids are expanded from a certain point to a fuzzy number as follows:

$$\tilde{x}_l^* = [\mathrm{x}_l^*(1 - \mathrm{a}^l), \mathrm{x}_l^*(1 + \mathrm{a}^l)] = [\underline{\mathrm{x}}_l^*, \bar{\mathrm{x}}_l^*] \tag{23}$$

where a^l is the spread percentage of cluster centre x_l^* in Fig.1. Then, the upper membership function, $\overline{\mu}_j^l(\mathrm{x_j})$, is defined by

$$\overline{\mu}_j^l(x_j) = \begin{cases} N(x_j, \overline{x}_l^*, \sigma), x_j > \overline{x}_l^* \\ 1, \underline{x}_l^* <= x_j <= \overline{x}_l^* \\ N(x_j, \underline{x}_l^*, \sigma), x_j < \underline{x}_l^* \end{cases} \tag{24}$$

And the upper membership function, $\underline{\mu}_j^l(\mathrm{x_j})$, is defined by

$$\underline{\mu}_j^l(x_j) = \begin{cases} N(x_j, \underline{x}_l^*, \sigma), x_j >= \frac{\underline{x}_l^* + \overline{x}_l^*}{2} \\ N(x_j, \overline{x}_l^*, \sigma), x_j < \frac{\overline{x}_l^* + \underline{x}_l^*}{2} \end{cases} \tag{25}$$

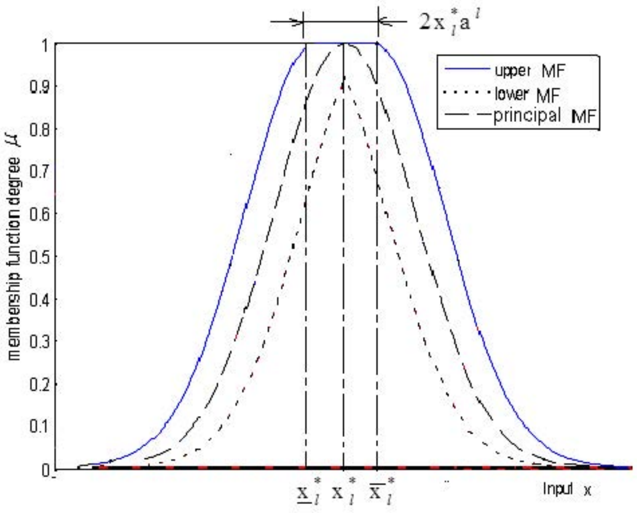


Fig. 1. Spread and centroid of Gaussian Type-2 FSs

Whereas, consequent parameters are obtained by expanding consequent parameters from its type-1 TSK model to fuzzy numbers by formula(26) where b_j^l is the spread percentage of fuzzy numbers $\tilde{c}_j^l$

$$\tilde{c}_j^l = c_j^l (1 \pm b_j^k) \tag{26}$$

The TSK FLS modelling algorithm be proposed as below:

Step 1: Use our proposed IT2-SC algorithm combined with least squares estimation algorithms to pre-identify a type-1 FLS form from input/output data.

Step 2: Calculate root-mean-square-error (RMSE), if RMSE is bigger than expected error limitation, go to Step 3. If not, go to Step 5, which means the model is acceptable, no need to use type-2 TSK model.

Step 3: Expand type-1 TSK model to type-2 TSK model:

- Spread cluster centroid to expanding premise membership functions from type-1 fuzzy sets to type-2 fuzzy sets using formulas (24) and (25)
- Spread the parameters of consequence to expanding parameters of consequences from certain value to fuzzy numbers below formula (26).

Step 4: Identify a type-2 TSK FLS

Step 5: Output the results of TSK FLS modelling.

The results of TSK FLS modeling algorithm are a type-1 or type-2 TSK FLS model.

4 Experimental Results

We consider the problem of type-1 TSK fuzzy model for approximating the following non-linear function:

$$y = (x - 2.5)^3 + x + 1 \tag{27}$$

where $x \in [0, 4]$, we used equally spaced values to generate 1001 data points. Here, we randomly selected 751 as training data and 250 as testing data. Table 1 describes four rules type-1 TSK model by using IT2-SC algorithm with initialization parameters, respectively,$\overline{\varepsilon} = 0.5$, $\underline{\varepsilon} = 0.15$, $r_a = 0.5$, $\eta = 1.25$ and two fuzzifiers: $m_1 = 1.85$ and $m_2 = 2.15$.

Table 1. Results of type-1 TSK model based on SC of Chiu and our proposed IT2-SC

Rules	If x then $y = p_1 * x + p_0$	
	TSK model based on SC of Chiu	TSK model based on our proposed IT2-SC
1	If $x = \exp(-\frac{1}{2}\left(\frac{x-2.464}{0.70711}\right)^2)$ then $y = 4.57271x - 6.91789$	If $x = \exp(-\frac{1}{2}\left(\frac{x-2.42}{0.70711}\right)^2)$ then $y = 6.4900x - 12.2933$
2	If $x = \exp(-\frac{1}{2}\left(\frac{x-0.9}{0.70711}\right)^2)$ then $y = 12.3745x - 24.1487$	If $x = \exp(-\frac{1}{2}\left(\frac{x-0.824}{0.70711}\right)^2)$ then $y = 25.3741x - 49.8226$
3	If $x = \exp(-\frac{1}{2}\left(\frac{x-3.724}{0.70711}\right)^2)$ then $y = 7.48193x - 21.9791$	If $x = \exp(-\frac{1}{2}\left(\frac{x-3.728}{0.70711}\right)^2)$ then $y = 8.04252x - 24.3523$
4	If $x = \exp(-\frac{1}{2}\left(\frac{x-0.188}{0.70711}\right)^2)$ then $y = 28.9666x - 10.2412$	If $x = \exp(-\frac{1}{2}\left(\frac{x-0.148}{0.70711}\right)^2)$ then $y = 42.1097x + 3.72731$

In this case, the RMSE-training is 0.01534 and the RMSE-testing is 0.01511. Type-1 SC algorithm is also used for identification a type-1 TSK model with initialization parameters, respectively,$\overline{\varepsilon} = 0.5$, $\underline{\varepsilon} = 0.15$, $r_a = 0.5$, $\eta = 1.25$. Then, the RMSE-training is 0.02196 and the RMSE-testing is 0.02184. We can see that TSK model generated by our proposed method has result accuracy with smaller RMSE. In Fig. 2, both training data and testing data, TSK model generated by our proposed method has result as quite same as real system.

We can change value of two fuzzifiers to obtain better type-1 TSK model. In table 2, we see that type-1 TSK model based on our proposed IT2-SC with values of two fuzzifiers, $m_1 = 1.85$ and $m_2 = 2.15$, has the best result. RMSE on training data and RMSE on testing data is quite small. The figure 2 shows plots of obtained TSK model on training and testing data.

We consider two rules type-1 TSK model by using IT2-SC algorithm with two fuzzifiers: $m_1 = 1.3$ and $m_2 = 2.7$. The type-1 TSK model is described in table 3. In this model, the RMSE on training data is 0.46258 and RMSE on test data is 0.50296. Two rules of type-2 TSK model obtain from type-1 TSK model by using above described spread of fuzzy numbers. The system gains RMSE of training and testing data, respectively, are 0.84944 and 0.35684.

Table 2. Result of type-1 TSK model by different values of two fuzzifiers

m_1 and m_2	Number of rules	RMSE_training	RMSE_testing
1.95 and 2.05	4	0.02262	0.02113
1.9 and 2.1	4	0.01977	0.01811
1.85 and 2.15	4	0.01534	0.01511
1.8 and 2.2	3	0.3071	0.3031
1.7 and 2.3	3	0.3241	0.3204
1.6 and 2.4	2	0.4534	0.4531
1.5 and 2.5	2	0.42705	0.43769
1.4 and 2.6	2	0.42245	0.43887
1.3 and 2.7	2	0.4704	0.48038
1.2 and 2.8	1	2.1763	2.1285

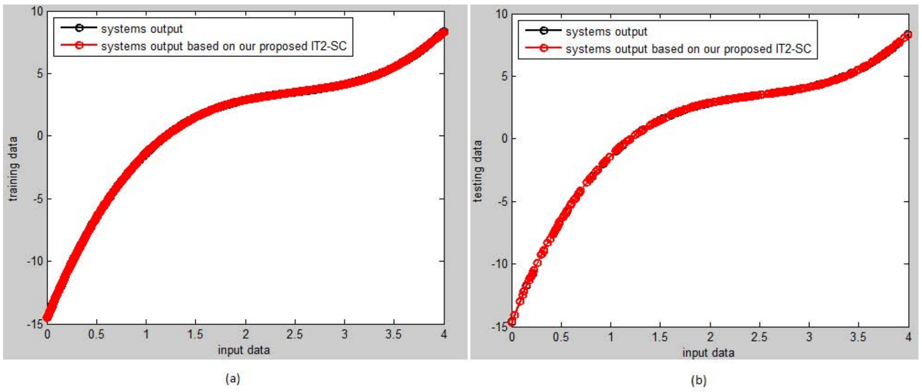


Fig. 2. Result of TSK model; (a): On training data; (b): On testing data

Table 3. Type-1 TSK model and type-2 TSK model

Rules	If x then $y = p_1 * x + p_0$	
	type-1 TSK model	type-2 TSK model
1	If $x = \exp(-\frac{1}{2}\left(\frac{x-2.004}{0.7064}\right)^2)$ then $y = 1.72375x - 1.48008$	If $x = \exp(-\frac{1}{2}\left(\frac{x-2.004*(1-10\%)}{0.7064}\right)^2)$ then $y = 1.72375*(1-20\%)x - 1.48008*(1-20\%)$
2	If $x = \exp(-\frac{1}{2}\left(\frac{x}{0.7064}\right)^2)$ then $y = 12.3839x - 13.9905$	If $x = \exp(-\frac{1}{2}\left(\frac{x}{0.7064}\right)^2)$ then $y = 12.3839*(1-20\%)x - 13.9905*(1-20\%)$

5 Conclusion

The paper presents a new approach to design TSK model. Here, we used an our proposed IT2-SC combined with least squares estimation algorithm. The result of experiments is shown the validity of our proposed method.

For the future works, we will improve the computational performance by speeding up the algorithm using GPU.

References

1. Mendel, J., John, R.: Type-2 fuzzy set made simple. IEEE Trans. on Fuzzy Systems 10(2), 117–127 (2002)
2. Karnik, N., Mendel, J.M.: Operations on Type-2 Fuzzy Sets, Fuzzy Sets and Systems, vol. 122, pp. 327–348 (2001)
3. Mendel, J.M., John, R.I., Liu, F.: Interval Type-2 Fuzzy Logic Systems Made Simple. IEEE Trans. on Fuzzy Systems 14(6), 808–821 (2006)
4. Chiu, S.L.: Fuzzy Model Identification Based on Cluster Estimation. Journal on Intelligent Fuzzy Systems 2, 267–278 (1994)
5. Chiu, S.L.: Extracting Fuzzy Rules from Data for Function Approximation and Pattern Classification. In: Dubois, H., Prade, R., Yager, R. (eds.) Fuzzy Information Engineering: A Guided Tour of Applications. John Wiley & Sons (1997)
6. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. on Sys., Man, and Cyb. 15(1), 116–132 (1985)
7. Sugeno, M., Kang, G.: Structure identification of fuzzy model. Fuzzy Sets and Systems 28(1), 15–33 (1988)
8. Ren, Q., Baron, L., Balazinski, M., Jemielniak, K.: Tool condition monitoring using the TSK fuzzy approach based on subtractive clustering method. News Frontiers in Applied Artificial Intelligence, pp. 52–61. Springer, Berlin (2008)
9. Ren, Q., Baron, L., Balazinski, M.: Type-2 Takagi-Sugeno-Kang fuzzy logic modeling using subtractive clustering. In: Proceedings of the NAFIPS, pp. 1–6 (2006)
10. Ren, Q., Baron, L., Balazinski, M.: Uncertainty prediction for tool wear condition using type-2 TSK fuzzy approach. In: Proceeding of the 2009 IEEE Int' Conf. on Systems, Man, and Cybernetics (IEEE-SMC 2009), pp. 666–671 (2009)
11. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. on Sys., Man, and Cyb. 15(1), 116–132 (1985)
12. Zhang, W.B., Liu, W.J.: IFCM: Fuzzy Clustering for Rule Extraction of Interval Type-2 Fuzzy Logic System. In: The 46th IEEE Conf. on Decs. & Control, pp. 5318–5322 (2007)
13. Demirli, K., Muthukumaran, P.: Higher Order Fuzzy System identification Using Subtractive Clustering. J. of Intelligent and Fuzzy Systems 9, 129–158 (2000)
14. Demirli, K., Cheng, S.X., Muthukumaran, P.: Subtractive Clustering Based on Modeling of Job Sequencing with Parametric Search. Fuzzy Sets and Systems 137, 235–270 (2003)
15. Ngo, L.T., Pham, B.H.: Approach to Image Segmentation Based on Interval Type-2 Fuzzy Subtractive Clustering. In: Horng, M.-F. (ed.) ACIIDS 2012, Part II. LNCS, vol. 7197, pp. 1–10. Springer, Heidelberg (2012)
16. Liang, Q., Mendel, J.M.: An Introduction to Type-2 TSK Fuzzy Logic Systems. In: IEEE International Conference on Fuzzy Systems, vol. 3, pp. 1534–1539 (1999)

Constrained Layout Optimization in Satellite Cabin Using a Multiagent Genetic Algorithm

Jing Liu

Key Lab of Intelligent Perception and Image Understanding of Ministry of Education of China
Xidian University, 710071, China
neouma@mail.xidian.edu.cn

Abstract. One application of constrained layout optimization problems (CLOPs) is to lay out the instruments in satellite cabin (CLOPs$_{sc}$), which concerns the two dimensional physical placement of a collection of objects within a satellite cabin. CLOPs$_{sc}$ are not only of significant theoretical interest, but also of real practical significance in industrial applications. In this paper, we use a multiagent genetic algorithm with a simple technique of handling constraints to solve CLOPs$_{sc}$. In experiments, the performance of the new algorithm is compared with existing algorithms on three benchmark problems with different complexities. Experimental results show that the new algorithm has a better global searching ability. Especially, it found a currently best solution for the constrained layout problem with 40 objects.

1 Introduction

Layout optimization problems (LOPs) are a kind of NP-hard problems and its objective is to place some objects in a space or divide the space into many sub-spaces and put each object in to one sub-space. Usually, the objects should not overlap with each other and the left space should be minimized. According to the type of spaces, LOPs can be divided into two-dimensional and three-dimensional layout. Some LOPs also have constraints, namely constrained LOPs (CLOPs), which take into account inertia, balance, stabilization, and vibration, etc. CLOPs are more difficult since their feasible search space may be non-convex and non-continuous. One application of CLOPs is to lay out the instruments in satellite cabin (CLOPs$_{sc}$), which concerns the two dimensional physical placement of a collection of objects within a satellite cabin. CLOPs$_{sc}$ are not only of significant theoretical interest, but also of real practical significance in industrial applications.

Many heuristic methods were used to solve CLOPs. Teng et al. [1] proposed the method of model-changing iteration (MCI) and the method of main objects topo-models (MOT). However, the performances of heuristic methods are hardly satisfactory due to the complexity of CLOPs. Thus, intelligent computational methods are widely used. Szykman and Cagan [2] used simulated annealing to solve constrained three-dimensional component layout problems. Tang and Teng [3] proposed a modified genetic algorithm based on decimal coding and adaptive control of parameters.

Qian et al. [4] proposed a human-computer interactive genetic algorithm (HCIGA) by making use of knowledge of human experts. Yu et al. [5] proposed a learning-based genetic algorithm (LGA) by utilizing local analytic information of the functions. Li et al. [6] proposed a particle swarm optimization with mutation operator (M-PSO). Xu et al. [7] proposed a genetic algorithm with the order-based positioning technique.

Evolutionary algorithms (EAs), as a kind of stochastic global optimization methods inspired by the biological mechanisms of evolution and heredity, have been widely used for various practical problems, such as VLSI floorplanning [9], numerical optimization [10], classification [11]. But it is realized from practice that EAs still have weakness [12], and it is worth stepping back and exploring how to best learn from nature and how to incorporate our existing knowledge in artificial intelligence into EAs.

Thus, in our previous work, we combined EAs with multiagent systems to improve EAs' performance, and a series of algorithms have been proposed to solve different practical problems, such as the multiagent GA (MAGA) proposed in [8] for global numerical optimization, which showed excellent performance in complex multimodal functions with as high as 10 000 dimensions; the multiagent EA for constraint satisfaction problems (MAEA-CSPs) proposed in [13]; the multiagent EA for combinatorial optimization problems (MAEA-CmOPs) proposed in [14]. All these algorithms show the huge potential of combining multiagent systems with EAs in solving hard challenging problems.

Therefore, in this paper, with the intrinsic properties of CLOPs$_{sc}$ in mind, we combine MAGA with a technique of handling constraints to solve this problem since the original MAGA is used for global optimization only. In experiments, the performance of the new algorithm is compared with existing algorithms on three benchmark problems with different complexities. Experimental results show that the new algorithm has a better global searching ability. Especially, it found a currently best solution for the constrained layout problem with 40 objects.

The rest of this paper is organized as follows: Section 2 introduces the definition of CLOPs$_{sc}$, and Section 3 describes the multiagent genetic algorithm for CLOPs$_{sc}$. The experimental study is given in Section 4. Finally, conclusions are presented in Section 5.

2 Definition of CLOPs$_{sc}$

In CLOPs$_{sc}$, there is a round clapboard which is vertical to the central axis of the rotating satellite cabin. Then, we need to place the objects, such as various equipments, on the clapboard optimally and satisfying the following conditions: (1) no object protrudes out of the clapboard; (2) no overlapping between any two objects; (3) all objects should assemble around the center of the clapboard as much as possible; (4) the static equilibrium error of the whole system should not exceed the predefined permissible value; (5) the dynamic equilibrium error of the whole system should not exceed the predefined permissible value when the satellite rotates at angular speed ω.

For the sake of simplicity, we assume all n objects are cylinders with uniform thickness and mass distribution and the radius of the clapboard is R. Suppose the Cartesian coordinate system coincides with symmetric axis of the clapboard, and $\boldsymbol{X}_i=(x_i, y_i)$, r_i, and m_i are respectively the centroid, the radius, and the mass of the ith object. Fig. 1 (a) shows the three dimensional case, while Fig. 1 (b) shows the two dimensional case.

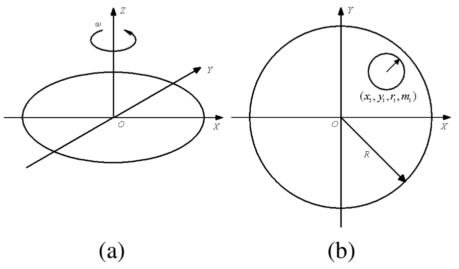


Fig. 1. The view of the layout under the Cartesian coordinate system (a) 3-dimension; (b) 2-dimension

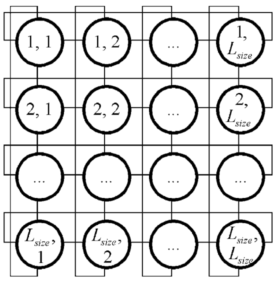


Fig. 2. The model of the agent lattice. Each circle represents an agent, the data in a circle represents its position in the lattice, and two agents can interact with each other if and only if there is a line connecting them.

Throughout this paper, we mainly study the problem of laying out objects on the 2-dimensional clapboard. Thus, the basic problem is to find the position of each object so that all objects can highly concentrate to the center of the clapboard while the constraints are satisfied. The mathematical model of this problem can be formalized as follows.

$$\min F(\boldsymbol{X}) = \min \max_{i\in\{1,\,2,\,\dots,\,n\}} \left\{\sqrt{x_i^2+y_i^2}+r_i\right\}, \quad x_i,\ y_i \in [-R,\ R]$$

s.t.

$$\begin{aligned} f_1(\boldsymbol{X}_i, \boldsymbol{X}_j) &= r_i + r_j - \sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \le 0, \quad i \ne j \\ f_2(\boldsymbol{X}_i) &= \sqrt{x_i^2+y_i^2}+r_i - R \le 0, \quad i \in \{1,\ 2,\ \dots,\ n\} \\ f_3(\boldsymbol{X}_i) &= \sqrt{\left(\sum_{i=1}^n m_i x_i\right)^2+\left(\sum_{i=1}^n m_i y_i\right)^2} - \delta_J \le 0 \end{aligned} \tag{1}$$

where $\boldsymbol{X}=(\boldsymbol{X}_1,\boldsymbol{X}_2,\dots,\boldsymbol{X}_n)$ is the set of objects, and δ_J is a predefined permissible value.

3 Multiagent Genetic Algorithm for CLOPs$_{sc}$

The previous MAGA [8] is proposed for solving global numerical optimization, which can be formulated as solving the following objective function:

$$\min f(\boldsymbol{x}), \quad \boldsymbol{x} = (x_1,\ \dots\ ,x_n) \in \mathcal{S} \tag{2}$$

Where $\mathcal{S} \subseteq \mathcal{R}^n$ defines the search space which is an n-dimensional space bounded by the parametric constraints $\underline{x}_i \le x_i \le \overline{x}_i$, i=1, 2, …, n. In MAGA, an agent represents a candidate solution to the optimization problem, namely a real vector (x_1, x_2, …, x_n), and its energy equals the negative value of objective function. All agents live in a latticelike environment with size of $L_{size}\times L_{size}$ and each agent is fixed on a lattice-point. Since each agent can only sense its local environment, it can only interact with its neighbors. Therefore the agent lattice can be represented as one in Fig. 2.

Since the purpose of the agent is to increases its energy as much as possible, acute competition among agents will be resulted, but this behaviors can only take place between the agent and its neighbors. As a result, the agents with low energy are cleaned out from the agent lattice so that there is more living space for the agents with high energy. Certainly, cooperation behavior may be happen among agents. On the other hand, the agents have intelligence, so they can use the domain knowledge to increase their energies. Based on the viewpoint above, MAGA designs four evolutionary operators for agents, neighborhood competition operator, neighborhood orthogonal crossover operator, mutation operator and self-learning operator, to realize the competition, cooperation, self-learning behaviors among agents. Algorithm 1 describes the MAGA briefly. Please see [8] for more details of MAGA.

