Advances in Intelligent Systems and Computing 378

Radek Matoušek *Editor*

Mendel 2015

Recent Advances in Soft Computing

Springer

Advances in Intelligent Systems and Computing

Volume 378

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

Radek Matoušek
Editor

Mendel 2015

Recent Advances in Soft Computing

Springer

Editor
Radek Matoušek
Faculty of Mechanical Engineering
Department of Applied Computer Science
Brno University of Technology
Brno
Czech Republic

ISSN 2194-5357 ISSN 2194-5365 (electronic)
Advances in Intelligent Systems and Computing
ISBN 978-3-319-19823-1 ISBN 978-3-319-19824-8 (eBook)
DOI 10.1007/978-3-319-19824-8

Library of Congress Control Number: 2015940969

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Preface

This proceedings book of the Mendel conference (http://www.mendel-conference.org) contains a collection of selected accepted papers which have been presented at this event in June 2015. The Mendel conference was held in the second largest city in the Czech Republic—Brno (http://www.brno.cz/en), which is a well-known university city. The Mendel conference was established in 1995 and is named after the scientist and Augustinian priest Gregor J. Mendel, who discovered the famous Laws of Heredity. In 2015 we are commemorating 150 years since Mendel's lectures, which he presented in Brno during February and March 1865.

The main aim of the Mendel conference is to create a regular possibility for students, academics and researchers to exchange their ideas on novel research methods as well as to establish new friendships on a yearly basis. The scope of the conference includes many areas of Soft Computing including: *Genetic Algorithms, Genetic Programming, Grammatical Evolution, Differential Evolution, Evolutionary Strategies, Hybrid and Distributed Algorithms, Probabilistic Metaheuristics, Swarm Intelligence, Ant Colonies, Artificial Immune Systems, Computational Intelligence, Evolvable Hardware, Chemical Evolution, Fuzzy Logic, Bayesian methods, Neural Networks, Data mining, Multi-Agent Systems, Artificial Life, Self-organization, Chaos, Complexity, Fractals, Image Processing, Computer Vision, Control Design, Robotics, Motion Planning, Decision-making, Metaheuristic Optimization Algorithms, Intelligent Control, Bio-Inspired Robots, Computer Vision and Intelligent Image Processing*.

Soft computing is a formal area of computer science and an important part in the field of artificial intelligence. Professor Lotfi A. Zadeh introduced the first definition of soft computing in the early 1990s: "Soft computing principles differs from hard (conventional) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation". The role model for soft computing is the human mind and its cognitive abilities. The guiding principle of soft computing can be specified as follows: exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability and robustness at a low solution cost.

The main constituents of soft computing include fuzzy logic, neural computing, evolutionary computation, machine learning, and probabilistic reasoning, whereby probabilistic reasoning contains belief networks as well as chaos theory. It is important to say that soft computing is not a random mixture of solution approaches. Rather, it is a collection of methodologies in which each part contributes in a distinct way to address a certain problem in its specific domain. From this point of view, the set of soft computing methodologies can be seen as complementary rather than competitive. Furthermore, soft computing is an important component for the emerging field of contemporary artificial intelligence.

Image processing is a complex process, in which image processing routines and domain-dependent interpretation steps often alternate. In many cases, image processing has to be extensively intelligent regarding the tolerance of imprecision and uncertainty. A typical application of intelligent image processing is computer vision in robotics.

Bio-inspired robotics is a fairly new sub-category of robotics. It is about learning concepts from nature and applying them to the design of real-world engineered systems. More specifically, this field is about making robots that are inspired by biological systems. It includes difficult mathematical theories as well as simple central pattern generators (CPG) based on biological neural networks.

This proceedings book contains three chapters which present recent advances in soft computing including intelligent image processing and bio-inspired robotics. The accepted selection of papers was rigorously reviewed in order to maintain the high quality of the conference. Based on the topics of accepted papers the proceedings book consists of three Parts: Part I: *Evolutionary Computing, Swarm Intelligence*, Part II: *Neural Networks, Self-organization, and Machine Learning*, and Part III: *Intelligent Image Processing, and Bio-Inspired Robotics*.