Algorithm 1. Multiagent Genetic Algorithm
L^t represents the agent lattice in the tth generation, and $L^{t+1/3}$ and $L^{t+2/3}$ are the mid-lattices between L^t and L^{t+1}. $Best^t$ is the best agent among L^0, L^1, …, L^t, and $CBest^t$ is the best agent in L^t. P_c and P_m are the probabilities to perform the neighborhood orthogonal crossover operator and the mutation operator.

```
Step 1: Initialize $L^0$, update $Best^0$, and $t\leftarrow 0$;
Step 2: Perform the neighborhood competition operator on each agent in $L^t$, obtaining $L^{t+1/3}$;
Step 3: For each agent in $L^{t+1/3}$, if $U(0, 1)<P_c$, perform the neighborhood orthogonal crossover operator on it, obtaining $L^{t+2/3}$;
Step 4: For each agent in $L^{t+2/3}$, if $U(0, 1)<P_m$, perform the mutation operator on it, obtaining $L^{t+1}$;
Step 5: Find $CBest^{t+1}$ in $L^{t+1}$, and then perform the self-learning operator on $CBest^{t+1}$;
Step 6: If $Energy(CBest^{t+1})>Energy(Best^t)$, then $Best^{t+1}\leftarrow CBest^{t+1}$; otherwise $Best^{t+1}\leftarrow Best^t$, $CBest^{t+1}\leftarrow Best^t$;
Step 7: If termination criteria are reached, output $Best^t$ and stop; otherwise $t\leftarrow t+1$, go to Step 2.
```

To solve CLOPs$_{\text{sc}}$, we use the static penalty method to handle constraints so that MAGA can solve problems with constraints. Let penalty coefficients λ_1, λ_2, λ_3>0, then penalty term is defined as follows:

$$\psi(\boldsymbol{X}) = \lambda_1 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \max\left(0, f_1\left(\boldsymbol{X}_i, \boldsymbol{X}_j\right)\right) + \lambda_2 \sum_{i=1}^{n} \max\left(0, f_2\left(\boldsymbol{X}_i\right)\right) + \lambda_3 \sum_{i=1}^{n} \max\left(0, f_3\left(\boldsymbol{X}_i\right)\right) \quad (3)$$

Thus, the CLOP is transformed into an unconstrained LOP,

$$\mathcal{F}(X) = F(X) + \psi(X) \tag{4}$$

To simplify the computing, a polar coordinate system is adopted. That is to say, the position of each object is represented as the polar coordinate of the center of the circle (l_i, θ_i), $l_i \in [0, (R\text{-}r_i)]$, $\theta_i \in [-\pi, \pi]$, i=1, 2, …, n, where $x_i=l_i\cos\theta_i$, $y_i=l_i\sin\theta_i$. Thus, the constraint f_2 can be removed, and the penalty function is changed to

$$\psi'(X) = \lambda_1 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \max\left(0, f_1\left(X_i, X_j\right)\right) + \lambda_3 \sum_{i=1}^{n} \max\left(0, f_3\left(X_i\right)\right) \tag{5}$$

Where λ_1 and λ_3 are set to 1 and 0.1, respectively.

Therefore, (4) is transformed into (6),

$$\mathcal{F}'(X) = F(X) + \psi'(X) \tag{6}$$

Accordingly, the agent used to solve CLOPssc is defined as follows.
Definition 1: An agent, a, represents an element in the search space of CLOPssc, which is denoted as

$$\boldsymbol{a} = \left(l_1, \theta_1, \cdots, l_n, \theta_n\right), \ l_i \in \left[0, R - r_i\right], \ \theta_i \in \left[-\pi, \pi\right], \ 1 \le i \le n \tag{7}$$

Where n is the number of objects. Apparently, the number of dimension of the search space is $2n$. The value of its energy is equal to the reciprocal of the objective value, namely $Energy(\boldsymbol{a}) = 1/\mathcal{F}'(\boldsymbol{a})$.

4 Experiments

Three benchmark problems with different complexities are used to validate the performance of our algorithm. The number of objects of the three problems is respectively 5, 7, and 40. The parameter setting of MAGA is: L_{size}=10, P_o=0.2, P_c=0.1, P_m=0.1, sL_{size}=3, sR=0.2, sP_m=0.05, $sGen$=100. The maximum number of generations is 2000 to 5000 which depends on the problem size. All experiments were executed on a 2.4GHz Pentium IV PC with 1G RAM.

4.1 CLOP with 5 Objects [3]

In order to validate the performance of MAGA, a CLOP with known optima [3] is used. In this problem, 5 objects need to be laid on a big circular container, of which the radius R is 125mm. In the optimum layout, the radius of the out warp circle, the static equilibrium error, and the interference are respectively 120.71(mm), 0(g·mm), and 0(mm). The obtained positions of the 5 objects of the three algorithms (I-GA, M-PSO, and MAGA), are given in Table 1.

Table 2 shows the comparison among the three algorithms. It can be seen that both MAGA and M-PSO found the global optimum while I-GA not. The static equilibrium error of MAGA is the smallest, that is, 0, while that of two other algorithms is larger than 0.002. Moreover, in terms of the computational cost, we can see that MAGA is better than two other algorithms, which only accounts for 59% and 26% of the computational time of M-PSO and I-GA, respectively. The above results show that MAGA has a strong global search ability and a faster convergence speed. Additionally, Fig. 3 shows the geometric layout of the three algorithms.

Table 1. The performance of I-GA, M-PSO, and MAGA for the CLOP with 5 objects

No.	r(mm)	m(g)	I-GA		M-PSO		MAGA	
			x(mm)	y(mm)	x(mm)	y(mm)	x(mm)	y(mm)
1	20.71	20.71	-0.5858	-1.058	0.000	0.000	0.000	0.000
2	50.00	50.00	65.979	26.179	0.000	70.711	-42.022	56.870
3	50.00	50.00	-26.252	68.061	70.711	0.000	42.022	-56.869
4	50.00	50.00	27.891	-66.894	0.000	-70.711	56.870	42.022
5	50.00	50.00	-66.685	-26.886	-70.711	0.000	-56.870	-42.022

4.2 CLOP with 7 Objects [4]

In this problem, the radius R is 50mm, and the number of objects is 7. The acceptable value of the static equilibrium error is δ_J=3.4(g·mm). The obtained positions of the 7 objects of the three algorithms (HCIGA, M-PSO, and MAGA), are given in Table 3.

Table 4 shows the comparison among the three algorithms. It can be seen that the optimal radius of the out warp circle of MAGA is 31.960mm, which is less than those of two other algorithms. The static equilibrium error of MAGA is optimal, namely 0(g·mm), while those of two other algorithm is larger than 0.018 (g·mm). Moreover, in terms of the computational cost, we can see that MAGA is also better than two other algorithms, which only accounts for 65% of the computational time of M-PSO and HCIGA. Therefore, MAGA outperforms the two other algorithms. Additionally, Fig. 4 shows the geometric layout of the three algorithms.

4.3 CLOP with 40 Objects [3]

In this problem, the radius R is 880mm, and the number of objects is 40. The acceptable value of the static equilibrium error is δ_J=20(g·mm). This problem is much more complex than the above two problems since its dimension is 80. Moreover, because the feasible solution space is non-convex and non-continuous, the algorithms can be trapped in local optima easily. The obtained positions of the 40 objects of the two algorithms (M-PSO and MAGA), are given in Table 5.

Table 6 shows the comparison among four algorithms. It can be seen that the optimal radius of the out warp circle of MAGA is 814.188mm, which is less than those of three other algorithms (among which the best value is 843.940(mm)). The smallest static equilibrium error of MAGA is 0.000362 (g·mm), while those of three other algorithm is larger than 0.003 (g·mm). In terms of the computational cost, although MAGA is worse than other algorithms, when the radius of the out warp circle is 870mm, the computation time is 1734s, which is nearly equal to those of HCIGA and M-PSO; when the radius of the out wrap circle is 843mm, the computation time is

2892s, which is slightly larger than that of M-PSO. However, MAGA continues to optimize instead of being trapped into local optima at this point. Therefore, MAGA outperforms the two other algorithms. Additionally, Fig. 5 shows the geometric layout of the MAGA and M-PSO.

Table 2. The Comparison among I-GA, M-PSO, and MAGA

Method	Radius of the out warp circle(mm)	Static equilibrium error (g·mm)	Interference (mm)	Computational time[*] (s)
I-GA	123.200	1.265000	0	648
M-PSO	120.711	0.002712	0	287
MAGA	120.711	0.000000	0	168

[*] All computational time is converted into the time of a PC with 166MHz.

Table 3. The performance of HCIGA, M-PSO, and MAGA for the CLOP with 7 objects

No	r(mm)	m(g)	HCIGA		M-PSO		MAGA	
			x(mm)	y(mm)	x(mm)	y(mm)	x(mm)	y(mm)
1	10.0	100.00	-12.883	17.020	14.367	16.453	6.712	20.905
2	11.0	121.00	8.847	19.773	-18.521	-9.560	20.438	4.647
3	12.0	144.00	20.662	0.000	2.113	-19.730	11.199	-16.495
4	11.5	132.25	-8.379	-19.430	19.874	-4.340	-19.689	-5.072
5	9.5	90.25	-1.743	0.503	-19.271	11.241	0.113	1.972
6	8.5	72.25	12.368	-18.989	-3.940	22.157	-8.637	-21.807
7	10.5	110.25	-21.639	-1.799	-0.946	2.824	-13.962	16.243

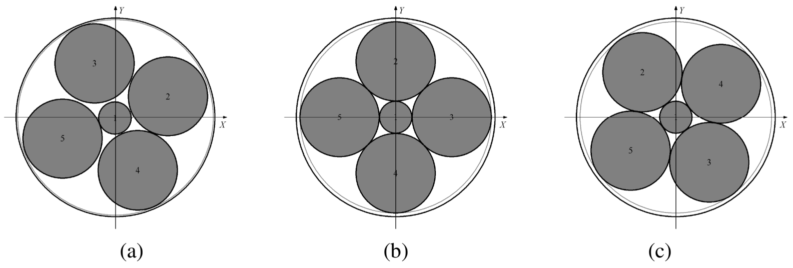
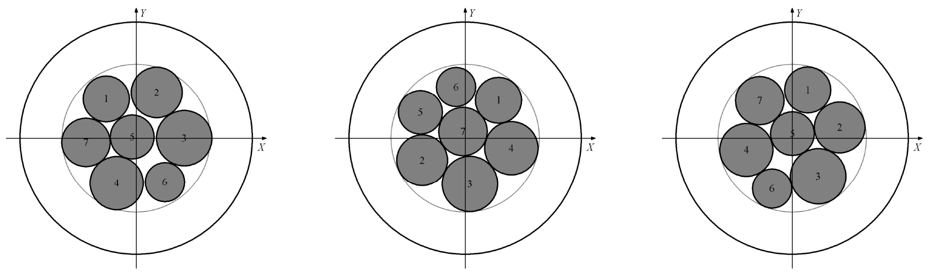


Fig. 3. The geometric layouts found by (a) I-GA, (b)M-PSO, and (c) MAGA

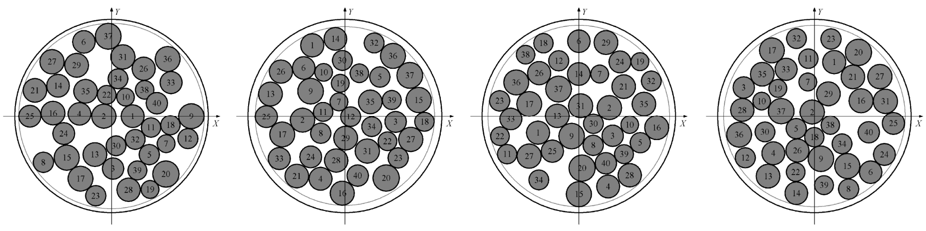
Table 4. The Comparison among HCIGA, M-PSO, and MAGA

Method	Radius of the out warp circle(mm)	Static equilibrium error (g·mm)	Interference (mm)	Computational time (s)
HCIGA	32.662	0.0290	0	1002
M-PSO	31.985	0.0182	0	1002
MAGA	31.960	0.0000	0	649



(a) (b) (c)

Fig. 4. The geometric layouts found by (a) HCIGA, (b)M-PSO, and (c) MAGA



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

Fig. 5. The geometric layouts found by (a) M-PSO, (b) MAGA (814.188mm), (c) MAGA (825.931mm), and (d) MAGA (826.020mm)

Table 5. The performance of M-PSO and MAGA for the CLOP with 40 objects

No	*r*(mm)	*m*(g)	M-PSO		MAGA	
			x(mm)	*y*(mm)	*x*(mm)	*y*(mm)
1	106	11	192.971	0	-296.281	643.224
2	112	12	-69.924	0	-384.475	-36.032
3	98	9	13.034	-478.285	460.235	-52.531
4	105	11	-291.748	21.066	-220.800	-571.721
5	93	8	343.517	-351.055	316.366	357.864
6	103	10	-251.143	674.025	-373.038	439.310
7	82	6	495.268	-252.899	-54.825	132.954
8	93	8	-619.634	-421.032	-218.102	-157.624
9	117	13	725.062	0	-310.213	228.306
10	81	6	127.487	175.174	-192.522	399.134
11	89	7	358.251	-104.181	-197.725	39.030
12	92	8	694.612	-206.946	52.147	-4.667
13	109	11	-151.494	-350.475	-674.766	203.161
14	104	10	-486.096	278.028	-85.712	702.534
15	115	13	-406.944	-378.282	668.424	148.913
16	110	12	-531.396	27.583	-23.678	-702.737
17	114	12	-281.428	-570.129	-567.499	-168.692
18	89	7	535.186	-82.365	721.325	-51.489
19	82	6	349.187	-668.540	-42.881	299.404
20	120	14	494.958	-527.668	374.394	-564.721
21	108	11	-696.916	236.466	-436.950	-546.683
22	86	7	-43.153	196.294	385.347	-229.800
23	93	8	-143.066	-725.316	483.361	-381.701
24	100	10	-433.688	-159.158	-310.756	-372.110

Table 6. (*Continued*)

25	102	10	-741.858	0	-711.783	-4.580
26	106	11	292.820	431.997	-580.061	406.275
27	111	12	-540.511	495.023	597.482	-212.605
28	107	11	154.296	-671.681	-79.774	-404.371
29	109	11	-317.971	463.365	6.855	-201.208
30	91	8	41.295	-271.016	-23.607	507.575
31	111	12	103.622	538.523	203.392	-319.681
32	91	8	215.467	-213.844	265.323	665.016
33	101	10	540.248	306.466	-594.960	-387.716
34	91	8	58.125	341.687	242.692	-96.435
35	108	11	-235.120	227.217	230.711	131.626
36	114	12	510.413	520.918	435.577	548.212
37	118	13	-29.219	725.331	588.346	371.869
38	85	7	300.625	240.313	130.012	390.345
39	87	7	234.066	-494.031	425.499	148.33
40	98	9	411.043	119.080	116.725	-542.517

Table 7. The Comparison among HCIGA, I-GA, M-PSO, and MAGA.

Method	Radius of the out warp circle(mm)	Static equilibrium error (g·mm)	Interference (mm)	Computational time (s)
HCIGA	870.331	0.006000	0	1358
I-GA	874.830	11.39500	0	1656
M-PSO	843.940	0.003895	0	2523
MAGA	814.188	0.000362	0	4409
	825.931	0.000626	0	4208
	826.020	0.000484	0	4120

*When the radius of out warp circle is 870mm, the computational time is 1734s, the static equilibrium error is 0.000939(g·mm), and interference is 0; when the radius of out wrap circle is 843mm, computation time is 2892s, the static equilibrium error is 0.000491(g·mm), and interference is 0.

5 Conclusions

Constrained Layout optimization problems are NP-hard problems. Due to the feasible search space is non-convex and non-continuous, the optimization algorithm can be trapped into local optima easily. Therefore, we combine the multiagent genetic algorithm with a technique of handling constraints to solve them. In experiments, three benchmark problems with different complexities are used to test the performance of MAGA. The results show that MAGA outperforms other existing algorithms. Especially, MAGA obtains the best solution for the problem with 40 objects, namely radius of the out wrap circle is 814.188mm, while the best known one is 843.940mm; static equilibrium error is 0.000362(g·mm), which is nearly an order of magnitude smaller than the best known one, namely 0.003895(g·mm).

What should be noted is that MAGA obtains good performances in solving constrained layout problems by only incorporating a simple version of the static penalty technique. It also illustrates that MAGA has a good potential to solve constrained optimization problems. Therefore, improving the performance of MAGA by developing novel constraints handling techniques is our future work.

Acknowledgment. This work is partially supported by the Fundamental Research Funds for the Central Universities and the National Natural Science Foundation of China under Grants 61271301, 61103119 and 60970067.

References

1. Teng, H., et al.: Layout optimization for the dishes installed on a rotating table. Science in China (Series A) 37(10), 1272–1280 (1994)
2. Szykman, S., Cagan, J.: Constrained three-dimensional component layout using simulated annealing. Journal of Mechanical Design, Transactions of the ASME 119(1), 28–35 (1997)
3. Tang, F., Teng, H.: A modified genetic algorithm and its application to layout optimization. Journal of Software 10(10), 1096–1102 (1999)
4. Qian, Z., Teng, H., Sun, Z.: Human-computer interactive genetic algorithm and its application to constrained layout optimization. Chinese Journal of Computers 24(5), 553–559 (2001)
5. Yu, Y., Cha, J., Tang, X.: Learning based GA and application in packing. Chinese Journal of Computers 24(12), 1242–1249 (2001)
6. Li, N., Liu, F., Sun, D.: A study on the particle swarm optimization with mutation operator constrained layout optimization. Chinese Journal of Computers 27(7), 897–903 (2004)
7. Xu, Y., Xiao, R., Amos, M.: A novel genetic algorithm for the layout optimization problem. In: The Proceeding of IEEE Congress on Evolutionary Computation, Singapore, pp. 3938–3943 (September 2007)
8. Zhong, W., Liu, J., Xue, M., Jiao, L.: A multiagent genetic algorithm for global numerical optimization. IEEE Trans. on System, Man, and Cybernetics—Part B 34(2), 1128–1141 (2004)
9. Liu, J., Zhong, W., Jiao, L., Li, X.: Moving block sequence and organizational evolutionary algorithm for general floorplanning with arbitrarily shaped rectilinear blocks. IEEE Trans. on Evolutionary Computation 12(5), 630–646 (2008)
10. Liu, J., Zhong, W., Jiao, L.: An organizational evolutionary algorithm for numerical optimization. IEEE Trans. on Systems, Man, and Cybernetics, Part B 37(4), 1052–1064 (2007)
11. Jiao, L., Liu, J., Zhong, W.: An organizational coevolutionary algorithm for classification. IEEE Trans. on Evolutionary Computation 10(1), 67–80 (2006)
12. Liu, J., Abbass, H.A., Green, D.G., Zhong, W.: Motif difficulty (MD): a novel predictive measure of problem difficulty for evolutionary algorithms based on network motifs. Evolutionary Computation Journal (MIT) 20(3), 321–347 (2012)
13. Liu, J., Zhong, W., Jiao, L.: A multiagent evolutionary algorithm for constraint satisfaction problems. IEEE Trans. on Systems, Man, and Cybernetics, Part B 36(1), 54–73 (2006)
14. Liu, J., Zhong, W., Jiao, L.: A multiagent evolutionary algorithm for combinatorial optimization problems. IEEE Trans. on Systems, Man, and Cybernetics, Part B 40(1), 229–240 (2010)

A Multi-Objective Approach for Master's Thesis Committees Scheduling Using DMEA

Lam T. Bui and Viet Hoang

Le Quy Don Technical University, Vietnam
lam.bui07@gmail.com, viethv76@yahoo.com

Abstract. In this paper, we propose a multi-objective approach for investigating the Multi-Objective Master's Thesis Committees Scheduling (MMTCS), a practical scheduling problem that arises from our university. For this problem, We need to schedule for a large set of students, each needs an oral defense in front of a committee, given that the time slots, rooms and professors are limited. For it, we first try to derive a mathematical formulation of the problems as a multi-objective problem with a set of hard constraints. We used the satisfaction values of soft constraints as objectives. We adjusted our previous published version of multi-objective evolutionary algorithm to work with this combinatorial problem. We conducted a case study to investigate the problem using our newly multi-objective design. The results showed clearly the efficiency of the multi-objective approach on this problem. The non-dominated solutions showed trade-off between two objectives.

1 Introduction

Multi-objectivity exists in most of the real life problems. This makes it become difficult to find a single solution that meet all objectives. Techniques dealing with these problems often offer a wide range of good solutions trading off on all objectives. Depending on the preference, the user can select the most suitable one among these trade-off solutions. Today, multi-objective techniques have been the popular tools for decision support. In this paper, we consider a special class of scheduling problems called Multi-objective Master's Thesis Committees Scheduling (MMTCS). This is a *practical problem* being handled at our university. For this problem, staff are expected to prepare a committee for every student so that they can conduct the oral defense of their theses. This requires a set of 5 professors, a time slot and a room for each committee. The task is to arrange the committees being optimized on two objectives: *balance of the workload among professors* and *the waiting time for professors* ,and satisfying a large set of constraints such as a supervisor is not in his(or her) student's committee, or two committees sharing a professor cannot be in the same time slot.

In a sense, MMTCS is a timetabling problem with finding an appropriate assignment of time slots and rooms to a set of committees. The difference from timetabling is only at how we model constraints for it since each practical problem has its own constraints. In other sense, MMTCS can be considered as a kind of resource allocation problems since it requires allocating committees with a number of different professors from a limited and predefined set of professors. The problem is first analyzed and mathematically formulated as a multi-objective scheduling problem with a number of hard constraints. We then design a suitable multi-objective approach for investigating it. Our design is based on our previously published multi-objective evolutionary algorithm, called Direction-based Multi-objective Evolutionary Algorithm [5]. Since MMTCS is a combinatorial optimization problem, the adjustment is based on an integer representation. A series of experimental studies are carried out to investigate the dynamics of the problem. The results showed that our designed algorithm is able to find a diverse set of feasible and non-dominated solutions for this problem.

The remainder of the paper is organized as follows: an overview of scheduling and resource allocation is presented in Section 2. The problem context and formulation are given in Section 3 The proposed framework is introduced in Section 4. Experimental studies are presented in Section 5. The last section is devoted to the conclusion of the work and lessons learnt.

2 Scheduling and Resource Allocation

In this section, we summarize the main development of research on Scheduling and resource allocation(SRA) [6], decision-making support process involving assignment and allocation of limited resources to tasks (also known as jobs, operations or activities) over time under certain constraints. It also needs to deal with defining which tasks to be executed at a specific time [13]. We start with the *machine scheduling* [4], a most original scheduling problem. A simple example of shop models can be seen as n jobs with different requirements of CPU time to be scheduled in a single(/or multi)-processor computer. The objective of this problem is to minimize the processing time (called the makespan). The machine scheduling problem is a specialized version of a broader problem called *project scheduling with multiple resources and tasks.* And when there is a limit of resources when scheduling and it is usually referred to as *resource constraint project scheduling problem* - RCPS. In general, this is a NP-hard problem [3].

Further, there are more practical problems that can be transformed in the form of a RCPS such as production sequencing, timetabling, and flight scheduling. For timetabling, the most popular one is the university timetabling problems where the focus is on how to find a good assignment of the time slots to courses (with students pre-enrolled and lecturer pre-assigned) satisfying a set of constraints [7]. The constraints are different from university to university. Hence, the solutions to university timetabling are often ad-hoc. The interested readers are referred to [12,10,14] for more details.

Although scheduling/timetabling inherently possesses a feature of multi- objectivity, the literature is dominated by work considered only single objective

[15,16]. A number of possible objectives for scheduling have been used such as time, cost, resource balancing, robustness, etc. These objectives might conflict with each other to different degrees. Perhaps, the the weighted-sum approach [1] is the common one in dealing with multi-objective RCPS (also for machine scheduling [11]). For this, objectives are summed according to a predefined vector of weights. However, this approach systematically faces several issues, such as objective scaling, and further it is not easy to address the matter of trade-off analysis. Alternatively, Pareto-based approaches are used to obtained a set of trade-off solutions in a single run [16,2]. In [9] Datta et al proposed a bi-objective approach to deal with university timetabling problems using NSGA-II.