We would like to thank the members of the International Program Committees and Reviewers for their hard work. We believe that the Mendel conference represents a high standard conference in the domain of Soft Computing. Mendel 2015 enjoyed outstanding keynote lectures by distinguished guest speakers: Julian Miller (United Kingdom), Swagatam Das (India), Wolfram Wiesemann (United Kingdom), Eva Matalová (Czech Republic) and René Lozi (France).

Particular thanks go to the conference organizers and main sponsors as well. In 2015 the conference is organized under the auspices of the Brno City Mayor Petr Vokřál and Brno University of Technology with support from WU Vienna University of Economics and Business, and University of Vaasa. The conference sponsors are Humusoft Ltd. (International reseller and developer for MathWorks, Inc., U.S.A.), B&R automation Ltd. (multinational company, specialised in factory and process automation software), and Autocont Ltd. (private Czech company that operates successfully in the area of ICT).

We would like to thank all contributing authors, as well as the members of the International Program Committees, the Local Organizing Committee and the Executive Organizing Committee namely Ronald Hochreiter and Jouni Lampinen for their hard and highly valuable work. Their work has definitely contributed to the success of the Mendel 2015 conference.

Radek Matoušek

Organization

International Program Committee and Reviewers Board

Executive Organizing Committee and Local Organizers

Local Organizers

Contents

Part I Evolutionary Computing, Swarm Intelligence

Avoidance Strategies in Particle Swarm Optimisation 3
Karl Mason and Enda Howley

Two-Stage Stochastic Programming for Transportation Network Design Problem 17
Dušan Hrabec, Pavel Popela, Jan Roupec, Jan Mazal and Petr Stodola

A Novel Hyper-Heuristic Approach for Channel Assignment in Cognitive Radio Networks 27
Emrullah Gazioglu, A. Sima Etaner-Uyar and Berk Canberk

Improved Bees Algorithm for Protein Structure Prediction Using AB Off-Lattice Model 39
Nanda Dulal Jana, Jaya Sil and Swagatam Das

Limited Randomness Evolutionary Strategy Algorithm 53
Tomas Brandejsky

Data Mining Application on Complex Dataset from the Off-Grid Systems 63
Tomas Vantuch, Jindrich Stuchly, Stanislav Misak and Tomas Burianek

Population Size Reduction in Particle Swarm Optimization Using Product Graphs 77
Iztok Fister Jr., Aleksandra Tepeh, Janez Brest and Iztok Fister

Cooperation of Evolutionary Algorithms: A Comparison of Several Hierarchical Models 89
Radka Poláková and Josef Tvrdík

A Switched Parameter Differential Evolution for Large Scale Global Optimization – Simpler May Be Better 103
Swagatam Das, Arka Ghosh and Sankha Subhra Mullick

The Initial Study on the Potential of Super-Sized Swarm in PSO 127
Michal Pluhacek, Roman Senkerik and Ivan Zelinka

A Levy Interior Search Algorithm for Chaotic System Identification 137
Rushi Jariwala, Rohan Patidar and Nithin V. George

Hybridization of Adaptivity and Chaotic Dynamics for Differential Evolution 149
Roman Senkerik, Michal Pluhacek, Donald Davendra, Ivan Zelinka, Zuzana Kominkova Oplatkova and Jakub Janostik

A Minimisation of Network Covering Services in a Threshold Distance 159
Miloš Šeda and Pavel Šeda

CUDA-based Analytic Programming by Means of SOMA Algorithm 171
Lumir Kojecky and Ivan Zelinka

Computing Trading Strategies Based on Financial Sentiment Data Using Evolutionary Optimization 181
Ronald Hochreiter

Part II Neural Networks, Self-organization, Machine Learning

An Approach to ANFIS Performance 195
Stepan Dalecky and Frantisek V. Zboril

Values and Bayesian Probabilities of Mental States from BSDT PL Analysis of Memory ROCs 207
Petro Gopych and Ivan Gopych