3 Master's Thesis Committees Scheduling

3.1 Problem Context

In higher education systems like Vietnam's, all master students must orally defend their thesis in order to complete their study. A defense session is usually conducted with a committee of 05 members including a chair, a committee secretary, and other three members. Note that a student might have one or two supervisors. This requires a large number of professors for the committees, especially when several committees are running in parallel. The task is to find an appropriate assignment of professors for each committee as well as its time slot and the defense room. In the ideal case, there is enough professors for every committee. However, the resource of professors is always scare; also the time is limited. This means a professor can attend several committees. The professors can be internal or external.

From the description above, we can see that our newly defined problem (called MMTCS) is a special class of the conventional university timetabling problem (CUTP). The special features is that professors are required to be scheduled for each committee and the student is fixed (only one); while for CUTP, each class (equivalent a committee in MMTCS) will be predefined a number of students and a lecturer before optimization process. Further, there will be a professor and student constraint for MMTCS , that usually not happens for CUTP. Also, the requirements from Vietnam Ministry of Education and Training makes MMTCS's constraints become more special such as internal/external members that not the case for CUTP.

Also, MMTCS can be considered as a version of RCPS where the tasks are committees, rooms and professors are resources. However, the time for tasks in MMTCS are equal in contrast with the usual style of RCPS where the task durations are different. Further, MMTCS does not have precedent constraints among tasks like RCPS; instead it has many additional constraints on resources, which are strictly required by Vietnam Ministry of Education and Training.

3.2 Multi-Objective Master's Thesis Committees Scheduling Problem - MMTCS

The problem formulation is described as follows:

– **Inputs:**

- A set C of committees (each committee is for a student), a set R of rooms, a set T of time slots, a set P of professors,
- An array *sp* representing the relationship between a student and supervisors

$$sp[s,p] = \begin{cases} 1, \text{student s is supervised by professor p} \\ 0, \text{otherwise.} \end{cases} \tag{1}$$

- An array *pc* representing the potential role of professor as a chair of committees

$$pc[p] = \begin{cases} 1, \text{professor p can be a chair} \\ 0, \text{otherwise.} \end{cases} \tag{2}$$

- An array *pi* representing the status of a professor is internal or external to the university

$$pi[p] = \begin{cases} 1, \text{professor p is an internal member} \\ \quad \text{of the university} \\ 0, \text{otherwise.} \end{cases} \tag{3}$$

- An array *pf* representing the preferred time slots of a professor

$$pf[p,t] = \begin{cases} 1, \text{professor p is available at time slot t} \\ 0, \text{otherwise.} \end{cases} \tag{4}$$

– **Variables:**

- Decision variables $x_{c,r,t,p}$.

$$x_{c,r,t,p} = \begin{cases} 1, \text{committee } c \text{ is scheduled at room } r, \\ \quad \text{time slot } t \text{ and with professor } p; \\ 0, \text{otherwise.} \end{cases} \tag{5}$$

- Auxiliary variables $sh_{t,p}$

$$sh_{t,p} = \begin{cases} 1, \text{professor p is located differently} \\ \quad \text{between } t \text{ and } t+1, \\ 0, \text{otherwise.} \end{cases} \tag{6}$$

- Auxiliary variables $sc_{t,p}$

$$sc_{t,p} = \begin{cases} 0, \text{professor p is scheduled} \\ \quad \text{at both } t \text{ and } t+1, \\ 1, \text{otherwise.} \end{cases} \tag{7}$$

– **Hard Constraints:**

HC1: A room is allocated to only one committee at a time slot and a professor cannot be in more than one committee at a time

$$\sum_{c \in C} x_{c,r,t,p} \leq 1, \forall r \in R, \forall t \in T, and \forall p \in P \tag{8}$$

HC2: Each committee is scheduled once

$$\sum_{t\in T,r\in R} x_{c,r,t,p} = 1, \forall c \in C, and \forall p \in P \tag{9}$$

HC3: Committee members are different

$$\sum_{p\in P} x_{c,r,t,p} = 5, \forall c \in C, \forall t \in T, and \forall r \in R \tag{10}$$

HC4: A professor cannot be a member of his student's committee

$$\sum_{c\in C}\sum_{p\in P} x_{c,r,t,p} \times sp[c,p] = 0, \forall r \in R, and \forall t \in T \tag{11}$$

HC5: A committee must have a chair (selected persons)

$$\sum_{p\in P} x_{c,r,t,p} \times pc[p] = 1, \forall c \in C, \forall r \in R, and \forall t \in T \tag{12}$$

HC6: At least an internal member

$$\sum_{p\in P} x_{c,r,t,p} \times pi[p] \geq 1, \forall c \in C, \forall r \in R, and \forall t \in T \tag{13}$$

HC7: At least 2 external members

$$\sum_{p\in P} x_{c,r,t,p} \times pi[p] \leq 3, \forall c \in C, \forall r \in R, and \forall t \in T \tag{14}$$

HC8: A professor can only serve at his preferred time slots

$$\sum_{p\in P}\sum_{t\in T} [x_{c,r,t,p} - pf[p,t]] = 0, \forall c \in C, and \forall r \in R \tag{15}$$

HC9: All rooms must be used. This is because they are all prepared.

$$\sum_{t\in T}\sum_{c\in C}\sum_{p\in P} x_{c,r,t,p} \geq 1, \forall r \in R \tag{16}$$

– **Objectives:**
  - Minimization of variance of workload between professors - F1

$$F1 = var_{p\in P}(\sum_{r\in R,c\in C,t\in T} x_{c,r,t,p}) \tag{17}$$

    with $var(S)$ is the function calculating the variance between members of set S
  - Minimization of waiting-time for professors - F2

$$F2 = \max_{p\in P}(\sum_{t=1}^{|T|-1} sc_{t,p}) \tag{18}$$

4 Design of an Evolutionary Multi-Objective Scheduling Algorithm

In this section, we describe our adjustment of DMEA for being suitable for MMTCS. Note that DMEA used real-valued representation and for unconstrained problems. Hence, we mainly focus on the solution representation and constraint handling. All other operators of DMEA such as direction-based crossover mutation and selection are kept unchanged. Representation is an important issue to scheduling problems [8]. In our MMTCS problem, it is possible to apply either binary or integer representation. For the earlier, a binary representation is used for a direct mapping to a schedule (decision variables $x_{c,r,t,p}$). For the later, we can apply indirect mappings to $x_{c,r,t,p}$. For it, sometimes we can automatically eliminate constraints. That is why we choose this type of representation for our algorithm.

A complete solution must contain information for both schedule of committees and assignment of professors to committees. Therefore, our design of representation is as follows: $S = (X_{1,1}, X_{1,2}, ..., X_{1,K}, ..., X_{C,1}, X_{C,2} ..., X_{C,K})$. The chromosome is considered as an array with consecutive committees. Each committee component X_i has 7 elements:

- $X_{i,1}$: The room number scheduled for X_i and ranges from 1 to $|R|$
- $X_{i,2}$: The slot number allocated to X_i and ranges from 1 to $|T|$
- $X_{i,3}$: The chair professor (ID) for X_i and this ID ranges from 1 to $|P|$
- $X_{i,4}, X_{i,5}, X_{i,6}, X_{i,7}$: Other members in X_i and these IDs range from 1 to $|P|$

Note that with this type of representation, it always eliminates the hard constraint HC2: C components X_i means all committees are scheduled once.

For handling constraints of DMEA, we use the degree of hard constraints violation as a measure for assessing solutions. A feasible solution is obviously preferred than an infeasible one. For two feasible one, we use dominance relation; while for two infeasible solutions, the one with less violation is better.

5 A Case Study

5.1 Test Scenarios and Settings

For demonstration of the concept, we installed a test scenario based on the real data from the Faculty of Information Technology, Le Quy Don Technical University. There are 4 rooms accommodating all 27 committees. All activities take place within 2 days and each day has 6 time slots; this means we have 12 time slots totally. Since each committee requires 5 members, we need 20 professors at a time if we need 4 committees running in parallel. Total number of professors are 30, so that it is likely that we can have 4 committees running in parallel.

From the problem description, we can see that the chromosome size is obviously 189. We used a population size of 200. The crossover and mutation rates were 0.9 and 0.01 respectively. The evolution was terminated after 120,000 evaluations. Each experiment was repeated for 10 times with different random seeds in the hope of eliminating the stochastic behavior caused by the random generator.

5.2 Results and Discussion

Performance Analysis. The performance of DMEA for MMTCS is illustrated by the obtained solutions, which are the non-dominated solutions after each run. We will exam how they behaved in terms of constraint violation and how diverse they were. First of all, in all 10 runs, DMEA found solutions without any hard-constraint violation after about 600 generations (in particular: 1000, 200, ...). Obviously, once DMEA found feasible solutions, the archive will be filled with feasible non-dominated ones. Further, in all runs, DMEA found quite diverse sets of non-dominated solutions as demonstrated in Figure 1 for some runs. Obviously, this shows that we need to make a trade-off between two objectives; a good solution in saving the free time of professors between their committees (the second objective) might make it difficult to keep the workload balance.

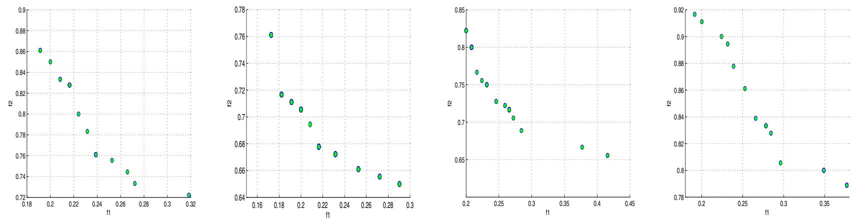


Fig. 1. Snapshots of the obtained non-dominated solutions

A further analysis on the the non-dominated sets in the decision space is given here (see Figure 2). We took two non-dominated solutions: one with the smallest first objective value (0.191) and the second with the smallest second objective value (0.722). We observed a contrast between them. While the first case (with the lowest objective value of the workload balance) obtained the number of committees for each professor from 3 to 6, the second one had a much wider range from 2 to 8 committees. On the other hand, considering the effect of the second objective, we saw that The first solution had only 2 professors with consecutive committees; meanwhile the second solution had 11 professors.

Dynamics. The behaviour analysis of the algorithms not only considers the end state, but it is quite important to understand how they behave during the time of evolution. For this, we recorded all hard-constraint violations over time and

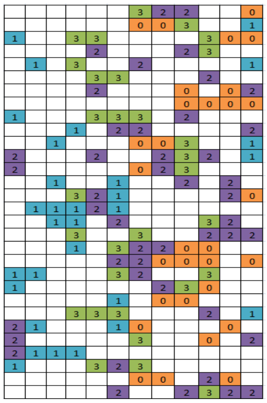
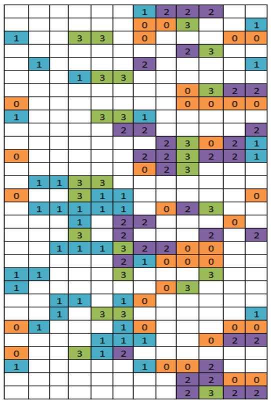



Fig. 2. Demonstration of non-dominated solutions in decision space: 30 professors (rows) and 12 time-slots (columns). The left graph one represented a non-dominated solution with the smallest first objective value (0.191) and the right one with a non-dominated solution having the smallest second objective value (0.722).

reported in Figure 3. For each run, we sum up all hard constraint violation for each solution and the results associated with the best solutions were reported. It is obvious from Figure 3 that our method converged vary fast without being stuck at any time. They are almost violation-free after 600 generations. Its performance among runs was quite stable.

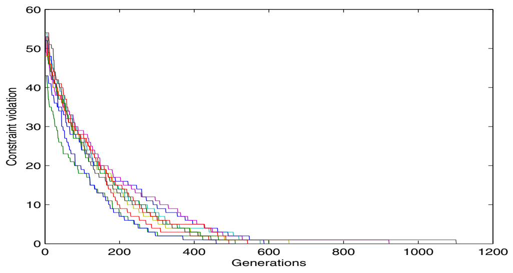


Fig. 3. The total number of hard-constraint violations over time for all 10 runs

Another aspect of dynamics is the number of non-dominated solutions over generations. We recorded it and visualized in Figure 4 for all 10 runs. It is interesting that the number of non-dominated solutions increased quickly to the maximum size (200) after 200 generations. However, it sharply dropped (even to only one solution) before increased again towards the end. This behavior can be explained by looking at the way DMEA handled the hard constraints. It considered the constraint violation as the primary criterion when examining solutions. At the beginning, solutions were infeasible and they were compared by the level they violate constraints. With this single value, normally only one solution was the best. However, as DMEA progressed, it found more solutions with the similar level of constraint violation that is reflected in the first half of the figure. Soon it found a feasible solution, all infeasible will be dropped out of the archive. That is why in the figure, the number of non-dominated solutions dropped sharply after 400 generations.

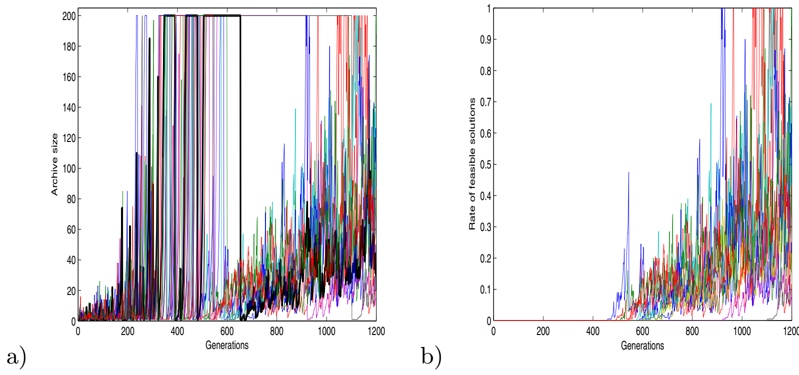


Fig. 4. a)Number of non-dominated solutions. b) The rate of feasible solutions.

Finally, we consider the rate of feasible solutions that DMEA found over time. It clearly shows that our previous finding is appropriate. In the first about 400 generations, most of the runs did not find any infeasible solution. However, after found one, DMEA quickly discover many over time.

6 Conclusion

In this paper, we proposed multi-objective approach to investigate the problem of master thesis defending committee scheduling. For this problem, we need to do both tasks: timetabling the committees and allocating professors to the committees. An evolutionary-based method for problem solving was introduced. We used an integer representation for the solution. A case study was carried out on a test problem adapted from the real data at the Faculty of Information Technology, the Le Quy Don Technical University, Vietnam. We also compared it with a multistart random search algorithm. The obtained results showed that our algorithm found feasible solutions and had a good convergence behavior.

Acknowledgments. We acknowledge the financial support from Vietnam's National Foundation for Science and Technology (Development Grant 102.01-2010.12), the Vietnam Institute for Advanced Study in Mathematics (VIASM), and the Research Fund RFIT of Faculty of Information Technology, Le Quy Don Technical University.

References

1. Abbasi, B., Shadrokh, S., Arkat, J.: Bi-objective resource-constrained project scheduling with robustness and makespan criteria. Applied Math. and Comp. 180, 146–152 (2006)
2. Belfares, L., Klibi, W., Lo, N., Guitouni, A.: Multi-objective tabu search based algorithm for progressive resource allocation. E. J. of Op. Res. 177, 1779–1799 (2007)
3. Blazewicz, J., Lenstra, J.K., Rinnooy Kan, A.H.G.: Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Math. 5, 11–24 (1983)
4. Brucker, P.: Scheduling algorithms. Springer (2007)
5. Bui, L.T., Liu, J., Bender, A., Barlow, M., Wesolkowski, S., Abbass, H.A.: Dmea: a direction-based multiobjective evolutionary algorithm. Memetic Computing 3(4), 271–285 (2011)
6. Bui, L.T., Michalewicz, Z., Parkinson, E., Abello, M.B.: Adaptation in dynamic environments: A case study in mission planning. IEEE Trans. Evolutionary Computation 16(2), 190–209 (2012)
7. Carter, M.: Timetabling. In: Encyclopedia of Operations Research and Management Science, pp. 833–836. Kluwer Academic Publishers (2001)
8. Corne, D., Ross, P., Fang, H.L.: Evolutionary timetabling: Practice, prospects and work in progress. In: UK Planning and Scheduling SIG Workshop (1994)
9. Datta, D., Deb, K., Fonseca, C.M.: Solving class timetabling problem of iit kanpur using multi-objective evolutionary algorithm. Technical report, KanGAL, IIK Kanpur India,Report No. 2006006 (2006)
10. Lewis, R.: A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum 30(1), 167–190 (2008)
11. Nagar, A., Haddock, J., Heragu, S.: Multiple and bi-criteria scheduling: a literature survey. European Journal of Operational Research 81, 88–104 (1995)
12. Petrovic, S., Burke, E.K.: University timetabling. Handbook of Scheduling Algorithms, Models, and Performance Analysis 45, 1–23 (2004)
13. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems, 2nd edn. Springer (2001)
14. Ross, P., Hart, E., Corne, D.: Genetic algorithms and timetabling (2003)
15. Slowinski, R.: Multiobjective project scheduling under multiple-category resource constraints. In: Advances in project Scheduling. Elsevier (1989)
16. Viana, A., de Sousa, J.P.: Using metaheuristics in multiobjective resource constrained project scheduling. European Journal of Operational Research 120, 359–374 (2000)

Coupler-Curve Synthesis of a Planar Four-Bar Mechanism Using NSGA-II

Jaideep Badduri, Rangaprasad Arun Srivatsan, Gurunathan Saravana Kumar, and Sandipan Bandyopadhyay

Department of Engineering Design, Indian Institute of Technology Madras, Chennai - 600036, India
{jaideep.badduri,rarunsrivatsan}@gmail.com,
{gsaravana,sandipan}@iitm.ac.in
http://www.ed.iitm.ac.in

Abstract. This paper applies a genetic algorithm-based optimisation procedure, namely, NSGA-II, to the problem of synthesis of a four-bar mechanism. The internal parameters of `NSGA-II` are tuned using a Design of Experiments (DoE) procedure to enhance the quality of the final results. Constraints are handled through a penalty formulation. Further, a scaling function is introduced, which transforms the penalty terms in a manner that leads to faster convergence of the solutions. The theoretical developments are illustrated via applications to two well-studied problems in the domain of coupler-curve synthesis. A comparison of the results vis-a-vis existing ones shows that the proposed enhancements of the basic scheme of `NSGA-II` deliver promising improvements in terms of accuracy, and rate of convergence of the solutions.

Keywords: Coupler-curve Synthesis, Optimisation, Genetic Algorithms, NSGA-II, Design of Experiments.

1 Introduction

Application of numerical optimisation techniques to the problem of synthesis of mechanisms have gained in popularity in the last few decades. Several methods from the "soft-computing" domain, such as Genetic Algorithms (GA), Differential Evolution (DE), and their variants have been applied successfully to different problems in mechanism synthesis, such as the synthesis of planar four-bar mechanisms for desired coupler curves [1–4].

Probability-based optimisation methods, such as GA and DE, have several *natural* advantages over the classical methods. These methods can handle non-smooth objectives as well as constraints; they are *global* in nature; they use a *population* rather than a single design point (or an *individual* in GA terminology), and hence generally explore the design space better, as well as obviate the need for "good" initial guesses that are crucial for the success of the traditional gradient-based local search methods. However, several inherent drawbacks of these algorithms are also well-known. For instance, the convergence of these algorithms is typically much slower than the local search methods. Attempts to improve the basic algorithms in these aspects have given rise to problem-specific algorithm variants, such as the MUMSA [4], special ways of choosing the initial population [3] etc. While these *off-springs* of the parent algorithms do show improvements in the specific problems to which they are applied by the proposer, these are typically not backed up by enough generic analysis to convince another researcher to adopt them for a different set of problems. One key ingredient in the success of any of these algorithms, or indeed, any probability-based method, is the *proper* choice of the numerous control parameters embedded in the algorithms. There is a possibility that one algorithm out-performs another simply due to the better tuning of these control parameters for some specific problems, rather than any generic superiority of the winning algorithm over the latter.

This paper makes an attempt to investigate this interesting issue of generic superiority of algorithms versus the efficacy of problem-specific tuning of parameters, as well as the impact of refinements in the formulation of the problem itself. A very well-studied problem is chosen as the test-case, namely, that of coupler-curve synthesis, on which a significant number of recent reports are available [1–4]. A well-known and popular GA-based optimiser, namely, the Non-dominated Sorting Genetic Algorithm (`NSGA-II`)[1] [5] is used in this work. The method requires the specification of four probability parameters in addition to the population size and the number of generations. A systematic study of these parameters is done to arrive at their *optimal* values for the chosen problems by means of the *Design of Experiments* (DoE) [6] formalism. Constraint handling is done via a penalty formulation. A non-linear scaling is applied to the values of the constraint function so that the order of magnitude of these values maintain a desired relation with the order of magnitude of the objective function. This, as expected, leads to faster convergence in the problems studied, showing that a better handling of constraints may alleviate the slow convergence problem of `NSGA-II` to some extent.

The rest of the paper is organised in the following manner: in Section 2, the objective and the constraint functions for the optimisation problems are derived using a kinematic model of the mechanism. The constraint handling scheme is described in Section 3. The tuning of the control parameters of `NSGA-II` via a DoE procedure for a specific problem is described in Section 4. The results and conclusions are presented in Sections 5 and 6, respectively.

2 Kinematic Formulation of the Objectives and Constraints

The formulation of the coupler-curve synthesis problem is fairly standard (see, e.g., [4]), as the problem has been studied extensively. Fig. 1 shows the schematic of the mechanism under consideration. The *coupler point* $\boldsymbol{p}_c$ is required to describe a desired curve as the crank, i.e., link 1, runs through a specified interval. From Fig. 1, the coordinates of $\boldsymbol{p}_c$ in the XY-frame are found as:

$$x = O_{1x} + l_1 \cos\theta_1 + x_c \cos\phi_2 - y_c \sin\phi_2 \tag{1}$$

$$y = O_{1y} + l_1 \sin\theta_1 + x_c \sin\phi_2 + y_c \cos\phi_2 \tag{2}$$

[1] Developed at the Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, India; available online for free download at: http://www.iitk.ac.in/kangal/codes.shtml.

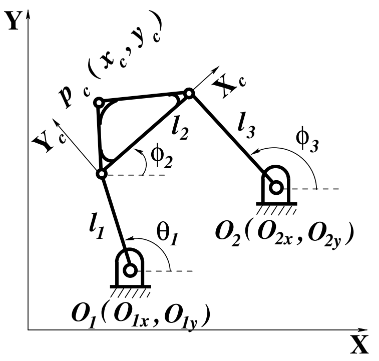


Fig. 1. A planar four-bar mechanism with rotary joints and actuator

where x_c, y_c are the coordinates of $\boldsymbol{p}_c$ in the *coupler frame*, X_cY_c. Most often, it is understood that the *coupler curve*, i.e., the locus of the point $\boldsymbol{p}_c$, cannot take any arbitrary shape[2]. Therefore, typically, a number of *target points* are chosen to represent the desired coupler curve, and the original problem is considered to be solved adequately when the coupler curve generated by the synthesised mechanism passes through these points, $\boldsymbol{p}_{di}(x_{di}, y_{di}),\ i = 1, \ldots, n$. Further, when the points are arrived at certain specified values of the crank angles θ_{1i}, the problem is known as *coupler curve synthesis with coordination*. This version of the problem would be studied in this work.