Scaled Conjugate Gradient Learning for Complex-Valued Neural Networks 221
Călin-Adrian Popa

Off-Grid Parameters Analysis Method Based on Dimensionality Reduction and Self-organizing Map 235
Tomas Burianek, Tomas Vantuch, Jindrich Stuchly and Stanislav Misak

Matrix-Valued Neural Networks. 245
Călin-Adrian Popa

A Feature Clustering Approach for Dimensionality Reduction and Classification 257
Kotte VinayKumar, R. Srinivasan and Elijah Blessing Singh

An Application of ANNs Method for Solving Fractional Fredholm Equations 269
Ahmad Jafarian and S. Measoomy Nia

Solving Circle Packing Problem by Neural Gas. 281
Jiri Pospichal

Cost Functions Based on Different Types of Distance Measurements for Pseudo Neural Network Synthesis 291
Zuzana Kominkova Oplatkova and Roman Senkerik

Part III Intelligent Image Processing, Bio-Inspired Robotics

Labyrinth Arrangement Analysis Based on Image Processing 305
Pavel Škrabánek

Neural Network Approach to Image Steganography Techniques 317
Robert Jarušek, Eva Volna and Martin Kotyrba

Rough-Fuzzy Collaborative Multi-level Image Thresholding: A Differential Evolution Approach. 329
Sujoy Paul, Shounak Datta and Swagatam Das

Fusion of 3D Model and Uncalibrated Stereo Reconstruction. 343
Jan Klecka and Karel Horak

Visual SLAM Based on Phase Correlation and Particle Filters 353
Michal Růžička and Petr Mašek

Notes on Differential Kinematics in Conformal Geometric Algebra Approach 363
Jaroslav Hrdina and Petr Vašík

Trident Snake Control Based on Conformal Geometric Algebra 375
Aleš Návrat and Radomil Matoušek

Author Index 387

Part I
Evolutionary Computing, Swarm Intelligence

Avoidance Strategies in Particle Swarm Optimisation

Karl Mason and Enda Howley

Abstract Particle swarm optimisation (PSO) is an optimisation algorithm in which particles traverse a problem space moving towards promising locations which either they or their neighbours have previously visited. This paper presents a new PSO variant with the Avoidance of Worst Locations (AWL). This variation was inspired by animal behaviour. In the wild, an animal will react to negative stimuli as well as positive, e.g. an animal looking for food will also be conscious of danger. PSO AWL enables particles to remember previous poor solutions as well as good. As a result, the particles change the way they move and avoid known bad areas. Balancing the influence of these poor locations is vital. The research in this paper found that a small influence from bad locations on the particles leads to a significant improvement on overall performance when compared to the standard PSO. When compared to previous implementations of worst location memory, PSO AWL demonstrates vast improvements.

1 Introduction

Particle swarm optimisation (PSO) can be classified as a swarm intelligence approach to optimisation. The PSO algorithm was initially proposed by Kennedy et al. in 1995 [7]. The algorithm came about as a result of modelling the flight of a flock of birds. The algorithm functions by inserting particles randomly into a problem space. These particles evaluate their position as they move around. The particles change their velocity based on their own personal best position and that of their neighbourhood. This gives the particle both a cognitive and a social manner of behaving. Most research into PSO focuses on either improving its performance [10] or applying it to real world problems [11]. Improvements on performance focus on areas such as

K. Mason · E. Howley (✉)
Discipline of Information Technology,
National University of Ireland Galway, Galway, Ireland
e-mail: enda.howley@nuigalway.ie

K. Mason
e-mail: k.mason2@nuigalway.ie

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_1

initialisation [4], topology [8, 9], particle memory [2, 5] and boundary conditions [14]. Currently the algorithm functions as a result of particles converging on the best location. This paper extends the particles' equation of motion to take account of negative stimuli, similar to how an animal would avoid a predator in the wild. The particles will still converge on the best solution while also avoiding the worst. This paper aims to answer the following three questions:

1. Can the PSO be significantly improved by incorporating the memory of bad previous locations?
2. How should the worst locations be implemented?
3. How does the proposed PSO perform when compared to previous similar implementations?

The rest of the paper will be structured as follows: Sect. 2 will define a standard PSO and will highlight the main research conducted in the area of influencing particles with the memory of bad previous locations. Section 3 will describe the proposed PSO with Avoidance of Worst Locations and how it will be evaluated. Section 4 will present performance results of PSO AWL. In Sect. 5, the results from the Sect. 4 will be discussed. Lastly, we conclude our paper by discussing the implications of our results and our plans for future work.

2 Background

2.1 Standard Particle Swarm Optimisation

As a result of the research conducted into PSO, certain features have been widely adopted as standard. At a time t, each particle has a position x_t and a velocity v_t [7]. The particles move through a problem space evaluating positions. Particles move towards positions with the best fitness scores using the following equations:

$$v_{t+1} = \chi(v_t + r_1 c_1(pbest_t - x_t) + r_2 c_2(gbest_t - x_t) \tag{1a}$$

$$x_{t+1} = x_t + v_t \tag{1b}$$

In these equations, r_1 and r_2 are random numbers between 0 and 1, c_1 and $c_2 = 2.05$ are acceleration coefficients. Pbest refers to a particles previous best location while gbest refers to the best previous location within a particle's neighbourhood. χ is a constriction factor and is calculated by:

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \tag{2a}$$

$$\varphi = c_1 + c_2 \tag{2b}$$

$\chi \approx 0.72984$ has been derived to give the best convergence when $c_1 = c_2 = 2.05$ [3]. The equations above are the accepted standard [1] and will be used as a benchmark to which our experimental results will be compared to.

2.2 *PSO and Worst Locations*

The idea of using the worst locations within the problem space to influence the motion of the particles was first introduced by Yang et al. in 2005 [15]. This early adaptation implemented only the worst locations as a means to dictate the particles' motion.

In 2007 Selvakumar et al. proposed their new PSO (NPSO) which updated a particle's velocity using both the particle's best and worst location [12]. The NPSO was the first attempt to merge the PSO of Yang et al. with the standard PSO. Below is the velocity update equation used by Selvakumar et al.:

$$\begin{aligned} v_{t+1} &= \omega v_t + r_1 c_1 (pbest_t - x_t) + r_2 c_2 (x_t - pworst_t) \\ &\quad + r_3 c_3 (gbest_t - x_t) \end{aligned} \tag{3}$$

In 2014 Jayabarathi et al. extended this velocity update equation to also include the worst location within a particle's neighbourhood [6]. Below is their velocity update equation:

$$\begin{aligned} v_{t+1} &= \omega v_t + r_1 c_1 (pbest_t - x_t) + r_2 c_2 (gbest_t - x_t) \\ &\quad + r_3 c_3 (x_t - pworst) + r_4 c_4 (x_t - gworst) \end{aligned} \tag{4}$$

3 PSO with Avoidance of Worst Locations

Here we introduce a new velocity update equation which incorporates the previous worst location of the particle and the previous worst location within its neighbourhood. The proposed velocity update equation ensures that the extra velocity given from the worst locations is always in the direction of the best locations. The new velocity update equation also ensures that the extra velocity given to a particle is maximised when the particle is at its worst location and decreases as the particle moves away. This is not the case in the implementations described in Sect. 2.2. Modifying the velocity update equation in this way encourages the particles towards good solutions and away from known bad solutions.

3.1 Proposed Equation

To implement this extra velocity from the worst locations, the velocity update equation has been changed as follows:

$$v_{t+1} = \chi(v_t + t_1 + t_2 + t_3 + t_4) \tag{5a}$$

$$t_1 = r_1 c_1 (pb_t - x_t) \tag{5b}$$

$$t_2 = r_2 c_2 (gb_t - x_t) \tag{5c}$$

$$t_3 = r_3 c_3 \left(\frac{t_1}{(1 + |x_t - pw_t|)}\right) \tag{5d}$$

$$t_4 = r_4 c_4 \left(\frac{t_2}{(1 + |x_t - gw_t|)}\right) \tag{5e}$$

The t_1 and t_2 terms are the same as in the standard PSO. The new t_3 and t_4 terms take into account the worst location by using the difference between the particles current position and the worst position. The absolute value of this difference is taken to ensure the extra velocity given to the particle from t_3 and t_4 is in the same direction as t_1 and t_2 respectively.