The *structural error* to be minimised is computed as the the sum of the squares of the Euclidean distances (denoted by d_i) of the actual coupler points generated (denoted by (x_{gi}, y_{gi})) from the respective target points at the n specified crank locations:

$$E = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} (x_{gi} - x_{di})^2 + (y_{gi} - y_{di})^2 \tag{3}$$

One key requirement of the synthesised mechanism is *full-cycle mobility*, i.e., link 1 should be a proper crank, that can rotate through 360° without the mechanism getting locked at *any* configuration. This is possible when the link lengths satisfy the Grashof's conditions (see, e.g., [7]). However, full-cycle mobility also means that the *mechanism does not encounter singularities for any crank angle*. From this perspective, an *equivalent* set of conditions are derived following [8]:

$$g_1 \triangleq \left| \frac{l_0^2 + l_1^2 - (l_2 + l_3)^2}{2 l_0 l_1} \right| - 1 > 0 \tag{4}$$

$$g_2 \triangleq \left| \frac{l_0^2 + l_1^2 - (l_2 - l_3)^2}{2 l_0 l_1} \right| - 1 > 0 \tag{5}$$

$$g_3 \triangleq l_1 + l_2 + l_3 - l_0 > 0 \tag{6}$$

[2] The coupler curve of a four-bar mechanism has a very specific algebraic structure – it is a *tri-circular sextic* curve in x, y [7]. As such, it cannot have any segment of it matching common geometric shapes, e.g., a straight-line segment or an arc of a circle *exactly*.

$$g_4 \overset{\Delta}{=} l_0 + l_2 + l_3 - l_1 > 0 \tag{7}$$

$$g_5 \overset{\Delta}{=} l_0 + l_1 + l_3 - l_2 > 0 \tag{8}$$

$$g_6 \overset{\Delta}{=} l_0 + l_1 + l_2 - l_3 > 0 \tag{9}$$

Using this form of the full-cycle mobility condition instead of the standard form of the Grashof's condition (i.e., $l + s < p + q$, where s, l are the shortest and largest link lengths respectively and p, q are the lengths of the other links) has the following advantage: the violation of the constraints, as and when it happens, is easily quantifiable via the numerical values of the functions g_i, and thus give a physical sense of "distance" from singularities, thanks to the continuous nature of these functions.

3 Development of the Constraint-Handling Scheme

A typical constrained optimisation problem has the form:

$$\begin{aligned} \text{Minimise} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \geq 0, \quad i = 1, 2, ..., n; \\ & h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, ..., m; \\ \text{with} \quad & x_k \in [a_k, b_k], \quad k = 1, 2, ..., p, \end{aligned}$$

where $f(\boldsymbol{x})$ is the objective function, $\boldsymbol{x} = (x_1, x_2, x_3, \cdots, x_p)^T$ is the vector of design variables, $g_i(\boldsymbol{x})$ and $h_j(\boldsymbol{x})$ are the inequality and equality constraints respectively, and a_k and b_k are the upper and lower bounds on x_k, respectively. The proposed approach uses a penalisation strategy in order to convert the constrained optimisation problem to an unconstrained one. Each constraint is imposed through a corresponding penalty term which is added to the objective function; the sum is then used to arrive at the total *fitness value* required by `NSGA-II` to rank the individuals. Each penalty term modifies the objective function in the region where it is violated. If the contribution from this term is *too small* in comparison to the original objective, then it may fail to impose the constraint strictly. On the other hand, if the penalty term is too high in value, the distortion of the original objective function may be so high as to introduce spurious local optima [9]. The inherent drawback in this approach is the difficulty in finding appropriate penalty functions and their numerical weights which offer the best compromise for a given problem [10]. In partial solution to the above problem, we use a transformation of the penalty function which confines its values to the interval [0,1], irrespective of the form of the function. We then use a single weight, α, on the sum of the penalty terms. The heuristics in the process is therefore reduced to the choice of a single parameter α so as to match the order of the nominal penalty terms to that of the unconstrained objective. The details of the scheme are described in the following.

The said transformation of the constraint functions has the following form (adopted from [11]):

$$\psi_\lambda(t) = \frac{t}{\lambda + t}, \ \lambda > 0. \tag{10}$$

Obviously, $\psi_\lambda(t) \in [0,1] \; \forall t \in [0,\infty]$, i.e., for any non-negative real value of t, the function $\psi_\lambda(t)$ is confined to the interval $[0,1]$. Incorporating these details, the steps for the evaluation of the unconstrained objective for any given individual having the design variables $\boldsymbol{x}$ are as follows:

1. Compute the objective function value, $f(\boldsymbol{x})$.
2. Compute the penalty term quantifying the extent of violation of the equality constraints by the sum of the squares of the *scaled* residuals: $h_{eq}(\boldsymbol{x}) = \sum_{j=1}^{m} \psi_\lambda(h_j^2(\boldsymbol{x}))$.
3. Compute the penalty term for the inequality constraints in a similar manner, iff they are violated: $g_{eq}(\boldsymbol{x}) = \sum_{i=1}^{n} P(g_i(\boldsymbol{x}))$, where the function $P(x)$ is defined as:

$$P(t) = \begin{cases} 0 & \text{if } t \geq 0, \\ \psi_\lambda(t^2) & \text{if } t < 0. \end{cases}$$

   The function $P(t)$ acts as a switch, adding the penalty term corresponding to a constraint iff it is violated.
4. Compute the unconstrained objective $F(\boldsymbol{x})$ as a sum of the penalty terms and the original objective:

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + \alpha(g_{eq}(\boldsymbol{x}) + h_{eq}(\boldsymbol{x})) \tag{11}$$

   In Eq. (11), the positive scalar α is used to weigh the penalty terms appropriately against the original objective.

It is expected that via proper tuning of the parameters, λ and α, the effect of constraints can be imparted on the overall objective in a *better* manner. As shown in Section 5.2, this scheme seems to favour faster convergence.

4 Tuning of NSGA-II Parameters Using DoE

NSGA-II is a generic optimiser based on the GA. As in any such probability-based method, convergence of the solution to the optima is affected by several internal parameters, namely, the probability of crossover (p_c), the probability of mutation (p_m), the distribution index for crossover (η_c) and the distribution index for mutation (η_m), in addition to two common parameters: the population size (N_{pop}) and the number of generations (N_{gen}). The mutation which brings about variation in the population have to be *balanced* with respect to cross-over, which preserves building blocks so as to achieve good convergence in this method. The distribution index affects the spread of the offspring solutions, generated by crossover or mutation, in the variable space. The size of the initial population too serves to introduce some variation in the population. Better results are expected with higher number of generations, albeit at the expense of time and computational resources. However, too many generations may not justify the computational cost, if the results have already converged in previous generations.

A systematic study using DoE is performed to select the optimal parameters so as to obtain good convergence of `NSGA-II` for the problem of coupler-curve synthesis. A sample problem, as described in Section 5, is used for this study. The parameters N_{pop} and N_{gen} are held fixed at 200 and 1000 respectively, and the other four parameters, i.e., p_c, p_m, η_c and η_m are systematically varied in appropriate ranges, i.e., $p_c, p_m \in [0, 1]$, $\eta_c \in [0, 20]$, and $\eta_m \in [0, 50]$. In the present DoE, an iterative scheme of two-level full-factorial experiments is done to search the parameter space. The method is similar to the *marching hypercube* method for exhaustive search [12]. The method requires $(2^4 + 1)$ points, of which (2^4) are the corner points of the 4-dimensional hypercube centred on the remaining point. For each of these $(2^4 + 1)$ combinations of parameters, `NSGA-II` is run for the chosen problem and the quality of the result is analysed to identify the best combination. If at a given iteration a better parameter set is not found, the size of the hypercube is reduced and the search is conducted on the set of $(2^4 + 1)$ vertices of the new hypercube. The process continues until the hypercube shrinks to a pre-determined small size, which is chosen to be 10% of the range values of the variables. In this case the initial set of parameter values for the method is chosen as $p_c = 0.5$, $p_m = 0.5$, $\eta_c = 10$ and $\eta_m = 25$, i.e., the mid-range values of the respective parameters, with an initial hypercube size of 40% of the range values (see Table 3).

5 Results and Discussions

Two previously studied problems, namely, Case 3 and Case 4 reported in [4] are revisited in this work, so as to compare the results obtained. The formulation of the primary objective and the design variables are the same, and so are the bounds on these variables: $l_1, l_2, l_3 \in [0, 50]$ and $x_c, y_c, O_{1x}, O_{1y}, O_{2x}, O_{2y} \in [-50, 50]$.

Problem 1: The problem statement is identical with Case 4 in [4]. The six target points and the corresponding crank angles are given in Table 1. The structural error function, E, defined in Eq. (3) is used as the objective function, and full-cycle mobility conditions formulated as inequalities (4-9) are used as constraints. Constraints are handled in two different ways, i.e., with and without the transformations introduced in Section 3, so as to bring out the impact of the transformations on the convergence of the solutions. The results are compared with those reported in [3, 4].

Problem 2: This problem statement is identical with Case 3 in [4]. There is an additional variable in this case: the initial position of the crank, $\theta_1^{(1)} \in [0, 2\pi]$. The 18 points are to be reached at 20° increments of the crank angle, i.e., $\theta_1^{(i)} = \theta_1^{(i-1)} + \pi/9$, $i = 2, \ldots, 18$. The target points are given in Table 2. This problem is studied in the same manner as in Problem 1. The results compared with those reported in [1, 4, 13].

Table 1. Coupler points to be traced for prescribed angles in Problem 1

θ_{1di}	30°	60°	90°	120°	150°	180°
x_{di}	0	1.9098	6.9098	13.09	18.09	20
y_{di}	0	5.8779	9.5106	9.5106	5.8779	0

Table 2. Coupler points to be traced for prescribed angles in Problem 2

$\theta_1^{(i)}$	$\theta_1^{(1)}$	$\theta_1^{(2)}$	$\theta_1^{(3)}$	$\theta_1^{(4)}$	$\theta_1^{(5)}$	$\theta_1^{(6)}$	$\theta_1^{(7)}$	$\theta_1^{(8)}$	$\theta_1^{(9)}$	$\theta_1^{(10)}$	$\theta_1^{(11)}$	$\theta_1^{(12)}$	$\theta_1^{(13)}$	$\theta_1^{(14)}$	$\theta_1^{(15)}$	$\theta_1^{(16)}$	$\theta_1^{(17)}$	$\theta_1^{(18)}$
x_{di}	0.5	0.4	0.3	0.2	0.1	0.05	0.02	0	0	0.03	0.1	0.15	0.2	0.3	0.4	0.5	0.6	0.6
y_{di}	1.1	1.1	1.1	1.0	0.9	0.75	0.6	0.5	0.4	0.3	0.25	0.2	0.3	0.4	0.5	0.7	0.9	1.0

5.1 Optimisation of NSGA-II Parameters Using DoE

Before arriving at the final solutions for the problems described above, optimal values of the NSGA-II parameter set, $[p_c, p_m, \eta_c, \eta_m]$, are determined based on Problem 1 following the method described in Section 4. The starting point for this search is [0.5,0.5, 10, 25], and the vertices of the hypercube in the first iteration defined by $[0.5 \pm 0.2, 0.5 \pm 0.2, 10 \pm 4, 25 \pm 10]$. NSGA-II is used to solve Problem 1 using $(2^4 + 1)$ distinct parameter sets, and the corresponding results are tabulated in Table 3. All these cases are run with the parameter values $\lambda = 10^6$ and $\alpha = 10^{14}$.

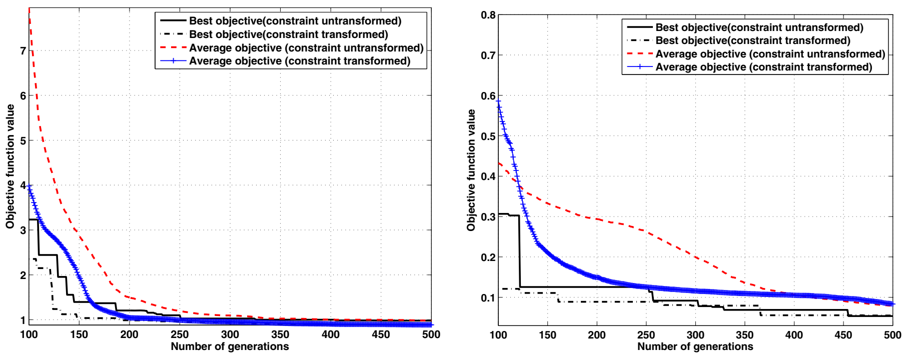
It can be seen that the parameter set #17: [0.3,0.3, 6, 15] results in the best solution. The next iteration takes this as the centre point and defines a hypercube around this to

Table 3. First set of $(2^4 + 1)$ experiments using NSGA-II on Problem 1

	Parameters $[p_c, p_m, \eta_c, \eta_m]$	$E = \sum_{i=1}^{n} d_i^2$ (See Eq. (3))	$\max(d_i)$ (See Eq. (3))
1	[0.5,0.5,10,25]	1.26	0.69
2	[0.7,0.7,14,35]	2.08	0.77
3	[0.7,0.7,14,15]	1.83	0.84
4	[0.7,0.7,6,35]	3.41	0.83
5	[0.7,0.3,14,35]	2.36	1.14
6	[0.7,0.3,6,35]	5.08	1.34
7	[0.7,0.3,14,15]	1.23	0.72
8	[0.7,0.7,6,15]	2.28	0.73
8	[0.7,0.3,6,15]	139.72	6.26
10	[0.3,0.7,14,35]	9.14	1.71
11	[0.3,0.7,14,15]	9.79	1.74
12	[0.3,0.7,6,35]	26.63	2.72
13	[0.3,0.3,14,35]	19.41	2.72
14	[0.3,0.3,6,35]	23.73	2.34
15	[0.3,0.3,14,15]	4.31	1.31
16	[0.3,0.7,6,15]	2.09	0.82
17	**[0.3,0.3,6,15]**	**0.89**	**0.53**

Table 4. Optimised mechanism for Problem 1 with and without constraint transformation

	l_1	l_2	l_3	x_c	y_c	O_{1x}	O_{1y}	O_{2x}	O_{2y}	E
Constraint untransformed	8.92	25.40	42.65	45.53	-24.74	17.25	-50.00	43.35	-16.30	0.89
Constraint transformed	8.85	22.45	35.62	49.95	45.82	16.65	-48.42	-19.84	-25.53	**0.80**



(a) Problem 1 (b) Problem 2

Fig. 2. Effect of constraint transformation on the convergence of NSGA-II

search for better parameters. However, the search converges to the parameter set $\#17$ as it turns out to be better than the vertices of the new hypercube. The rest of this paper uses the same set of parameters for solving Problems 1 and 2.

5.2 Improvement of Convergence of NSGA-II Due to Transformation of Constraint Functions

The convergence of the average as well as the best objective values to the respective optimum values for Problem 1,2 are shown in Fig. 2(a) and Fig. 2(b), respectively. It can be seen that for both the problems constraint handling improves the convergence rate of the best as well as average objective value. In particular, in Problem 1, the acceleration in convergence due to the constraint modification is very much pronounced – the required number of generations being practically halved in this case. The converged results for Problem 1 are tabulated in Table 4 . The optimised mechanism obtained by NSGA-II with constraint handling function shows better performance, i.e., lower value of the structural error, E.

5.3 Comparison of Results

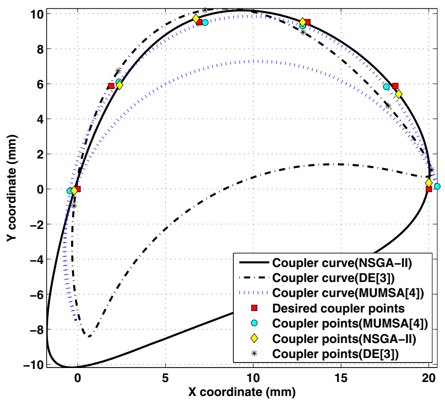
For Problem 1 the results are compared with those obtained by a problem-specific modification of GA (called MUMSA) in [4], and using DE in [3]. In order to ensure fair comparison of the converged results with respect to the computational effort involved, the number of function evaluations is kept the same as those reported in the corresponding literature. Table 5 shows the optimised variable values, as well as two measures of performance: the structural error E and the largest deviation at any point $\max(d_i)$. It is evident that the results obtained in this work are better than those in [4], and far superior to those in [3]. Table 6 presents the corresponding results for Problem 2, and as can be seen they are comparable to those reported in [1, 4, 13].

Table 5. Comparison of the results for Problem 1

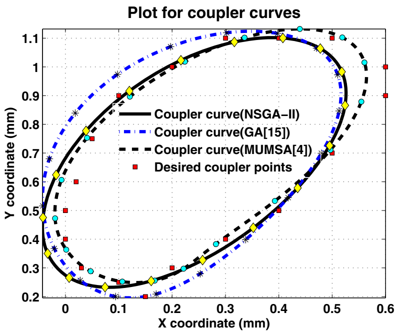
	l_1	l_2	l_3	x_c	y_c	O_{1x}	O_{1y}	O_{2x}	O_{2y}	$E = \sum_{i=1}^{6} d_i^2$	$\max(d_i)$
NSGA-II	8.85	22.45	35.62	49.95	5.82	16.65	-48.42	-19.84	-25.53	**0.80**	**0.48**
MUMSA [4]	1.35	1.35	49.99	11.38	4.44	10.19	-3.69	60.05	-7.06	1.21	0.52
DE [3]	5.00	5.91	50.00	18.81	0.00	14.37	-12.44	59.10	9.92	5.49	1.21

Table 6. Comparison of the results for Problem 2

	l_1	l_2	l_3	x_c	y_c	O_{1x}	O_{1y}	O_{2x}	O_{2y}	$\theta_1^{(1)}$(in rad)	$E = \sum_{i=1}^{18} d_i^2$	$\max(d_i)$
NSGA-II	0.46	5.47	7.57	2.61	1.20	1.03	3.42	4.02	-4.51	1.03	**0.048**	**0.09**
MUMSA [4]	0.30	3.91	0.85	-2.07	1.66	-1.31	2.81	-5.41	4.55	7.59	0.019	0.08
GA [13]	0.27	1.18	2.14	-0.83	-0.38	1.13	0.66	0.47	-1.10	6.91	0.064	0.11
GA [1]	0.24	4.83	2.06	0.77	1.85	1.78	-0.64	3.42	1.93	1.23	0.034	0.10



(a) Problem 1



(b) Problem 2

Fig. 3. Comparison of coupler curves generated with existing ones

6 Conclusion

The paper introduces certain improvements in the application of a GA-based optimiser to the problem of synthesising mechanisms. A new scheme for better handling the constraints through a transformation of the penalty terms is introduced. Internal control parameters of the chosen optimiser, NSGA-II, are tuned for this type of problems by means of a DoE procedure. Results obtained show promising improvements in terms of the quality of solutions obtained, as well as the rate of convergence, even when the scheme is applied to very well-studied problems in the field.

References

1. Cabrera, J., Simon, A., Prado, M.: Optimal synthesis of mechanisms with genetic algorithms. Mechanism and Machine Theory 37(10), 1165–1177 (2002)
2. Laribi, M., Mlika, A., Romdhane, L., Zeghloul, S.: A combined genetic algorithm – fuzzy logic method (GA – FL) in mechanisms synthesis. Mechanism and Machine Theory 39(7), 717–735 (2004)
3. Acharyya, S., Mandal, M.: Performance of EAs for four-bar linkage synthesis. Mechanism and Machine Theory 44(9), 1784–1794 (2009)
4. Cabrera, J., Ortiz, A., Nadal, F., Castillo, J.: An evolutionary algorithm for path synthesis of mechanisms. Mechanism and Machine Theory 46(2), 127–141 (2011)
5. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
6. Montgomery, D.C.: Design and Analysis of Experiments. John Wiley & Sons (2006)
7. Mallik, A., Ghosh, A., Dittrich, G.: Kinematic Analysis and Synthesis of Mechanisms. CRC Press, Boca Raton (1994)
8. Ghosal, A.: Robotics: Fundamental Concepts and Analysis. Oxford University Press, New Delhi (2006)
9. Sarker, R., Mohammadian, M., Yao, X., Runarsson, T., Yao, X.: Constrained evolutionary optimization - the penalty function approach. In: Evolutionary Optimization. International Series in Operations Research & Management Science, vol. 48, pp. 87–113. Springer US (2003)
10. Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186(2-4), 311–338 (2000)
11. Rimon, E., Koditschek, D.E.: Exact robot navigation using artificial potential functions. IEEE Transactions on Robotics and Automation 8, 501–518 (1992)
12. Lin, Y., Mistree, F., Allen, J.K., Tsui, K.-L., Chen, V.C.P.: A sequential exploratory experimental design method: Development of appropriate empirical models in design, vol. 1, pp. 1021–1035 (2004)
13. Kunjur, A., Krishnamurty, S.: Genetic algorithms in mechanism synthesis. Journal of Applied Mechanisms and Robotics 4(2), 18–24 (1997)

A Simulation Model for Optimise the Fire Evacuation Configuration in the Metro Supermarket of Hanoi

Manh Hung Nguyen[1,2], Tuong Vinh Ho[1], and Jean-Daniel Zucker[1,3]

[1] IFI, Equipe MSI; IRD, UMI 209 UMMISCO, Institut de la Francophonie pour l'Informatique, Hanoi, Vietnam
[2] Posts and Telecommunication Institute of Technology (PTIT), Hanoi, Vietnam
[3] UPMC Univ Paris 06, UMI 209, UMMISCO, F-75005, Paris, France
`nmhufng@yahoo.com`, `ho.tuong.vinh@auf.org`, `jean-daniel.zucker@ird.fr`

Abstract. When constructing a new public building such as a supermarket, an important question raises is how to optimise the fire evacuate configuration in the building to reduce the lost when fire occurs. This paper presents a model and implement it as a tool to optimise the evacuation configuration in a buildings, in particular, the evacuation signs plan of the Metro supermarket of Hanoi. The evacuation signs setup is a critical and several changes from the existing one could, according to simulation, reduce casualties in case of fire emergency.

Keywords: Multiagent system, fire evacuation, agent behavior, optimisation, simulation.

1 Introduction

Recently, there has been many researches in simulating of human behaviors in the fire evacuation of buildings. For instance, the models of Lin et al. [5] presented an agent-based simulation model developed for a 2-story office building. Kuligowski and colleagues [4] argued for the inclusion of a comprehensive behavioral conceptual model in computer evacuation models.Wang et al. [8] proposed an approach using the extended hierarchical node-relation model (EHI-NRM) to represent a building's internal structure. In the work of Weifeng and Hai [9], a numerical model based on cellular automaton was proposed to simulate the human behavior termed "flow with the stream" in emergency evacuation from a large smoke-filled compartment. Luo et al. [6] presented their work on designing behavior model for virtual humans in a crowd simulation under normal-life and emergency situations.

In our previous work [7], a model for fire evacuation simulation has been proposed for optimise the evacuee behaviors during evacuation, in a realistic environment with GIS data of the Metro supermarket of Hanoi. We now continue with this work to develop a model and implement it as a tool to optimise the evacuation configuration in a buildings, in particular, the evacuation signs plan, the position of shelves, and the position of emergency exits of the Metro supermarket of Hanoi.

This paper is organised as follows: Section 2 presents the design of this simulation. Section 3 presents the results of simulations on evacuation configurations to improve the fire evacuation. Finally Section 4 presents a brief conclusion and a discussion about future research.

2 Simulation Modelling

2.1 Agents

In this model, we model four kinds of entities: occupant, fire, alarm, evacuation signs and environment.

Human agents. They have two main properties: (i) *observable-zone:* the space around an agent that can be observed and perceived by the agent; and (ii) *power:* the health of agent. This is initially based on its health, and then reduced due to the effect of fire/smoke. The human agents have also some behaviors: (i) *target-move:* move to a specific target; and (ii) *perceive:* observe other agents or objects in his observable-zone. This includes hearing of fire alarm's ring.

Fire agents. They have three main properties: (i) *duration:* its duration; (ii) *propagation-speed:* the speed of propagation of the fire; and (iii) *affected-zone:* the space around a fire which can affect people inside it. Their main behaviors: (i) *kill:* affect people in the *affected-zone* until they died; and (ii) *propagate:* propagate to a new position with his own *propagation-speed*.

Alarm agents. They have two main properties: (i) *affected-zone:* people in this zone can hear this ringing; and (ii) *detected-zone:* this rings if there is fire or smoke appearing in this zone. Their main behavior is *ringing:* it occurs when there is fire/smoke appearing in its *detected-zone*. This affects people in *affected-zone*.

Sign agents. They have two main properties: (i) *position:* its position in the building; and (ii) *direction:* it points out the direction to one of emergency exits.

2.2 Evacuee Movement Principle

The fire evacuation experts of Hanoi Fire Evacuation Association have suggested us to respect the fire evacuation guidelines when the evacuee meets obstacles: the evacuee should move along the border of the obstacle until the door (target) or there no more obstacle in front of the evacuee. We take into account this principle of evacuee movement. Note that most of existing simulation models did not respect these guidelines.

We use approach *priority direction* for agent evacuating. Therefore, agent chooses a direction to move following the one with the highest priority. Other directions will then be prioritised in relative with the highest priority direction. There are two movement strategies: *4 directions* or *8 directions* (as depicted in Fig.1). We use the *8 directions* strategy in all simulations. The more the direction is near the highest priority direction, the more the direction is high priority. At each step, the agent considers the highest priority direction to move. If it is not possible, the agent will consider the next lower priority direction, and so on. A candidate direction is not possible if only if: either it

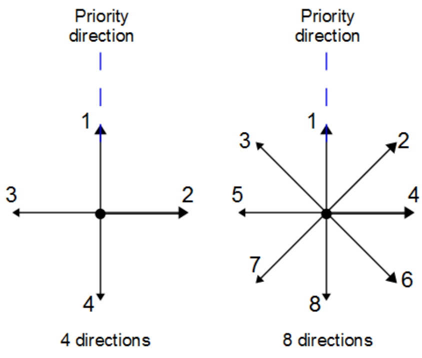


Fig. 1. The movement strategy based on 4 and 8 directions

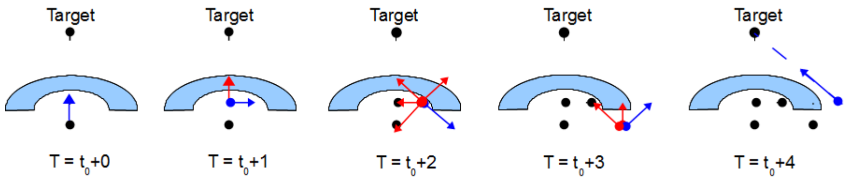


Fig. 2. The movement principle to avoid obstacles

leads to an obstacle, or it leads to a position which is in the *recent passed positions list* of the agent.

In order to avoid the infinite loop of agent movement in the case having obstacles on the agent direction, we use a *recent passed positions list* which contains n last positions of agent. Agent thus consider the next position to move which is not in its recent passed positions list. We do not save all the passed positions of agent because of dynamic environment: some kind of obstacles, such as fire, can dynamically change. There may be a fire at the position x at the moment t_1, but may be no more fire at x at the moment $t_2 \neq t_1$. So we limit the size of the list in order to give to agent some change to return to the positions which it passed in long time. In our simulations, we use the size of the list is 20 to balance the two objectives: avoid infinite loop and adapt to the dynamic environment.

Agent determines its own *recent movement tendency* by consider m last positions ($m < n$, n is the size of *recent passed positions list*). Therefore, the priority direction is the arc from the $m-$latest position to the current position of agent. Fig.2 illustrates the movement principle of an agent when there is an obstacle on its evacuation way. At the time $t = t_0$, the agent does not meet the obstacle yet, so it continues to move to its target. Next step, $t = t_0 + 1$, the agent meets the obstacle, it finds its recent movement tendency which is still direct to target because the two latest positions are on the same line ($m = 2$). But the 1^{st}, 2^{nd} and 3^{rd} directions are impossible (in the obstacle, case of *8 directions*), so the 4^{th} and 5^{th} direction are possible. Assume that the agent turns right. At the time $t = t_0 + 2$, the priority direction is the arc from the position at $t = t_0$

to the one at $t = t_0 + 2$. In this direction, the first 4 priorities are not possible, the 5^{th} is possible (the blue arc), and so on. At the time $t = t_0 + 4$, there no more obstacle in front of agent, so its priority direction is the line direct to its target.

2.3 Evacuee Power Reduce Principle

The evacuee's power is decreased in two cases. First, if the evacuee is in the fire or near by the fire, his/her power is reduced. The closer the fire is near by him, the fast his power is decreased. Assume that p_i^t and p_i^{t+1} are the powers of evacuee i at the time t and $t + 1$, we have:

$$p_i^{t+1} = p_i^t - \alpha . \frac{N_i^t}{D_i^t + 1} \tag{1}$$

where α is the influence factor of fire; N_i^t is the number of fires which have affected on the evacuee i at time t; D_i^t is the average distance from evacuee i to the fire which have affected on i. We add 1 in the denominator to avoid the case of division by zero.

Second, if the evacuee runs to escape, his/her power is reduced. The more he moves (and the faster he moves), the more he loses his power. Assume that p_i^t and p_i^{t+1} are the power of the evacuee i at the time t and $t + 1$, we have:

$$p_i^{t+1} = p_i^t - \beta . V_i^t . M_i^t \tag{2}$$

where β is the influence factor of run (this is much more smaller than α); M_i^t is the number of other people who touches evacuee i at time t; V_i^t is the speed of evacuee i.

2.4 Simulation Parameters

The random aspect in initial values of simulation parameters could influence on the results of simulation. By consequence, the results could not be comparable. For instance, the difference of the start ring time of fire alarm causes the difference of number (and rate) of survivals in a fire evacuation simulation.

In order to make the results comparable, and in order to avoid the effect of random aspect in initial value of simulation parameters, we use the same values for input parameters of all simulations: steps of simulations; number of people; initial power, initial distribution, speed and observable range of people; number, initial position and affected zone of fire; start ring time of alarm. These values are estimated from the Europe guideline on *Fire safety engineering concerning evacuation from buildings* (CFPA-E No.19:2009 [2]), and from the *Human factors: Life safety strategies Occupant evacuation, behaviour and conditions* (PD7974-6:2004 [3]). These parameters are presented in Table 1.

2.5 Analysis and Evaluation Criteria

For each simulation scenario, we will run many times (100 times at least). At the output, we need to calculate the following parameters:

Table 1. Simulation parameters

Parameter	**Values**
Number of simulation steps	300
Number of people	1000
Initial power	100(%)
Influence factor of fire (α)	0.05
Influence factor of run (β)	0.002
Number of fire	1

- percentage of survivals/death. A person is considered as dead when his/her power is reduced to zero or if s/he is still blocked inside at the end of simulation.
- average time for a person to be escaped. It is the time duration from the moment when s/he starts to evacuate until s/he escapes.
- average of percentage of remaining power of survivals. As the initial power of people is randomised, so this parameter calculates the percentage of remain power after escaped regarding the initial power.

An evacuation behavior is considered as better if: (1) the % survivals is higher (% of death is lower); (2) the average time to escape is shorter; and (3) the average of % remaining power of survivals is higher.

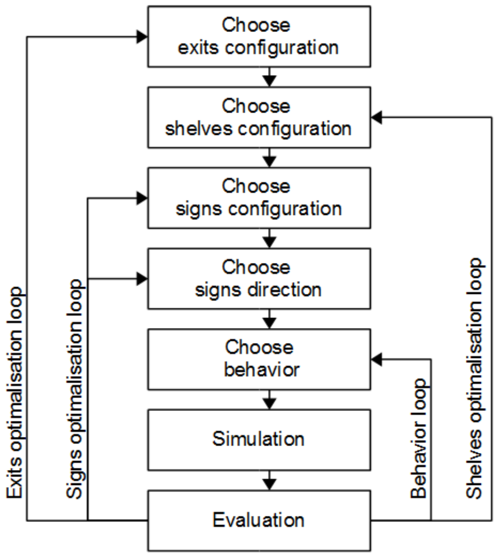

Choose exits configuration
Choose shelves configuration
Choose signs configuration
Choose signs direction
Choose behavior
Simulation
Evaluation
Exits optimalisation loop
Signs optimalisation loop
Behavior loop
Shelves optimalisation loop

Fig. 3. The detail optimal loops in the simulations

2.6 Simulation Plan

Our simulation of fire evacuation is implemented in the simulation platform GAMA [1]. The simulations will be taken step-by-step as depicted in Fig.3, from bottom to up. The first optimisation is optimal behaviors loop, we will run simulations with three behaviors of evacuee: following the own path, following the crowd, and following the signs. This is done in the previous work (Nguyen et al. [7]) and the results indicated that evacuees should follow the evacuation signs. This best behavior will be applied for all next optimal simulations.

The second optimisation is optimal fire evacuation configuration of the supermarket. This consists of three sub-optimisations: optimal signs loop (including signs direction and signs configuration), optimal shelves configuration loop, and exits configuration loop.

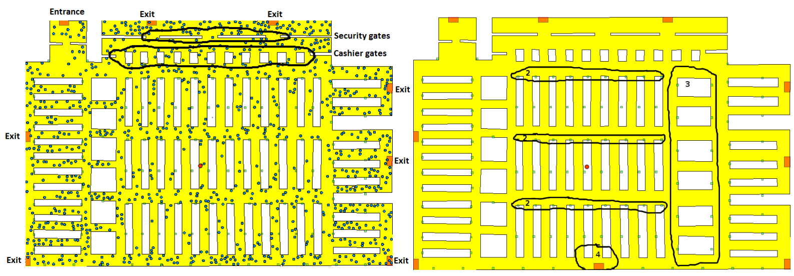
Our objective is to minimise the change of the current configuration of signs, shelves and exits. So, our proposals are limited in changing the current situation of the supermarket.

3 Optimisation of Evacuation Configuration

3.1 Optimisation 1: Change of Signs Direction

Proposal of Change. In the simulations with the actual signs direction, we found that there are too many of people who exit through Exit 2 and 3. This leads them blocked there for long time before escaped.

We propose, in the first optimisation, to change the signs directions as following: the new direction indicated on a sign is the one to the nearest exit of the supermarket.



(a) The original plan (b) The proposals of changes

Fig. 4. plan of the Metro supermarket of Hanoi

Results. The % of survivals (Fig.5.a) in case of changing the signs direction is significantly higher than in case of actual sign direction ($M(change\ direction) = 95.96\%$, $M(real\ signs) = 94.30\%$, significant difference with $p - value < 0.05$).

Time to escape. The average time to escape (Fig.5.b) in case of changing the signs direction is significantly shorter than in case of actual sign direction ($M(change\ direct-$

$ion) = 96.52$, $M(real\ signs) = 110.52$, significant difference with $p - value < 0.001$).

Remaining power. The average of remaining power of survivals (Fig.5.c) in case of changing the signs direction is significantly higher than in case of actual sign direction ($M(change\ direction) = 81.25\%$, $M(real\ signs) = 78.00\%$, significant difference with $p - value < 0.001$).

3.2 Optimisation 2: Change of Signs Position

Proposal of Change. There could be some limitations of current signs position of the supermarket: the signs are putted on the middle of shelves. This limit the observed range of the signs from evacuees.

We propose to change the signs position as following: (1) put the signs in a constant distance, start with each top of shelves; and (2) add some more signs in the zone of vegetables and foods (Fig.4.b, parts number 2).

Results. The % of survivals (Fig.5.a) in case of changing the signs configuration is significantly higher than in the case changing the signs direction ($M(change\ signs) = 98.11\%$, $M(change\ direction) = 95.96\%$, significant difference with $p - value < 0.001$).

Time to escape. The average time to escape (Fig.5.b) in case of changing the signs configuration is significantly shorter than in the case changing the signs direction ($M(c-\ hange\ signs) = 91.10$, $M(change\ direction) = 96.52$, significant difference with $p - value < 0.001$).

Remaining power. The average of remaining power of survivals (Fig.5.c) in case of changing the signs configuration is significantly higher than in the case changing the signs direction ($M(change\ signs) = 83.37\%$, $M(change\ direction) = 81.25\%$, significant difference with $p - value < 0.001$).

3.3 Improvement 3: Change of Shelves Configuration

Proposal of Change. In the simulations of optimisation 2, we discover that there exists some group of people who are blocked in between the rightest shelves of the central zone. They have some difficulties to escape to the right exits because of these shelves configuration. While the leftest side of the zone is good evacuation thank to the low squares configured there.

We propose to change the shelves configuration as following: on the right side of the central zone, change the three last shelves to the low squares as those in the left side of the zone (Fig.4.b, part number 3).

Results. The % of survivals (Fig.5.a) in case of changing the shelves configuration is significantly higher than in the case changing the signs position ($M(change\ shelves) = 99.43\%$, $M(change\ signs) = 98.11\%$, significant difference with $p - value < 0.001$).

Time to escape. The average time to escape (Fig.5.b) in case of changing the shelves configuration is significantly shorter than in the case changing the signs position ($M(c\text{–}hange\ shelves) = 86.49$, $M(change\ signs) = 91.10$, significant difference with $p - value < 0.001$).

Remaining power. The average of remaining power of survivals (Fig.5.c) in case of changing the shelves configuration is significantly higher than in the case changing the signs position ($M(change\ shelves) = 85.01\%$, $M(change\ signs) = 83.37\%$, significant difference with $p - value < 0.001$).

3.4 Improvement 4: Change of Exits Configuration

Proposal of Change. For the last optimisation, we propose to change the exits configuration as following: create one more emergency exit at the back of the supermarket (Fig.4.b, part number 4). This is possible to do because at there is a private road behind the supermarket which is reserved for supply chains.

Results. The % of survivals (Fig.5.a) in case of changing the exits configuration is significantly higher than in the case changing the shelves configuration ($M(change\ exits) = 99.95\%$, $M(change\ shelves) = 99.43\%$, significant difference with $p - value < 0.001$).

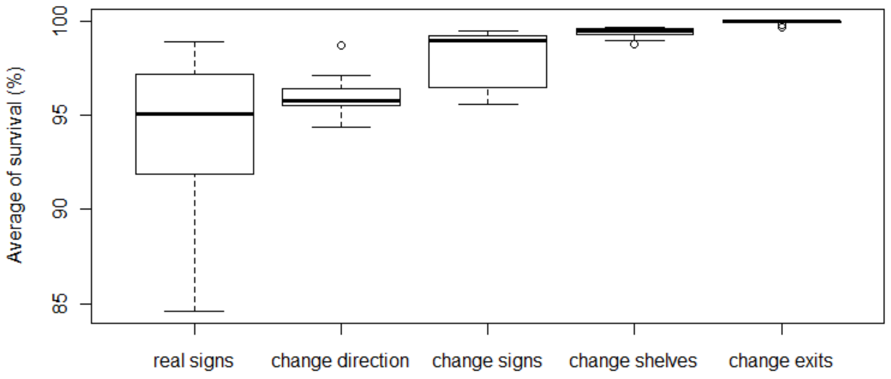
Time to escape. The average time to escape (Fig.5.b) in case of changing the exits configuration is significantly shorter than in the case changing the shelves configuration ($M(change\ exits) = 75.42$, $M(change\ shelves) = 86.49$, significant difference with $p - value < 0.001$).

Remaining power. The average of remaining power of survivals (Fig.5.c) in case of changing the exits configuration is significantly higher than in the case changing the shelves configuration ($M(change\ exits) = 87.00\%$, $M(change\ shelves) = 85.01\%$, significant difference with $p - value < 0.001$).

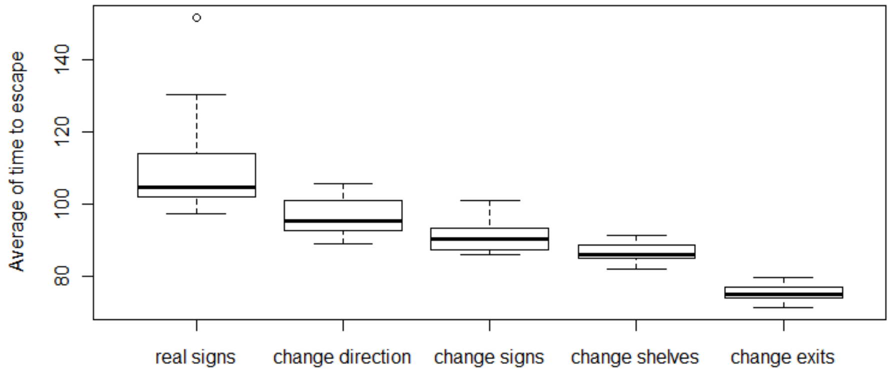
Table 2. Summary of evacuation configuration optimisation

Improvements	Rate of survivals (%)	Time to escape	Remain power (%)
Current signs	94.30	110.52	78.00
Change direction	95.96	96.52	81.25
Change signs	98.11	91.10	83.37
Change shelves	99.43	86.49	85.01
Change exits	99.95	75.42	87.00

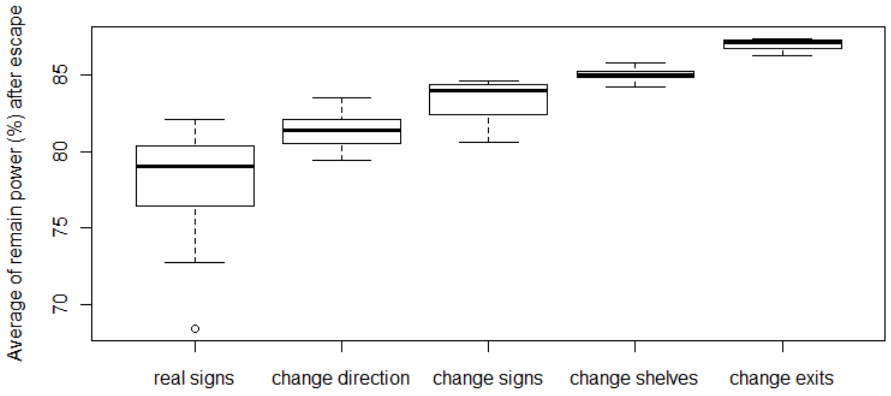
Table 2 summaries the results of five solutions proposed for improvement and/or optimization of evacuation plan. The results suggest that in case of a fire evacuation of the Metro supermarket of Hanoi, the change should follow the following order: change of exits configuration, change of shelves configuration, change of signs configuration, and change of signs directions. And if all these changes were made, we will have a much better evacuation plan for the Metro supermarket of Hanoi.



(a) The % of survivals at the simulation step = 300



(b) Average time to escape



(c) Average remaining power of survivals after escaping

Fig. 5. Results of optimisation

4 Conclusion and Future Works

This paper presented simulations to minimise casualties by modifying the evacuation plan of a realistic supermarket in Hanoi. Metro supermarket plan is represented in GIS data. The simulations were used to study three cases: change in signs directions and configurations, changes in shelves configuration, and change in exits configuration. The simulation results suggest that these changes do improve significantly the current fire evacuate configuration of the supermarket.

The problem of optimisation of items store configuration in order to better separate people in crowd are some of our future research directions.

References

1. Amouroux, E., Chu, T.-Q., Boucher, A., Drogoul, A.: GAMA: An Environment for Implementing and Running Spatially Explicit Multi-agent Simulations. In: Ghose, A., Governatori, G., Sadananda, R. (eds.) PRIMA 2007. LNCS (LNAI), vol. 5044, pp. 359–371. Springer, Heidelberg (2009)
2. Europe Guideline. Fire safety engineering concerning evacuation from buildings. Technical report, CFPA-E No.19:2009 (2009)
3. British Standards Institute. The application of fire safety engineering principles to fire safety design of buildings. part 6: Human factors: Life safety strategies occupant evacuation, behaviour and conditions (sub-system 6). Technical report, PD7974-6:2004 (2004)
4. Kuligowski, E.D., Gwynne, S.M.V.: The need for behavioral theory in evacuation modeling. In: Klingsch, W.W.F., Rogsch, C., Schadschneider, A., Schreckenberg, M. (eds.) Pedestrian and Evacuation Dynamics 2008, pp. 721–732. Springer, Heidelberg (2010)
5. Lin, Y., Fedchenia, I., LaBarre, B., Tomastik, R.: Agent-based simulation of evacuation: An office building case study. In: Klingsch, W.W.F., Rogsch, C., Schadschneider, A., Schreckenberg, M. (eds.) Pedestrian and Evacuation Dynamics 2008, pp. 347–357. Springer, Heidelberg (2010)
6. Luo, L., Zhou, S., Cai, W., Low, M.Y.H., Tian, F., Wang, Y., Xiao, X., Chen, D.: Agent-based human behavior modeling for crowd simulation. Comput. Animat. Virtual Worlds 19(3-4), 271–281 (2008)
7. Nguyen, M.H., Ho, T.-V., Nguyen, T.A.N., Zucker, J.-D.: Which behavior is best in a fire evacuation: Simulation with the metro supermarket of hanoi. In: 2012 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), Ho Chi Minh City, Vietnam, 2012, February 27 - March 1, pp. 183–188. IEEE (2012)
8. Wang, Y., Zhang, L., Ma, J., Liu, L., Zhang, L., You, D.: Combining building and behavior models for evacuation planning. IEEE Computer Graphics and Applications 31, 42–55 (2011)
9. Weifeng, Y., Hai, T.K.: A model for simulation of crowd behaviour in the evacuation from a smoke-filled compartment. Physica A: Statistical Mechanics and its Applications (2011)

Interactive GA Flock Brush for Non-Photorealistic Rendering

Hsueh En Huang, Meng Hiot Lim, Xianshun Chen, and Choon Sing Ho

Centre for Computational Intelligence
School of Computer Engineering,
Nanyang Technological University Singapore
50 Nanyang Avenue, Singapore 639798
{hehuang1,emhlim,chen0469,csho}@ntu.edu.sg

Abstract. Art styles are modes of expressing creative artistic ideas. Non-photorealistic rendering is a process of projecting artistic expressions in a digital representation. In this paper, we consider the use of evolutionary computation techniques to explore the variability of artistic styles through an evolutionary process. Our system, a union of biological swarms in the form of flocks and interactive genetic algorithm (IGA), generates artistic styles to produce stylized digital photographs. By varying a finite set of parameters, we transform photo-realistic scenes to artistic imagery. Our most distinct styles bear close resemblance to familiar traditional art styles like Impressionism and Pointillism.