Within t_3 and t_4, t_1 and t_2 are divided by $|x_t - pw_t|$ and $|x_t - gw_t|$ respectively, so that the extra velocity given to the particle is at its highest at the worst position and decreases as the particle moves away.

Finally, to avoid dividing by 0 when the particle is at the worst location and to limit the extra velocity given to the particle from t_3 and t_4, 1 is added to $|x_t - pw_t|$ and $|x_t - gw_t|$. All r values are random numbers between 0 and 1 while all c values are the acceleration coefficients.

3.2 Selection of Acceleration Coefficients

Choosing the correct parameters for the PSO is critical to its performance. To do this, a series of parameter sweeps were conducted over the different values of c_1, c_2, c_3 and c_4 with $\chi = 0.72984$. These parameter sweeps are outlined in the table below (Table 1):

Table 1 Acceleration coeffients parameter sweeps

Parameter sweep	Initial c_1,c_2	Final c_1,c_2	Initial c_3,c_4	Final c_3,c_4	Incremental change
1	1.025	2.05	1.025	0	0.1025
2	1.025	1.5375	1.025	1.5375	0.05125
3	1.025	1.025	0	3.075	0.205
4	1.5375	1.5375	0	3.075	0.205
5	2.05	2.05	0	3.075	0.205

3.3 Selection of Constriction Value

A final parameter sweep was then conducted over the set of c values which gave the most promising results from Sect. 3.2. This parameter sweep kept the c values constant while varying the constriction value χ from 0.4 up to 0.9 in increments of 0.05. The aim of this last parameter sweep was to investigate which χ value yields the optimum performance.

3.4 Evaluation

All of the parameter sweeps above were conducted on PSO AWL with three different topologies: global, ring and von Neumann. These were carried out on 7 benchmark functions: Sphere, Rosenbrock, Ackley, Griewank, Rastrigin, Schaffer 2D and Griewank 10D. The best performing topology and set of parameters from the first 7 functions was further tested on the 25 functions outlined in [13]. This set of results was compared to the standard PSO (SPSO) with global, ring and von Neumann topologies. These functions were all evaluated in 30 dimensions with the exception of the Schaffer 2D function and the Griewank 10D which were tested in 2 and 10 dimensions respectively. The results from each function were averaged over 25 runs with each run consisting of 10,000 iterations and a swarm size of 50 particles. When a particle reaches a boundary, it is reflected back into the problem space except for problems 14 and 32 which have optima outside of the initialisation region. The PSO AWL was then compared with the PSO of Selvakumar et al. (NPSO1) [12] and the PSO of Jayabarathi et al. (NPSO2) [6].

4 Results

4.1 Parameter Selection

The parameters c_1 and $c_2 = 1.845$ and c_3 and $c_4 = 0.205$ were selected as the acceleration coefficients which gave the best overall performance. These values were used in the final parameter sweep over the constriction values outlined in Sect. 3.3. This last parameter sweep showed the best value for the constriction to be $\chi = 0.72984$ as is the case for the standard PSO. Of the three topologies tested, the von Neumann topology performed the best overall. This set of parameters, with a von Neumann topology, was then evaluated on 32 functions.

4.2 PSO AWL Vs Standard PSO

The PSO AWL with these parameters was then compared to the standard PSO on the 32 functions. The SPSO was implemented on three different topologies: global, ring and von Neumann. Table 2 shows the performance results of each PSO and the number of functions where each PSO performed exclusively the best and worst. The PSO AWL was then compared individually against each SPSO. Table 2 gives the number of functions in which the PSO AWL performed significantly better, worse and equal compared to each other PSO. In order to determine if the difference in results were significant, the Wilcoxon signed rank test was conducted to compare the PSO AWL with the SPSO on each function. A p-value of 0.05 was used to determine significant differences in performance. Values under 10^{-14} were then rounded down to 0 when displayed in Table 2.