Keywords: Artificial intelligence, Computer graphics, Evolutionary art, Evolutionary computation, Non-photorealistic rendering, Flocking.

1 Introduction

Artistic expression has a rich history. Some examples of pioneering work include Jubal in Genesis and the Altamira cave painting, see Fig. 1. A great number of visual styles and artists followed. Egyptian, Greek, Roman and ancient East Asian art flourished one after another, the Renaissance ensued. The Renaissance gave birth to famous works such as the Sistine Chapel Ceiling (Fig. 1) and Mona Lisa. Baroque and Romanticism triumphed in the 17th to 18th century. Realism, Pointillism, Impressionism, Post-Impressionism dominated the 19th century followed by Fauvism, Dada and Surrealism. Styles such as Pop art, Installation art, Minimalism and Hyperrealism are contemporary forms that are still in-vogue [2].

Photography became an established art form in the 1900s and photo-realism joined the ranks of well-known art styles. Finally, digital art forms burst onto the scene. It differed from its predecessors. The widespread of information technology since the mid-twentieth century created a transition from a "material-based value system to a system in which immaterial information controls physical reality" This has had profound effects on the paradigm of art [3].

Today, digital art co-exists with traditional art forms. In computer science research, Non-Photorealistic Rendering (NPR) is an area where digital methods

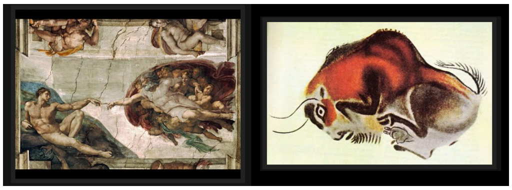


Fig. 1. The Sistine Chapel Ceiling and a cave painting at Altamira, Spain [1]

are used in conjunction with traditional artistic notions to express and replicate ideals of beauty, creativity and style. Artistic style is subjective and personal; individuals harbor different views on which styles are most fascinating. The work presented in this paper is about creating a platform for individuals to express themselves and discover their preferred artistic styles. Our proposed system models a 2D brush with a flock of autonomous agents. We used interactive genetic algorithm (IGA) as a means to evolve both flock and stroke parameters in order to generate artistic depictions of digital photographs. The goals of our research can be defined as follows:

1. To construct an interactive platform for individuals to experiment with artistic expression.
2. To discover interesting visual art styles by evolving flock and stroke parameters.
3. To discuss how the control parameters affect the resultant style.

The rest of the paper is structured as follows. Firstly, a summary of related works in NPR, Interactive Genetic Algorithm (IGA) and biological flock modelling is provided. This section also discusses how flocks are used as a brush model. We then describe the design of the flock brush system and how brush parameters are encoded and evolved using Genetic Algorithm (GA). This is followed by a discussion of the art styles generated. Finally, we conclude by summarizing our work and suggest possibilities for future research.

2 Background

2.1 Non-Photorealistic Rendering

One goal of Non-Photorealistic Rendering (NPR) is to create artistic styles that effectively communicate the "essence of the scene." This idea of communication through style is divergent from another major subject of study in computer graphics - photorealistic rendering. It is this goal that makes this field unique [4].

To accomplish this goal, researchers have used various methods and achieved a variety of results. Two broad approaches can be found in existing literature, stroke-based [5–7] and filter based [8,9]. A stroke based approach defines a painting or sketch as a series of strokes. Each stroke can have different attributes such as length, color and angle. Different artistic styles can be generated by varying stroke attributes, essentially like tuning different parameters. Stroke based methods were first used by Haeberli's system which used strokes of different color, length and orientation to create a painterly style [5]. Subsequent works introduced elements to enhance the artistic look of the resultant image. Stroke clipping using edge detection was introduced by Litwinowicz and Salisbury [6,7]. This ensured that a stroke would not extend beyond the outline of objects and improves the overall definition of the resultant artwork. Curved strokes modeled using B-splines were used by Hertzmann [10]. Besides these deterministic solutions, evolutionary computing (EC) techniques have also been used to create new NPR results.

2.2 Evolutionary Computing and NPR

Some existing work involving evolutionary computing and non-photorealistic rendering include the use of genetic programming (GP) to generate animated sketches [11], image stylization using a triangular brush and GP [12]. The input images were approximated using triangles of various sizes and then the resultant stylized images are combined into a video. Evolutionary search strategies can also be used to find an optimal artistic rendering of a photograph [13]. Each output from the system was scored using a fitness function and the optimal piece was then selected as the best artistic rendering. A distributed agent approach in the form of artificial ant colonies to generate different stylistic effects was employed by Semet [14].

Recently, the use of a Flock Brush, a brush modeled using flocks, coupled with edge following has also produced stylistic images from 2d digital photographs [15]. As the flock moves across the canvas a trail of paint is left as residue. (Fig. 2) depicts a painted stroke rendered by the Flock Brush. The system presented in this paper is an exploration and experimentation in the same vein as these previous works of synthesis between EC and NPR.

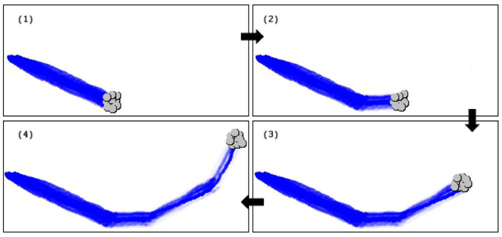


Fig. 2. A Single Stroke Rendered with the Flock Brush

2.3 Flock Modelling

Flock simulation was pioneered by Reynolds [16, 17]. Typically, agents in the same flock interact based on steering forces. The four forces in Reynold's model are separation, cohesion, alignment and seeking. These four steering forces are used to control the movement of the flock. His model and its derivatives have various usage and application. For example, Multi-agent systems use modified versions of Reynold's algorithm to improve the realism of flocking behaviour and simulate co-operation between agents [18,19].

2.4 Interactive Genetic Algorithms

Genetic algorithm (GA) was invent by John Holland in the 1970s [20]. The algorithm is a nature-inspired, population based meta-heuristic. It is often used as a optimization method for problems involving large search spaces. Traditionally, GA uses a fitness function to evaluate the quality of a solution. Under the Interactive Genetic Algorithm framework (IGA), a form of Interactive Evolutionary Computing (IEC), a "human in the loop" approach is used instead of a fitness function. IEC has been applied to a broad range of problems in various fields including, design [23] evolutionary music and storytelling [21,22]. Generative art made by cross-breeding mathematical functions is another common application of IEC [24–26].

3 Flock Paint with Interactive Genetic Algorithm

A Flock Brush is a digital brush modeled by a flock of agents [15]. It is possible to create stylistic effects by varying stroke and flock parameters and then guiding flock movement stochastically or deterministically with local gradient information. There are many different possible parameter combinations that can be used with the Flock Brush. Each combination has the potential to become a unique art style. Therefore, it is apt to use IGA to locate distinct artistic styles that can be produced with the Flock Brush. The remaining sections of this paper will detail the design and development of a system that uses IGA and the Flock Brush to explore and discover interesting artistic styles.

3.1 Flock Brush Parameters and Representation

The Interactive Genetic Algorithm Flock Brush System (IGA Flock Brush) utilizes various parameters to represent artistic styles. A list of the parameters used by the system can be found in Table. 1. The parameters are encoded as index-value pairs. Each parameter is defined over a suitable range of values which have been determined empirically. An illustration of the chromosomal encoding scheme is as shown in Fig. 3. For certain parameters we adopt the use of discrete intervals or bins to reduce the search space size and also to encourage distinct variations among the resultant images. For example, stroke length has a possible range of 10 pixels to 200 pixels. We divide the range into intervals of 10 pixels, hence the range of values are discretized to 10,20,30,...100,110,...200.

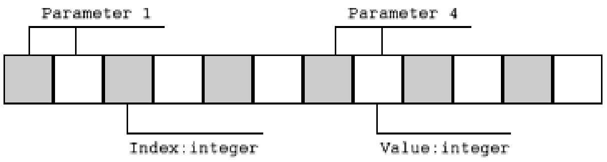


Fig. 3. Chromosome Encoding

3.2 System Modules

Our system consists of two main modules; the Flock Brush module and the Genetic Algorithm (GA) module, see Fig. 4. The Flock Brush module is primarily responsible for chromosome decoding, style rendering, display and capturing user inputs. The GA module is used for population initialization and performing the exploration of art style via the operations of crossover and mutation. Our system uses one-point crossover and a uniform mutation operator. The termination condition is user dependent. Usually, termination occurs when the user has found a satisfactory art style. Fitness of the resultant artwork is evaluated by the users. In general, aesthetics is subjective, therefore allowing individual users to assess the quality of the resultant images is not only viable but recommended.

Table 1. Flock Brush System Parameters

Name	Range	Description
Stroke Angle	{Fixed Deg., Grad., Orientation}	Orientation of the stroke
Color	{RGB, B/W}	Color source
Fade Rate	[0...1]	Paint fade rate
Brush Size	[1,2...25]	Radius of the brush
No. of Strokes	[0,1,2...].10000	No. of strokes painted per image
Separation	[0...1]	Separation steering force
Alignment	[0...1]	Alignment steering force
Cohesion	[0...1]	Cohesion steering force
Seek	[0...1]	Seek steering force
Sample Radius	[1,2...40]	Size of sampling area for flock creation
Flock Size	[1...12]	No. of agents in the flock
Flock Energy	[1,2...200]	Total energy level of the flock
Separation Dist.	[0,1...20]	Min. distance between two agents
Cohesion Range	[0,1,2...100]	Distance required for agents to be in a flock
Mark Shape	{Oval, Ring}	Shape of paint mark

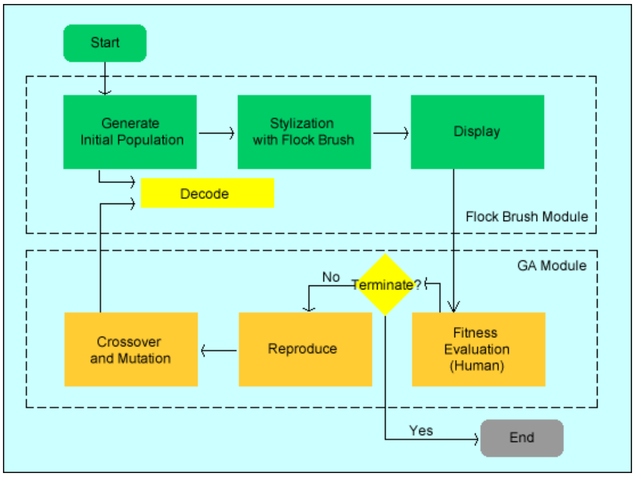


Fig. 4. Flock Brush with IGA System Work Flow

Algorithm 1: IGA Flock Brush

```
----------------------------------------------------------------
program IGA Flock Brush
----------------------------------------------------------------
BEGIN
  INITIALIZE a population of Flock Brush config. chromosomes
  WHILE (user has not found satisfactory stylistic image)
  DO
    STYLIZE image with Flock Brush based on parameters
            define by each chromosome
    DISPLAY image and prompt user to rate fitness
    REPRODUCE new population:
        Place user's most liked images in new population
        Select chromosomes for CROSSOVER
        Place offspring in the new population
        Select chromosomes for MUTATION
        Place offspring in the new population
  END WHILE
END
```

4 Results and Discussion

Using genetic algorithm to evolve flock brush parameters has lead to a variety of stylistic results. Some of these are depicted in Fig. 5 and Fig. 6. Each style is defined by a unique combination of the parametric values given in Table. 2. The results demonstrate the plasticity of the flock based brush model. We have also attempted to generalize the effects that a flock based brush model has on the generated art styles.

1. A flock based brush model creates non-uniform strokes that improves overall artistic realism.
2. Strokes made with the flock allows a certain degree of stochasticity for the shape and position of paint marks. This makes the resulting styles appear not completely random yet at the same time unpredictable.
3. Besides colour variations, changing the shape of the paint mark used by the flock can also create different styles. For example, Mosaic and Sandy style.

The generalized effects aside, each distinct style also possess its own set of attributes which are summarized as follow:

Impressionist: First, broad colored strokes are used followed by decreasing flock sampling radius for each layer. A tight flock with high energy is used to create a concentrated brush tip and longer strokes. The resultant image has a feel of loose brush strokes and coarse detail.

Smudge Paint: Multiple layers of oval shaped dabs are place over the canvas. The smudging effect is completed by the last layer of very short paint strokes that simulates the action of smudging wet paint. A tight flock with high energy is used.

Sketch: We use a grey-scale color source and a constant flock sampling radius to simulate a pencil tip. The use of a tight flock ensures the paint is concentrated at the tip. But varying the energy level used for each layer, the sketch is first outline in long strokes and then fine-tuned using shorter strokes.

Sandy and Pointillism: Both of these styles are created by making dabs on the canvas. Pointillism uses a tight flock while sandy uses flocks that are spread out. Sandy uses a much smaller brush size so the resultant texture looks like it's made by spreading fine sand on the canvas.

Mosaic: Similar to Pointillism but the shape painted by each agent is a ring and not oval. This gives the resultant image a tiled look so we name this style "Mosaic".

Rendering Using Multiple Layers: For some of the styles, we found that using multiple images or layers, led to a more aesthetically pleasing result. Each layer has its own parameter setting and the resultant image is a composite of the individual layers In our implementation, we used a simple over-paint approach when combining layers. Fig. 7 shows the the layered rendering process. The parameter settings for each style is shown in Table 2.

Table 2. Style Names and Key Parameter Values. For multi-layered styles, the value used for each layer are enclosed in braces.

Style Name	Impressionist	Sketch	Smudge Paint	Mosaic	Pointillism	Sandy
Color	RGB	B/W	RGB	RGB	RGB	RGB
Brush Size	2	2	{2,4,8,12,4}	{4,2}	{6,6}	{2,2,2}
Separation	0.20	0.20	0.20	0.20	0.20	0.20
Alignment	0.10	0.10	0.10	0.10	0.10	0.10
Cohesion	0.30	0.30	0.30	0.30	0.30	0.30
Seek	0.40	0.40	0.40	0.40	0.40	0.40
Sample Radius	{10,8,4,2}	{2,2,2,2}	{2,2,2,8,8}	{12,8}	{2,2}	{12,2,20}
Flock Size	{10,4,2,2}	{2,2,2,2}	{2,2,2,4,4}	{12,2}	{2,2}	{12,2,2}
Flock Energy	{180,100,60,40}	{100,50,30,10}	{0,0,0,0,30}	{0,0}	{0,0}	{0,0,0}
Separation Dist.	2	2	2	12	12	12
Cohesion Range	70	70	70	40	40	40
Mark Shape	Oval	Oval	Oval	Ring	Oval	Oval

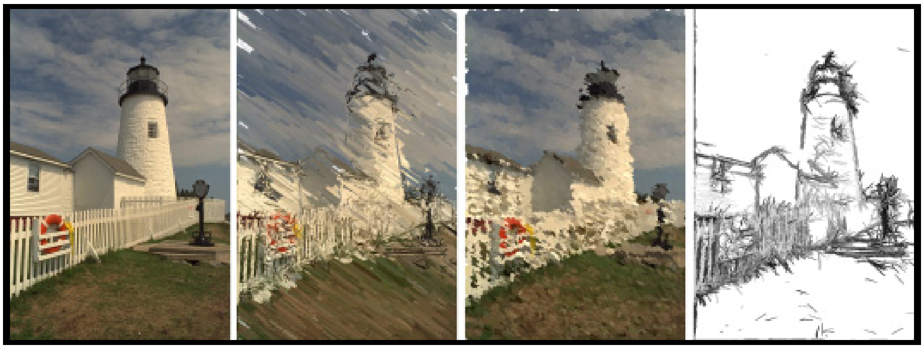


Fig. 5. From Left: (a) Source (b) Impressionist (c) Smudge Paint (d) Sketch

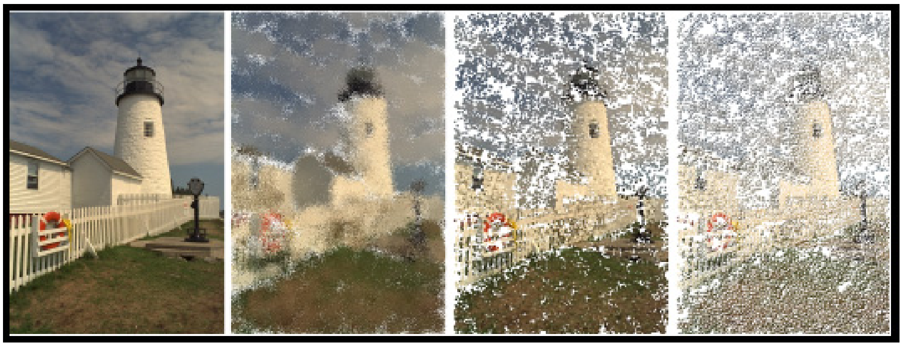


Fig. 6. From Left: (a) Source (b) Sandy (c) Pointillism (d) Mosaic

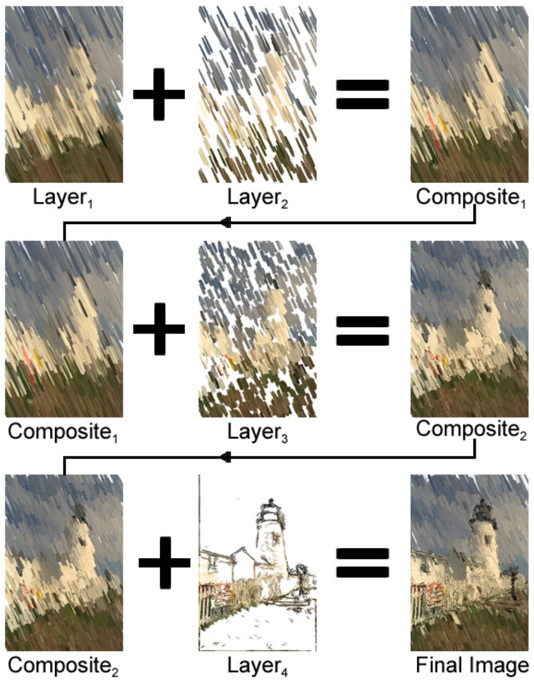


Fig. 7. Layers and Composite Images for the Impressionist Style

5 Conclusion and Future Work

In this paper, we present an IGA Flock Brush System; an interactive program for evolving digital non-photorealistic art styles by means of genetic algorithms (GA) and the Flock Brush [15]. The Flock Brush models a digital brush via steering forces and shape formation. In our program, this brush model provides a foundation for quantization of art styles. Through the use of this parameterized model, encoded as bit strings, we isolated salient characteristics of certain art styles. GA operators, crossover, mutation coupled with a "human in the loop" fitness enabled the interactive exploration of the art style space. The experimental explorations uncovered art styles that resembled a variety of traditional art styles. This work is yet another demonstration of the versatility of IGA and its usefulness in creative applications. Our future works will focus on incorporating additional parameters in to the model. We believe additional parameters will allow a more complete solution to the problem of quantifying art styles. Possible parameter candidates include the i) use of statistical distributions for the flock's initial position, ii) use of simple filters for image post-processing and iii) use of bitmap textures to simulate different mediums.

The flock brush model has the potential to be used in other applications. As the brush parameters are tunable, one could conveniently conceive that for certain settings the brush becomes a viable tool in digital calligraphy.

Although some existing parameters may need to be modified so the Flock Brush can effectively simulate a calligraphy brush. Lastly, the configuration we used for the parametrization of art styles can be thought of as a form of “building blocks” or more specifically cultural memes [27,28]. With the emergence of the memetic computing field [29–32], we foresee that meme inspired techniques would be very relevant to enhancing our system and poses interesting opportunity for future research.

Acknowledgments. The authors thank the staff of Centre for Computational Intelligence (C2I) at Nanyang Technological University Singapore for their support as well as Dr. Yew-Soon Ong, Director of C2I for his technical advices provided on the current work. Images used are taken from Kodak Lossless True Color Image Suite [33].

References

1. Public Domain Photos and Images, `http://public-domain-images.blogspot.com`
2. Farthing, S.: Art: From Cave Painting to Street Art- 40,000 Years of Creativity. Universe., New York (2010)
3. Grau, O.: MediaArtHistories. MIT Press, Cambridge (2007)
4. Gooch, B., Gooch, A.: Non-photorealistic rendering, AK Peters, MA (2001)
5. Haeberli, P.: Paint By Numbers: Abstract Image Representation. In: SIGGRAPH 1990 Conference Proceedings (1990)
6. Litwinowicz, P.: Processing images and video for an impressionist effect. In: Proceedings of SIGGRAPH 1997, pp. 407–414 (1997)
7. Salisbury, M.P., Anderson, S.E., Barzel, R., Salesin, D.H.: Interactive Pen-and-Ink Illustration. In: ACM SIGGRAPH 1994 Conference Proceedings, pp. 101–108 (1994)
8. Wang, H.: A Non-Stroke Based Method to Generate Sketching Style from Original Image Image and Signal Processing. In: Congress, CISP 2008 (2008)
9. Kyprianidis, J.E.: Image and video abstraction by multi-scale anisotropic Kuwahara filtering. In: NPAR, pp. 55–64 (2011)
10. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes. In: Proceedings of SIGGRAPH 1998, pp. 453–460 (1998)
11. Barile, P., Ciesielski, V., Trist, K.: Non-photorealistic Rendering Using Genetic Programming. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 299–308. Springer, Heidelberg (2008)
12. Izadi, A., Ciesielski, V., Berry, M.: Evolutionary Non Photo–Realistic Animations with Triangular Brushstrokes. In: Li, J. (ed.) AI 2010. LNCS (LNAI), vol. 6464, pp. 283–292. Springer, Heidelberg (2010)
13. Collomosse, J.P.: Evolutionary Search for the Artistic Rendering of Photographs. In: Romero, J., Machado, P. (eds.) The Art of Artificial Evolution, Natural Computing Series, pp. 39–62. Springer, Heidelberg (2008)
14. Semet, Y., O’Reilly, U.-M., Durand, F.: An Interactive Artificial Ant Approach to Non-photorealistic Rendering. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 188–200. Springer, Heidelberg (2004)

15. Huang, H.E., Ong, Y.S., Chen, X.: Flock Brush for Non-photorealistic Rendering. In: IEEE Congress on Evolutionary Computation (2012)
16. Reynolds, C.W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics 21(4), 25–34; SIGGRAPH 1987 Conference Proceedings (1987)
17. Reynolds, C.: Steering behaviors for autonomous characters. In: Game Developers Conference, pp. 763–782 (1999)
18. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: Algorithms and theory. IEEE Transactions on Automatic Control 51(3), 401–420 (2006)
19. Ho, C.S., Nguyen, Q.H., Ong, Y.-S., Chen, X.: Autonomous Multi-agents in Flexible Flock Formation. In: Boulic, R., Chrysanthou, Y., Komura, T. (eds.) MIG 2010. LNCS, vol. 6459, pp. 375–385. Springer, Heidelberg (2010)
20. Holland, J.: Adaptation in natural and artificial systems. MIT Press, Cambridge (1992)
21. Bui, V., Abbass, H.A., Bender, A.: Evolving stories: Grammar evolution for automatic plot generation. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
22. Horowitz, D.: Generating Rhythms with Genetic Algorithms. In: AAAI (1994)
23. Li, Y., Hu, C., Yao, X.: Innovative Batik Design with an Interactive Evolutionary Art System. J. Comput. Sci. Technol. 24(6), 1035–1047 (2009)
24. Sims, K.: Artificial evolution for computer graphics. Computer Graphics, 319–328 (1991)
25. Unemi, T.: SBArt4 - Breeding abstract animations in realtime. In: IEEE Congress on Evolutionary Computation, pp. 1–6 (2010)
26. Ross, B.J., Ralph, W., Zong, H.: Evolutionary Image Synthesis Using a Model of Aesthetics. In: Proceedings of the 2006 IEEE Congress on Evolutionary Computation (2006)
27. Chen, X.S., Ong, Y.S., Lim, M.H., Tan, K.C.: A Multi-Facet Survey on Memetic Computation. IEEE Transactions on Evolutionary Computation 15(5), 591–607 (2011)
28. Ong, Y.S., Chen, X.S., Lim, M.H.: Research Frontier: Memetic Computation - Past, Present & Future. IEEE Computational Intelligence Magazine 5(2), 24–36 (2010)
29. Liang, F., Ong, Y.S., Tan, A.-H., Chen, X.S.: Towards Human-like Social Multi-agents with Memetic Automaton. In: IEEE Congress on Evolutionary Computation (June 2011)
30. Winfield, A., Erbas, M.: On embodied memetic evolution and the emergence of behavioural traditions in robots. Memetic Computing 3, 261–270 (2011)
31. Satizbal, H., Upegui, A., Perez-Uribe, A., Rtornaz, P., Mondada, F.: A social approach for target localization: simulation and implementation in the marxbot robot. Memetic Computing 3, 245–259 (2011)
32. Le, M.N., Ong, Y.S., Jin, Y., Sendhoff, B.: A Unified Framework for Symbiosis of Evolutionary Mechanisms with Application to Water Clusters Potential Model Design. IEEE Computational Intelligence Magazine 7(1), 20–35 (2012)
33. Kodak Lossless True Color Image Suite, http://r0k.us/graphics/kodak/

Generating Diverse Behaviors of Evolutionary Robots with Speciation for Theory of Mind

Si-Hyuk Yi and Sung-Bae Cho

Dept. of Computer Science Yonsei University
50 Yonsei-ro, Seodaemoon-gu, Seoul 120-749, Korea
theshy@sclab.yonsei.ac.kr, sbcho@cs.yonsei.ac.kr

Abstract. Theory of Mind (ToM) is the ability to read another person's mind. To apply ToM in robots, robot should read the intention from target. However, it is difficult to read target's intention directly. Robot uses the sensors to measure distance from target because distance is the feature to read target's intention. Neural network has been widely used to control the robot for generating a diverse speciation. It has been less explored in behavior-based robotics. Speciation usually relies on a distance measure that allows different from the robot to target to be compared. In this paper, we proposed novel measure to generate diverse behaviors of a robot with speciation for ToM. It includes some distance measure such as Euclidean distance, cosine distance, arctangent distance, and edit distance. It generates diverse behaviors of the robot by neural network for ToM. The proposed method has been experimented on a real e-puck robot platform.

Keywords: Theory of mind, Evolutionary Robotics, Robot Controller, Speciation, Distance Measure.

1 Introduction

A theory of mind (ToM) is the ability to understand other's thoughts and feelings. Primates and great apes have the ability to read behavior. For example, they can figure out behavior by reading gestures, intention movements, and facial expressions of emotions [1]. In case of robot environment, they use the sensors such as infrared, camera, and sound because they cannot directly read target's intention. The distance is the most widely used information for trajectory, direction, and strategy. In a prey-predator model, if a predator reads the prey's behavior strategy from trajectory, it can easily hunt on discovered route. The robot needs a controller which can make the proper behavior patterns because the method of measuring distance can generate diverse patterns. The distance, therefore, is a fundamental feature for robot.

Evolutionary Robotics (ER) is a promising methodology, intended for the autonomous development of robots, in which their behaviors are obtained as a consequence of the structural coupling between robot and environment [2]. That is, robots get primarily inputted through the sensors as an environment, and then their behavior is determined by evolution of neural networks. It is a method to generate a behavior which interacts between robot and environment [3].

Speciation is incorporated to the evolution process in order to obtain a diverse set of solutions using single algorithm. Although speciation is a kind of evolutionary computation, it has been less explored in behavior-based robotics. Speciation usually relies on a distance measure from own robot to target [4].

In this paper, we proposed novel measure to generate diverse behaviors of a robot. Measure combines in Euclidean distance, cosine distance, arctangent distance, and edit distance. The purpose of measure is to generate diverse behaviors of the robot by neural network for ToM. The proposed method has been implemented on a real e-puck robot platform.

Table 1. Approaches of distance measure

Author	**Distance measure**	**method**	**Year**
S. Luke [8]	Euclidean distance	Genetic Programming	1996
L. Spector [9]	Euclidean distance	Genetic Programming	2005
L. Trujillo [10]	Edit distance	NEAT	2008
T.-S. Lee [11]	Euclidean distance, Angle distance	Proposed algorithm	2010

2 Related Work

2.1 ToM Applications in Simulation

There are related works on ToM based application in simulated environments. Kaliouby et al. proposed a mind-reading machine that recognizes person's discrete mental states from video of the facial expression [5]. It is modeled on a Bayesian network. Kondo et al. used the ToM in carrying a stick task for the cooperation of two computer programs. They use the neural network for modeling [6]. Takano et al. applied ToM to complex agent-based simulations for collision avoiding system [7]. These works focus on modeling on simulated environment. In this paper, we implement on the real robot platform for ToM.

2.2 Speciation with Distance Measure

Table 1 shows related works using other distance measure for robot. In speciation, one of the best approaches is the Neuro-Evolution of Augmenting Topologies (NEAT) method, a specialized GA that evolves a population of ANNs with variety topologies [4]. Drchal et al. describe a simulation of autonomous robots controlled by recurrent neural networks [12]. HyperNEAT algorithm is evolved through indirect encoding. The robots utilize 180 degrees wide sensor array. Trujillo et al. employed edit distance for behavior based speciation [10]. The environment is partitioned into a graph and each node labeled string. The fitness value is calculated by the string track of a robot. To use this value, controller adopts some features using fitness sharing among the neural networks. It represents relatively more diverse speciation than NEAT in reflecting trajectory. It is, however, difficult to measure similarity in the trajectory from another behavioral space which cannot generate diverse patterns without an obstacle.

Table 2. Specificity of each distance measure

Measurement	Specificity
Euclidean distance	Various movement pattern
Cosine distance	Wide and length of route
Arctangent distance	Wide and length of route
Edit distance	Alternate location in same route

3 Proposed Method

Fig. 1 shows the overview of proposed method using the evolution algorithm. The model generates the number of p with initialization in neural networks. And it generates changed each network by mutation operation. The robot finds a goal such as light using each neural network. We apply to fitness sharing how exactly network gets to the destination. It analyzes whole robot trajectory with location through sum of some distance each time. The model conducts evolution based on the highest value among the estimated fitness values.

3.1 Neural Network Controller

Controller designed by neural networks. Goal of neural network is to conquer the light such as food, prey, fodder, and home. Neural networks are made up of 2-inputs and 2-outputs. Inputs are IR sensor value that can know a relative distance and direction of light from a robot. Outputs are speed of wheels. The controller can determine the robot's direction and rate of movement using speed of each wheel. The fitness estimation designed that how exactly networks to find light. We use the Euclidian distance to measure from a destination of the robot last generation.

$$f(i) = \sqrt{\left(x_{light} - x(last)_i\right)^2 + \left(y_{light} - y(last)_i\right)^2} \quad (1)$$

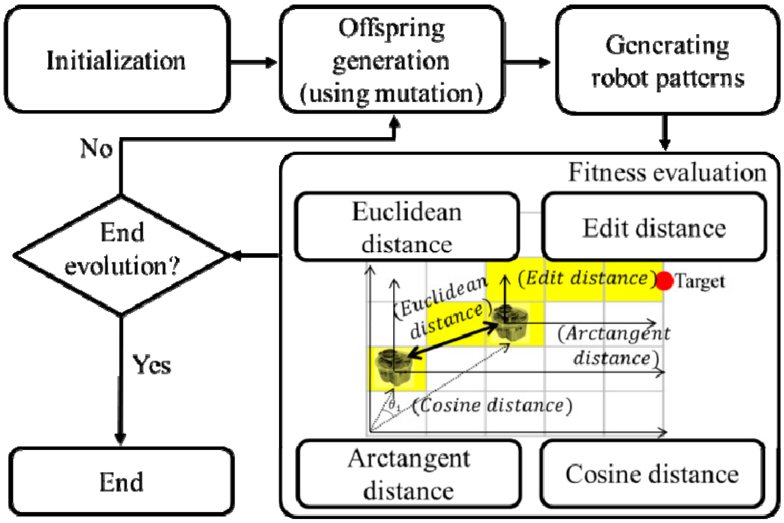


Fig. 1. Overview of proposed method

3.2 Behavior Based Fitness Sharing

A speciation is a method to maintain the diverse behavior. In this paper, we employ fitness sharing which transforms fitness value for maintaining a diverse behavior. In this respect, the distance between objects is more proper than other measurements. It is a method to prevent local optimized solution in dense individual who located near distance.

f_i is fitness value of network i, and fitness sharing value f'_irepresents fitness divided by the sum of sharing function value in eq. (2).

$$f'_i = \frac{f_i}{\sum_{j=1}^{\mu} \mathrm{sh}(d(i,j))} \tag{2}$$

$$\mathrm{sh}\big(d(i,j)\big) = \begin{cases} 1-(d(i,j)/\sigma_{\mathrm{share}})^{\alpha} & if\ d(i,j) < \sigma_{\mathrm{share}} \\ 0 & otherwise \end{cases} \tag{3}$$

where μ is the size of population, σ_{share} is sharing radius. Fitness sharing is conducted when the objects locate within the sharing radius. $\mathrm{d(i,j)}$ represents the distance from network i to j.

To generate a variety pattern of the robot, we apply the controller to distance measure such as Euclidean distance, cosine distance, arctangent distance, and edit distance. Based on these measures, we designed controller evolution speciation using the fitness sharing. As a result, we can find out the feature as Table 2. To combine these measures linearly, controller can generate a pattern that applied the characteristic of each measure.

$$\mathrm{d(i,j)} = \mathrm{C_1 d(i,j)_{euclidian}} + \mathrm{C_2 d(i,j)_{editdistance}} + \mathrm{C_3 d(i,j)_{arctan}} + \mathrm{C_4 d(i,j)_{cosine}} \tag{4}$$

The eq. (4) which applies four distance measure is as follows. C_1, C_2, C_3, and C_4 are decided according to the weight among each distance measure. Euclid distance must consider as the most important factor due to generating the diversity of movement pattern. Cosine distance and arctangent distance is used for changing the width and length of diversified pattern. In case of the cosine distance, robot cannot find the goal when generation proceeds over the specific generation. For the reason, weight of cosine distance must be a low value. In case of edit distance can change whole trajectory. So its weight has anything value. Therefore, the weight of each distance measure is like eq. (5).

$$d(i,j) = C_1 > C_2 \geq C_3 > C_4 \tag{5}$$

3.3 Physical Distance

As the feature of the trajectory used by fitness sharing, there is a pattern because of the physical distance difference. It can keep the diversity of the trajectory made by network. Because the pattern changes for finding the goal despite the physical distance is long.

$$d(i,j)_{euclidian} = \sum_{n=0}^{max\_steps} \sqrt{(x(n)_i - x(n)_j)^2 + (y(n)_i - y(n)_j)^2} \tag{6}$$

The Euclidean distance between neural networks i and j can be calculated by summation of each distance about robot's movement coordinates on step. It can make the diverse trajectory, but not make dramatically the width.

3.4 Angle Distance

When sharing fitness, we can use behavior of angle by features. Angle difference of behavior has an influence width of the trajectory. It can be measured by two kinds of method. Firstly, cosine distance calculates cosine value at location of behavior on each step.

$$d(i,j)_{cosine} = \sum_{n=0}^{max\_steps} (1 - \cos\theta) = \sum_{n=0}^{max\_steps} \left(1 - \frac{A \cdot B}{|A| \cdot |B|}\right) \tag{7}$$

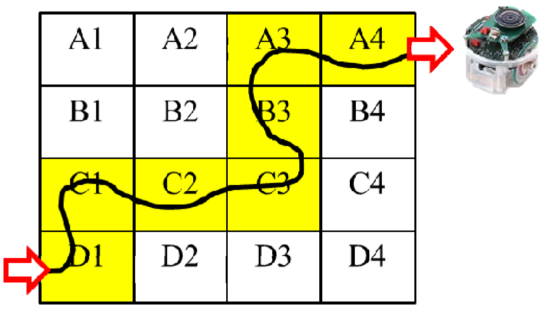


Fig. 2. Map is divided by 4×4 grid, and gives a letter. Colored parts are mapping route.

Cosine value has from 0 to 1. The cosine value has the closer 1, angle has the closer 0 degree. Then the angle location of two points is closer than a previous point. Cosine distance value of neural network i and j can be calculated by eq. (7). After calculating the angle difference of robot location of each step and summing each value, controller gets the angle of distance in the trajectory. Secondly, arctangent distance calculates distance from the mobile robot to a goal on each behavioral step.

Arctangent value is where time position is located in light. Thus distance from robot to goal can calculate like eq. (8).

$$\begin{aligned} subY(n)_i &= (y_{goal} - y(n)_i) \\ subX(n)_i &= (x_{goal} - x(n)_i) \\ subY(n)_j &= (y_{goal} - y(n)_j) \\ subX(n)_j &= (x_{goal} - x(n)_j) \end{aligned} \tag{8}$$

where n is stepped when the robot move in, that is, means time. Location of light defines (x_{goal}, y_{goal}). In n step, location robot reaches by network i and j define each $(x(n)_i, y(n)_i)$and $(x(n)_j, y(n)_j)$. Arctangent distance can calculate difference such as eq. (9).

$$d(i,j)_{euclidian} = \sum_{n=0}^{max\_steps} \begin{matrix} |arctan(subY(n)_i / subX(n)_i) \\ -arctan(subY(n)_j / subX(n)_j)| \end{matrix} \tag{9}$$

Arctangent value of neural network between i and j is an absolute value of difference from a goal in n step. The robot is moved by j network and arctangent value from a goal in n point by i network. It means that the smaller this value size is, the closer two points are located in angle by goal. That is, features of angle values are similar characteristic from goal.

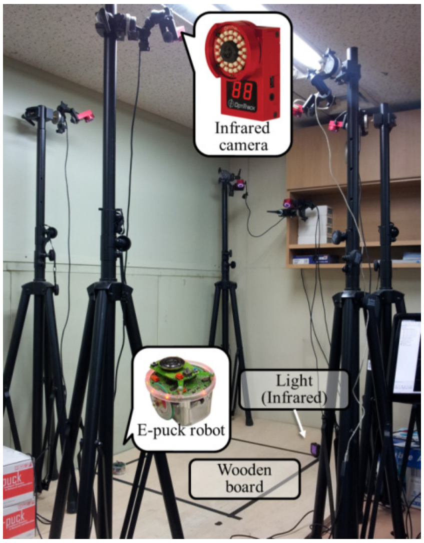


Fig. 3. Tracking system environment

Although features of angle don't generate variously a form of robot pattern, give feature to change width and distance of a same trajectory. After cosine distance evolves some generation, robot cannot find a goal because of using only angle value. If it moves any close place at the starting point, distance based on the angle is sensitively decided.

3.5 Edit Distance

To avoid the same movement route, by using the edit distance and measuring the distance of networks, it can be the distance measure of fitness sharing.

The edit distance is an algorithm to edit given two strings by the minimum value from original string to wanted string. In other words, edit distance of given two strings is the operator number to transfer a letter to another letter.

Map of behavioral space is divided by plaid form, and give a specific letter by each area called cell. When the robot passes the cell, controller adds the letter of this area at letter string. An example of the process is shown in Fig 2. Map is divided by 4×4 grid, and gives a letter by each part. Other color parts are a route of the robot. Letter string value, ‘s’, is determined by this method. The controller conducts neural network evolution by fitness sharing of edit distance using this value.

Behavior route found no significant differences. However, we confirm that the same pattern of route appears at different location.

4 Experiment

4.1 Experiments Settings

Experiment proceeds in real robot environment which is used an e-puck robot platform. Size of the map defines 120cm in width and length. The Coordinate is calculated by 1cm and mark (x, y) form. The location of light is (100, 100) and starting point of the robot set up (0, 0). The value of p, which means a number of neural networks, sets 20. And the parameters C1, C2, C3, and C4 are 10, 2, 1, and 0.5, respectively. We tested features of each distance measure such as the maximum weight each physical distance, angle distance and edit distance. And controller reflects all features of each distance measure through various trajectories to generate. Details see Table 3.

4.2 Tracking Environment

To track robot patterns, we use a motion capture system called “Optitrack”. The Optitrack consists of multiple infrared cameras like Fig. 3. And the system detects the markers like Fig. 4. The marker based system generally uses spherical retro-reflective markers that can be identified by the cameras. In this experiment, the markers which patched on the robot body are provided by the NaturalPoint Company. Markers are placed on the robot body suit in a configuration that is defined by the software.

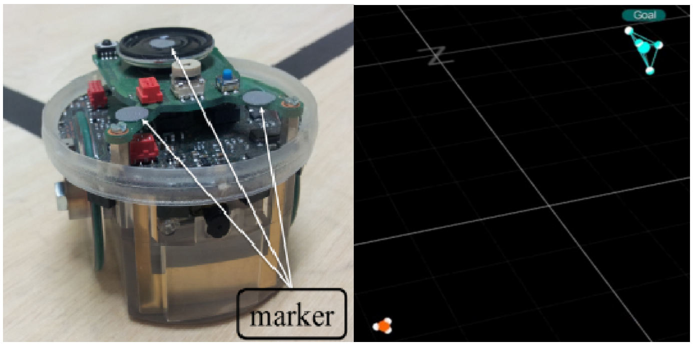
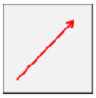
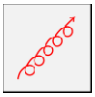
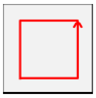
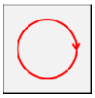
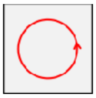
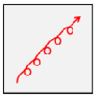
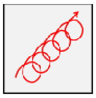


Fig. 4. Markers for detection on e-puck robot platform (left), marker based tracking software (right)

4.3 Result

Controller was evolved for 20 generations on the real e-puck platform. It represents three types of patterns-right, left, and straight like Fig. 5. An analysis of the generated pattern is shown in the Table 3. We can see the different from in simulation results which 1000 generations generated. In point of success, robot got more the goal in simulation. The real robot environment can vary the value of various external features called reality gap. The light has high uncertainty, which is not always same value as input in similar place. Controller can generate other values from uncertain input value despite same location.

Table 3. Analysis of generated patterns

Spin			***Direction***		***Turning***	
None	Forward	Loop	CW	CCW	Tight	loose
						
Success : 60% (in simulation, 92.3)						
58.33	16.67	25	67.67	33.33	58.33	41.67
Failure : 40% (in simulation, 0.7)						
37.5	25	37.5	60	40	40	60

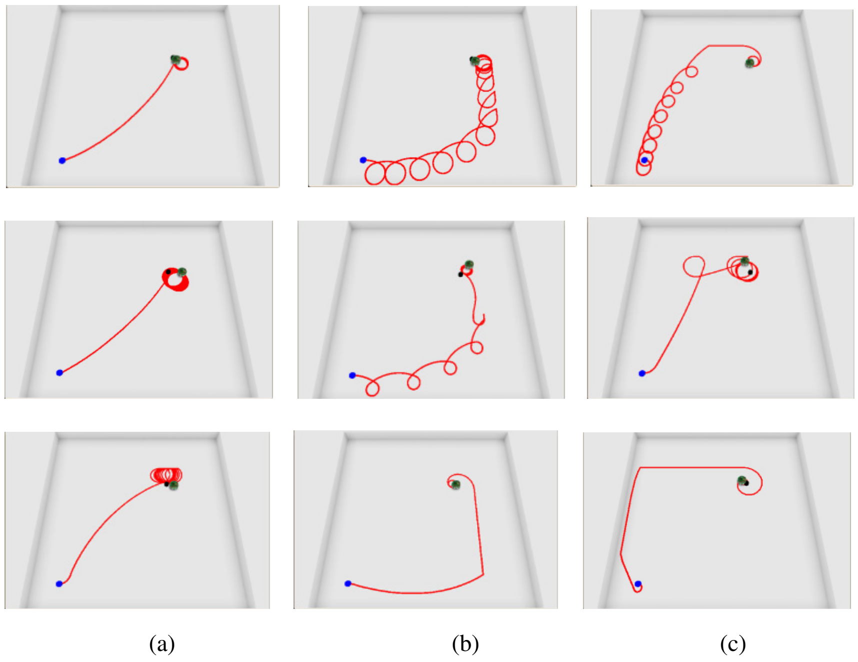


Fig. 5. Result of typical patterns (a) Go straight, (b) keep right, (c) keep left

5 Conclusion

In this paper, we proposed a method to generate diverse behaviors of the evolutionary robot by using speciated neural network based on behavior. Generated behavior patterns are important features to read target's mind or intention based on ToM. If the robot behaves according to the proper pattern of target's intention, can generate the robust behavior strategy. Features of a robot are physical distance, cosine distance, arctangent distance, and edit distance. We set up standard distance based on this features. We applied Euclidian distance to measure for diversity of the trajectory. Next, arctangent distance and edit distance are used based on moving area. Finally, we appropriate cosine distance for trajectory range size. In the experiment, we confirm that diverse trajectories to reflect all characteristics of each distance measure in real robot platform. We also can generate the patterns' combination of spin, direction, and turning.

In future work, we plan to extend the model to generate various patterns in harsh environment such as moving target or obstacle.

Acknowledgment. This research was supported by the Original Technology Research Program for Brain Science through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0018948).