4.3 PSO AWL Vs Previous Implementations

The NPSO1 and NPSO2 gave similar performance when evaluated with global, ring and von Neumann topologies. A global topology was selected for NPSO1 and NPSO2 when comparing with the PSO AWL as it was the best performing of the three topologies and was the topology used in their origional implementations [6, 12]. Table 3 shows how each PSO performed on all 32 functions. The same comparisons and rounding was conducted for Table 3 as was conducted for Table 2.

5 Discussion

5.1 Parameter Selection

The results show that an improvement was made to the performance of the PSO. The parameter sweeps demonstrate that higher values of c_3 and c_4 lead to improved performance on functions with many local minima such as Rastrigin and Schaffer. Conversely lower values of c_3 and c_4 will improve performance on unimodal functions such as Sphere and Rosenbrock. These lower values have a smaller influence on the particles' velocity and performed best overall. The optimal acceleration coefficients are c_1 and $c_2 = 1.845$ and c_3 and $c_4 = 0.205$. This answers research question 2. These values for c_1, c_2, c_3 and c_4 yield a value of $\varphi = 4.1$, if Eq. 2b from Sect. 2.1 is extended to include the new acceleration coefficients. When inserting $\varphi = 4.1$ into Eq. 2a from Sect. 2.1, the constriction value is calculated to be $\chi = 0.72984$. This correlates with the result of the final parameter sweep in which $\chi = 0.72984$ was found to be the optimal constriction value. Using these parameters for PSO AWL gave the best balance between exploration with convergence. These parameters were used to compare PSO AWL to the standard PSO.

Table 2 PSO AWL vs standard PSO

Function	PSO AWL mean	PSO AWL std	Global mean	Global std	Ring mean	Ring std	von Neumann mean	von Neumann std
Sphere	0.00E+000	0.00E+000	**0.00E+000**	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
Rosenbrock	6.68E+000	1.50E+000	**2.15E+000**	2.26E+000	1.45E+001	2.39E+000	1.04E+001	1.96E+000
Ackley	1.28E-014	0.00E+000	1.03E+000	9.21E-001	1.51E-014	0.00E+000	**1.27E-014**	0.00E+000
Griewank	3.45E-003	5.35E-003	1.39E-002	2.45E-002	**1.18E-003**	2.71E-003	7.19E-003	1.10E-002
Rastrigin	4.38E+001	1.01E+001	4.82E+001	1.30E+001	4.95E+001	1.02E+001	**3.86E+001**	9.60E+000
Schaffer2D	1.20E-004	3.20E-004	5.52E-004	4.89E-004	1.99E-004	3.93E-004	**3.94E-005**	1.93E-004
Griewank10D	2.79E-002	2.05E-002	7.04E-002	2.98E-002	**2.18E-002**	1.27E-002	3.22E-002	1.51E-002
f1	-4.50E+002	2.78E-014	-4.50E+002	1.14E-014	-4.50E+002	3.94E-014	-4.50E+002	3.01E-014
f2	-4.50E+002	1.07E-013	-4.50E+002	1.53E-013	-4.50E+002	2.30E-004	-4.50E+002	1.72E-008
f3	-4.50E+002	2.54E-014	-4.50E+002	2.27E-014	-4.50E+002	3.77E-014	-4.50E+002	1.61E-014
f4	-4.14E+002	5.37E+001	**-4.49E+002**	3.17E+000	2.09E+003	2.16E+003	-3.82E+002	1.23E+002
f5	**2.23E+004**	1.44E+004	3.03E+004	1.55E+004	3.05E+004	1.04E+004	2.39E+004	1.23E+004
f6	**3.94E+002**	5.85E+000	3.98E+002	2.86E+001	3.97E+002	6.96E+000	4.14E+002	3.77E+001
f7	-1.80E+002	2.94E-002	-1.80E+002	2.57E-002	-1.80E+002	2.25E-002	**-1.80E+002**	1.28E-002
f8	**-1.19E+002**	7.43E-002	-1.19E+002	9.03E-002	-1.19E+002	9.80E-002	-1.19E+002	5.77E-002
f9	-2.77E+002	1.29E+001	-2.75E+002	1.51E+001	-2.73E+002	1.16E+001	**-2.86E+002**	1.12E+001
f10	**-2.64E+002**	1.88E+001	-2.21E+002	3.19E+001	-2.51E+002	1.75E+001	-2.58E+002	1.84E+001
f11	1.17E+002	2.29E+000	**1.15E+002**	3.25E+000	1.17E+002	1.86E+000	1.16E+002	2.29E+000
f12	**5.90E+003**	7.19E+003	1.81E+004	2.91E+004	6.09E+003	4.71E+003	6.72E+003	6.02E+003
f13	-1.25E+002	1.45E+000	-1.24E+002	3.09E+000	-1.23E+002	1.99E+000	**-1.25E+002**	9.83E-001
f14	**-2.90E+002**	7.35E-001	-2.90E+002	7.89E-001	-2.89E+002	5.23E-001	-2.90E+002	7.42E-001
f15	1.20E+002	0.00E+000	1.44E+002	1.18E+002	1.20E+002	0.00E+000	1.20E+002	0.00E+000