References

1. Brune, M.: "Theory of Mind" in Schizophrenia: A review of the Literature. Schizophrenia Bulletin 31(1), 21–42 (2005)
2. Zagal, J.C., del Solar, J.R.: Combining Simulation and Reality in Evolutionary Robotics. J. of Intelligent Robotics and Systems (2007)
3. Nolfi, S., Floreano, D.: Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. Bradfor Book (2004)
4. Trujillo, L., Olague, G., Lutton, E., Vega, F.F., Dozal, L., Clemente, E.: Speciation in behavioral space for evolutionary robotics. J. of Intelligent & Robotic Systems 64(3), 323–351 (2011)
5. Kaliouby, R.E., Robinson, P.: Mind reading machines: Automated inference of cognitive mental states from video. In: Int. Conf. on Systems, Man and Cybernetics, pp. 682–688 (2004)
6. Kondo, K., Nishikawa, I.: The role that the internal model of the others plays in cooperative behavior. In: Proc. of Int. Workshop on Robot and Human Interactive Communication, pp. 265–270 (2003)
7. Takano, M., Arita, T.: Asymmetry between even and odd levels of recursion in a theory of mind. In: Proc. of ALIFE X, pp. 405–411 (2006)
8. Luke, S., Spector, L.: Evolving teamwork and coordination with genetic programming. In: Proc. of Conf. on Genetic Programming, pp. 150–156 (1996)
9. Spector, L., Klein, J., Feinstein, C.P.M.: Emergence of collective behavior in evolving populations of flying agents. Genetic Programming and Evolvable Machines 6(1), 111–125 (2005)
10. Trujillo, L., Olague, G., Lutton, E., Vega, F.F.: Behavior-based speciation for evolutionary robotics. In: Proc. of Conf. on Genetic and Evolutionary Computation, pp. 297–298 (2008)
11. Lee, T.-S., Eoh, G.-H., Kim, J., Lee, B.-H.: Mobile robot navigation with reactive free space estimation. In: Int. Conf. Intelligent Robots and Systems, pp. 1799–1804 (2010)
12. Drchal, J., Koutnik, J., Snorek, M.: HyperNEAT controlled robots learn how to drive on roads in simulated environment. In: Proc. of Congress on Evolutionary Computation, pp. 1097–1092 (2009)

Improving Gender Recognition Using Genetic Algorithms

Abbas Roayaie Ardakany and Sushil J. Louis

Dept. of Computer Science and Engineering
University of Nevada, Reno
roayaei@yahoo.com, sushil@cse.unr.edu

Abstract. This paper attacks the problem of gender classification using genetic algorithms. We use local binary patterns and principle component analysis to extract a set of features from face images in the FERET database. The genetic algorithm searches the space of feature subsets to find a set of features that maximizes gender classification accuracy using a support vector machine classifier with hand tuned parameters. Starting with 142 features, the genetic algorithm reduces the feature set size by approximately half resulting in 98.5% accuracy with 100% reliability. This accuracy and reliability are better than when using all 142 features.

Keywords: Feature Subset Selection, Genetic Algorithms, Gender recognition.

1 Introduction

The face is an important biometric feature for humans. Automatically recognizing and analyzing faces is a challenging task in the object recognition research area. Successfully performing this task allows many applications in human computer interaction, psychology, and security [1]. Prior research has shown that we can obtain information on ethnicity, identity, age, gender, and expression from face images [2]. This paper investigates a new approach to gender classification from face images.

Gender plays a significant role in our interactions in society and with computers [3]. Gender classification is the binary classification problem of predicting whether a given image contains a picture of a man or of a woman. Identifying gender from face images has received much attention recently because of its applications in improving search engine retrieval accuracy, demographic data collection, and human–computer interfaces (adjusting the software behavior with respect to the user gender) [5]. Moreover, gender classification can be used as a preprocessing step for face recognition since it may halve the number of face candidates, assuming we have equal numbers of images of both genders, before the recognition. Such preprocessing can sometimes double the speed of face recognition systems [1].

Like other image classification problems, we need to extract relevant features and then apply our classifier. From the point of view of feature extraction, there are four kinds of methods. First, we can simply use gray-scale or color pixel vectors as features [6]. Second, subspace transformation theory presents us with approaches such as Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Linear Discriminant Analysis (LDA) which transform images into a low-dimensional space [7]. The disadvantage with these kinds of methods is that they are sensitive to face orientation and cannot tolerate large variations [8]. Third, we can use texture information like wrinkle and complexion [9]. Finally, we can extract local facial features for classification such as the analysis of facial wrinkles and shapes [8]. This is done using a combination of facial feature detection with wavelet transforms [10, 11].

From the view of classifier learning, many different methods have been tried. [12, 13] used a two-layer neural network where the first layer is responsible for feature extraction and the second layer performs the classification. In [14] radial basis function (RBF) networks and inductive decision trees were employed. In [15] maximum-likelihood classification is used for face detection and the superiority of nonlinear SVMs over traditional linear pattern classifiers was proven. [16] utilizes Gaussian process classifiers which are a kind of Bayesian kernel classifier. This method obviates the problem of determining kernel parameters in SVMs. [17] tried to improve generalization for gender classification using fuzzy SVMs. Another method which is used for gender classification is Adaboost. [18] applied a threshold Adaboost classifier for gender and ethnicity classification. Look up table (LUT) based Adaboost classifier is used in [19]. [20] used pixel comparison operators with Adaboost on low resolution grayscale face images. [21, 1] combined face detection and gender classification and carried out a comparison study for the state-of-the-art gender classification methods. It has been shown that combining the outputs of different gender classifiers can improve the classification rate [1].

In this paper we present a method which uses local binary pattern for feature extraction and genetic algorithms for feature selection. We use a support vector machine for classification. The next section provides some background on our feature extractors, genetic algorithm, and classifier system. Methodology and genetic algorithm parameters are in section three. Results and conclusions are presented in the last section.

2 Tools

2.1 Local Binary Pattern

Local binary pattern was first presented in [22]. This algorithm can be used for describing texture or shapes in digital images and we can extract information in a specified neighborhood of a point using this method. In this method, a binary descriptor (LBP code) is used for describing a neighborhood. This code is obtained by comparing the intensity value of a point (the central point of a window) with its neighboring points. In an 8-bit neighborhood, if the intensity value of a pixel is greater or equal than the central point value then "1" is assigned to the corresponding bit in an 8-bit binary code which is representative of the neighborhood texture for the central point. But if the intensity value is less than the central point value then "0" is assigned to the corresponding bit (Figure 1). By applying this operator on an image we can compute the histogram for new pixel values. This histogram is used as a feature vector.

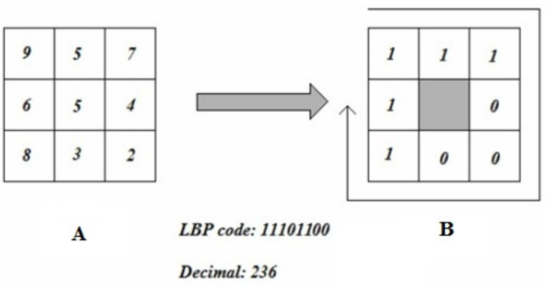


Fig. 1. A) Intensity values. B) LBP code

In this paper, we use uniform local binary pattern which is an extended version of LBP. A local binary pattern is called uniform if it has two or less bit transitions from zero to one or vice versa.

2.2 Genetic Algorithms

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They efficiently exploit historical information to speculate on new search points with expected improved performance [23]. The CHC is a non-traditional GA which combines a conservative selection strategy (that always preserves the best individuals found so far) with a highly disruptive recombination (HUX) that produces offspring that are maximally different from their two parents. We use CHC's selection algorithm with Uniform Crossover (UX).

2.3 Principal Component Analysis

We also use Principal Component Analysis (PCA) to reduce the number of features for our support vector machine classifier. PCA is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (i.e., uncorrelated with) the preceding components [24]. As a result, we can use PCA for dimensionality reduction purposes without losing considerable amounts of information.

2.4 Support Vector Machine

A support vector machine (SVM) is a learning algorithm for pattern classification, regression and density estimation [25, 26]. The idea behind SVMs is constructing the optimal linear hyperplane between two classes so that the classification error is minimized. So SVMs can be applied for our gender classification task. SVM compromises between model complexity and generalization to achieve the best generalization (that is to classify any face image correctly). We use openCV SVM which is based on LIBSVM implementation of SVM with linear kernel and fixed parameter C=900.

2.5 Database

FERET [27] is a large-scale face database which currently contains 11,383 pose images of 994 individuals. FERET contains thousands of samples with 12 different poses. In this work, we use frontal face images (fa) which includes 1364 images from 994 individuals. Table 1 shows the database in detail.

Table 1. Part of FERET database which has been used in this paper

Database	Original size	Total number	Number of Individuals	Female	Male
FERET	512×768	1364	994	504	860

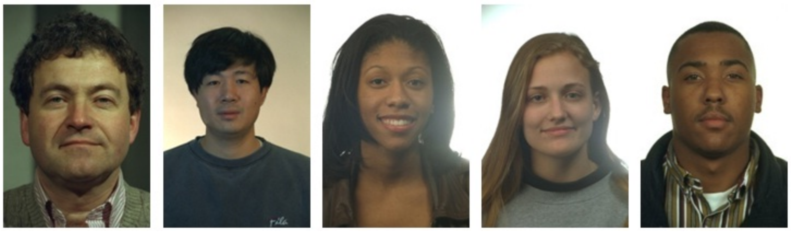


Fig. 2. Some samples from FERET database

2.6 Feature Extraction

Before we can extract features from face images we need to extract face coordinates which, in this project, is carried out using the Viola Jones [28] face detection algorithm (Figure 3).

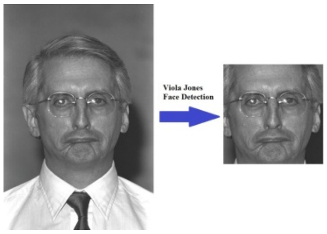


Fig. 3. Viola Jones face detection result

Then, for extracting features, we compute the LBP histogram for each face image (Figure 4). This histogram contains information about the distribution of local patterns in each image, such as edges, spots and flat areas.

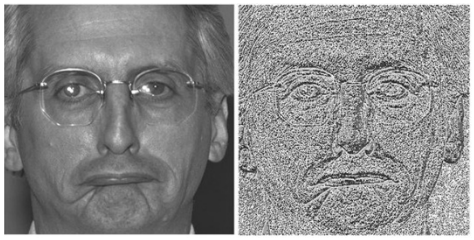


Fig. 4. LBP output for a particular image

For efficient face representation, we need to also consider spatial information. For this purpose, each face image is divided into K×K separate regions which are equal size blocks. Then, the histogram is computed in each block separately. Finally, we put these sub-regional histograms together to form a single feature histogram (Figure 5).

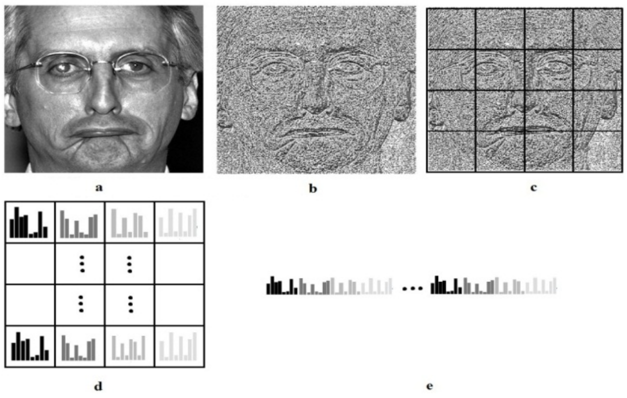


Fig. 5. (a) Original image (b) LBP encoded image (c) dividing each LBP image into K×K equal blocks (d) computing histogram for each block separately (d) putting sub-regional histograms together and create a single histogram

After feature extraction, we apply PCA to reduce the number of features from 1475 to 142 while preserving 98% of the information from our LBP feature extractor.

3 Encoding and Methodology

We use genetic algorithms to select features that maximized classification accuracy for our SVM. In other words, after applying PCA, we use a GA to select the best possible combination of features that can achieve the highest classification accuracy.

There are three main reasons that can justify applying feature selection on the output of PCA in a classification problem. First, there is always some noise in our resulting feature set that can be resolved using a good feature selection algorithm. Second, dimensionality can lead to faster classification. Third, non-linear feature interactions may result in lower principal components having higher discriminatory power at the particular classification task for the given classification algorithm.

We represent each individual chromosome as an array of ones and zeros. Each index corresponds to a feature. Having "1" at an index implies that we are selecting that feature, a "0" means that we are not going to use that feature for classification. We evaluate the selected features by training a system using our SVM classifier. We investigate two approaches that differ in how we initialize the GA's population.

1. We run the system for ten different seeds with population size of 100. In each run, each individual i in the population is initialized with a different probability of having a "1" for each position in the chromosome. This probability is $P_i = {}^{i}/_{100}$ for individual i. Consequently, we start the GA with a population that contains individuals with the number of features ranging from 1 to 142 (the original number of features in dataset). We call this Multiple Initialization or "MI".
2. We run the system for ten different seeds, same population size (100), and probability $P_i = 0.5$. We call this Regular Initialization or "RI". The whole process of training and testing is illustrated in Figure 6 and we present our results in the next section.

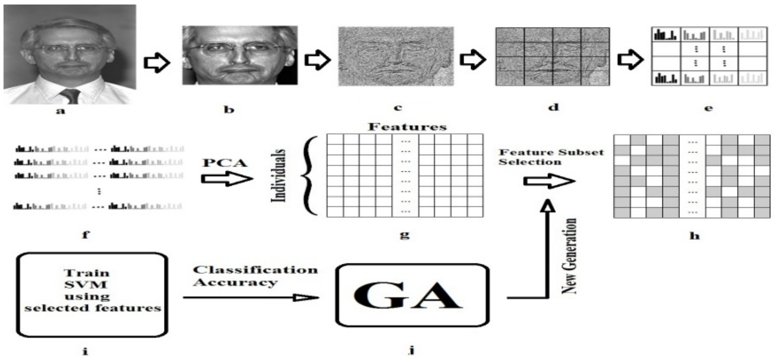


Fig. 6. a) Original face image b) Detected face plus face normalization. c) LBP coded image d) divided each image into k×k blocks e) histogram each block separately f) extracted feature vectors for all images by putting together the histograms g) resulted dataset after applying PCA h) a set of individuals (feature vector) after feature selection i) train SVM j) genetic algorithms.

4 Results and Discussion

For the MI approach the best result that we have obtained is 98.5% with 69 selected features. Figure 7 shows performance plotted against number of iterations. The quality of the first approach which is the average of best individual over all runs is 98.5%.

The reliability measure of our approach, defined as the percentage of runs in which we get within 99% of our highest quality (0.99×Quality) is 100%.

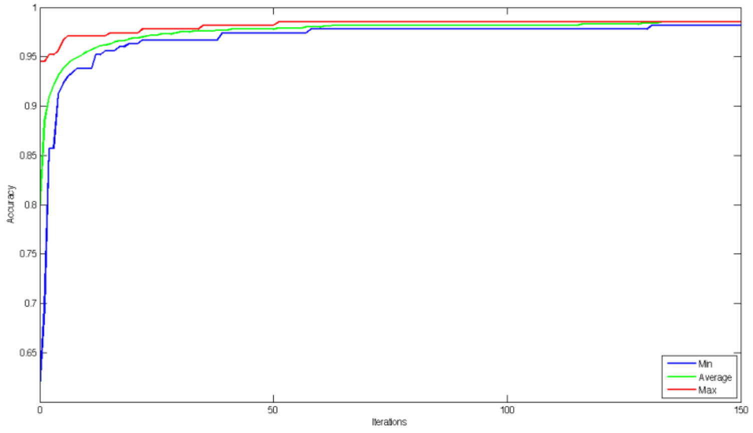


Fig. 7. Performance graph for MI approach

Figure 8 presents an illustration of the best obtained feature set using MI approach (108 features out of 142). White area corresponds to the selected regions.

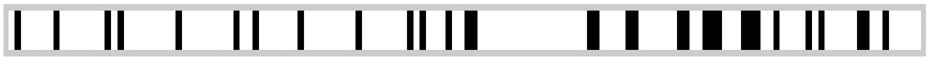


Fig. 8. Selected Features for MI approach

For RI, the best obtained classification accuracy is again 98.5%. The quality of the second approach over all runs is 98.5% and the reliability is 100%. Figure 9 shows classification accuracy plotted against the number of iterations.

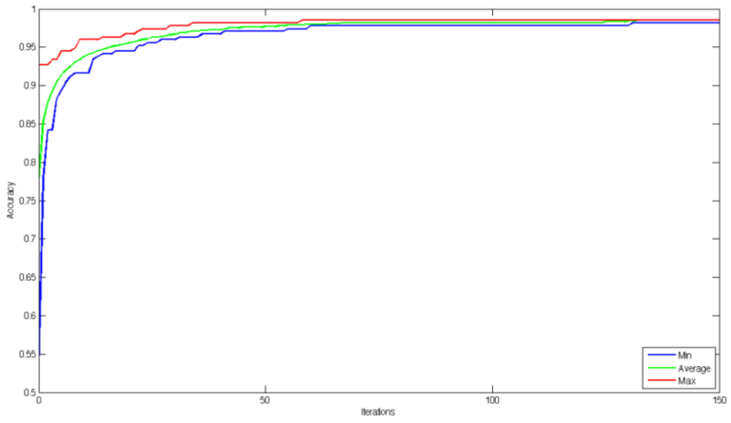


Fig. 9. Performance Graph for RI approach

Figure 10 present an illustration of best obtained feature set using RI approach (69 features out of 142).

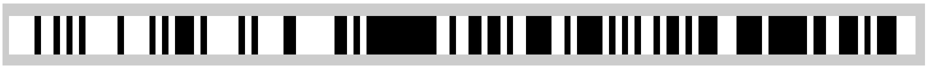


Fig. 10. Selected Features for RI approach

Both sets of results show that our method works well in improving the performance of gender recognition system. We showed that, using CHC genetic algorithm we can both reduce the number of features from 142 to 69 and improve the classification accuracy at the same time. The best result we can obtain simply and directly using our SVM classifier without feature subset selection is 94.36% classification accuracy. We thus believe that genetic algorithms hold much potential for feature subset selections for computer vision problems as well as for other classification problems. We plan on continuing this work to analyze features that were selected by the genetic algorithm. For example, we are working on comparing performance when using the top 69 PCA components for classification and should have results in time for the final version of this paper (if accepted).

5 Conclusion

In this paper we present a combinational method uses local binary pattern for feature extraction and genetic algorithms for feature selection. We also use a support vector machine for classification purposes. We use genetic algorithm for searching the space of feature subsets to find a set of features that maximizes gender classification accuracy. By evaluating our method on FERET database we showed that using genetic algorithms we can both increase classification accuracy and speed up testing phase at the same time. Starting with 142 features, the genetic algorithm reduces the feature set size by approximately half resulting in 98.5% accuracy with 100% reliability.

References

1. Mäkinen, E., Raisamo, R.: An experimental comparison of gender classification methods. Pattern Recognition Letters 29, 1544–1556 (2008)
2. Wu, B., Ai, H., Huang, C.: Real-time gender classification, pp. 498–503 (2003)
3. Wu, J., Smith, W.A.P., Hancock, E.R.: Semi-supervised Feature Selection for Gender Classification. In: Zha, H., Taniguchi, R.-i., Maybank, S. (eds.) ACCV 2009, Part II. LNCS, vol. 5995, pp. 23–33. Springer, Heidelberg (2010)
4. Xu, Z., Lu, L., Shi, P.: A hybrid approach to gender classification from face images. In: 19th International Conference on Pattern Recognition, ICPR 2008, pp. 1–4 (2008)
5. Alexandre, L.A.: Gender recognition: A multiscale decision fusion approach. Pattern Recognition Letters 31, 1422–1427 (2010)

6. Moghaddam, B., Yang, M.-H.: Gender Classification with Support Vector Machines. Presented at the Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (2000)
7. Balci, K., Atalay, V.: PCA for Gender Estimation: Which Eigenvectors Contribute? Presented at the Proceedings of the 16th International Conference on Pattern Recognition, ICPR 2002 (2002)
8. Lian, H.-C., Lu, B.-L.: Multi-view Gender Classification Using Local Binary Patterns and Support Vector Machines. In: Wang, J., Yi, Z., Żurada, J.M., Lu, B.-L., Yin, H. (eds.) ISNN 2006. LNCS, vol. 3972, pp. 202–209. Springer, Heidelberg (2006)
9. Iga, R., Izumi, K., Hayashi, H., Fukano, G., Ohtani, T.: A gender and age estimation system from face images. In: SICE 2003 Annual Conference, pp. 756–761 (2003)
10. Hosoi, S., Takikawa, E., Kawade, M.: Ethnicity estimation with facial images. In: Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 195–200 (2004)
11. Lian, H.-C., Lu, B.-L., Takikawa, E., Hosoi, S.: Gender Recognition Using a Min-Max Modular Support Vector Machine. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3611, pp. 438–441. Springer, Heidelberg (2005)
12. Fleming, M.K., Cottrell, G.W.: Categorization of faces using unsupervised feature extraction. IJCNN 2, 65–70 (1990)
13. Golomb, B., Lawrence, D., Sejnowski, T.: SexNet: A neural network identifies sex from human faces. Presented at the Proceedings of the 1990 Conference on Advances in Neural Information Processing Systems 3, Denver, Colorado, United States (1990)
14. Gutta, S., Huang, J.R.J., Jonathon, P., Wechsler, H.: Mixture of experts for classification of gender, ethnic origin, and pose of human faces. IEEE Transactions on Neural Networks 11, 948–960 (2000)
15. Moghaddam, B., Ming-Hsuan, Y.: Learning gender with support faces. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 707–711 (2002)
16. Burton, A.M., Bruce, V., Dench, N.: What's the difference between men and women? Evidence from facial measurement. Perception 22, 153–176 (1993)
17. XueMing, L., YiDing, W.: Improving generalization for gender classification. In: 15th IEEE International Conference on Image Processing, ICIP 2008, pp. 1656–1659 (2008)
18. Shakhnarovich, G., Viola, P., Moghaddam, B.: A unified learning framework for real time face detection and classification. In: Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 14–21 (2002)
19. Wu, B., Ai, H., Huang, C.: LUT-Based Adaboost for Gender Classification. In: Kittler, J., Nixon, M.S. (eds.) AVBPA 2003. LNCS, vol. 2688, pp. 104–110. Springer, Heidelberg (2003)
20. Baluja, S., Rowley, H.: Boosting Sex Identification Performance. International Journal of Computer Vision 71, 111–119 (2007)
21. Erno, M.K.: Evaluation of Gender Classification Methods with Automatically Detected and Aligned Faces. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 541–547 (2008)
22. Ojala, T., Pietikäinen, M., et al.: A comparative study of texture measures with classification based on featured distribution. Pattern Recognition 29(1), 51–59 (1996)
23. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)
24. Principal Component Analysis. In Wikipedia, `http://en.wikipedia.org/wiki/Principal_component_analysis` (retrieved April 29, 2012)

25. Makinen, E., Raisamo, R.: An experimental comparison of gender classification methods. Pattern Recognition Letters 29(10), 1544–1556 (2008)
26. Wu, B., Ai, H., Huang, C.: Real-time Gender Classification. In: Proceedings of SPIE 5286 Third International Symposium on Multispectral Image Processing and Pattern Recognition, pp. 498–503 (2003)
27. Phillips, P., Moon, H., Rizvi, S., Rauss, P.: The FERET evaluation methodology for face-recognition algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 1090–1104 (2000)
28. Viola, P., Jones, M.: Robust Real-time Object Detection. International Journal of Computer Vision (2002) (to appear)

Author Index