(continued)

Table 2 (continued)

Function	PSO AWL mean	PSO AWL std	Global mean	Global std	Ring mean	Ring std	von Neumann mean	von Neumann std
f16	1.20E+002	0.00E+000	1.20E+002	0.00E+000	1.20E+002	2.13E-014	1.20E+002	0.00E+000
f17	1.20E+002	0.00E+000	1.20E+002	1.17E-014	1.20E+002	0.00E+000	1.20E+002	0.00E+000
f18	1.39E+002	2.30E+002	5.53E+002	1.09E+001	1.39E+002	2.29E+002	**1.17E+002**	2.14E+002
f19	**2.63E+002**	1.72E+000	5.04E+002	2.62E+002	2.71E+002	3.15E+001	2.74E+002	2.87E+001
f20	5.50E+002	5.40E+000	6.07E+002	1.55E+002	**4.88E+002**	1.66E+002	5.26E+002	1.05E+002
f21	**6.33E+002**	2.84E+002	1.23E+003	1.76E+002	9.61E+002	5.24E+001	7.03E+002	2.82E+002
f22	**1.00E+003**	1.51E+002	1.22E+003	1.01E+002	1.09E+003	5.32E+001	1.03E+003	6.25E+001
f23	**7.29E+002**	2.79E+002	1.18E+003	1.44E+002	9.83E+002	8.00E+001	9.17E+002	1.24E+002
f24	**8.70E+002**	3.91E+002	1.30E+003	5.26E+001	1.26E+003	6.23E+001	1.02E+003	3.61E+002
f25	6.71E+002	4.19E+002	1.33E+003	6.09E+001	1.27E+003	8.88E+001	**4.74E+002**	3.31E+002
Best	11		4		3		8	
Worst	1		16		10		1	
PSO AWL statistically better			17		17		9	
PSO AWL statistically worse			3		1		4	
PSO AWL statistically equal			12		14		19	

Table 3 PSO AWL vs NPSO1 & NPSO2

Function	PSO AWL mean	PSO AWL std	NPSO1 mean	NPSO1 std	NPSO2 mean	NPSO2 std
Sphere	**0.00E+000**	0.00E+000	2.79E+001	2.62E+000	7.77E+001	7.62E+000
Rosenbrock	**6.68E+000**	1.50E+000	7.70E+002	1.62E+002	1.90E+003	2.22E+002
Ackley	**1.28E-014**	0.00E+000	1.47E+001	6.15E-001	1.89E+001	2.77E-001
Griewank	**3.45E-003**	5.35E-003	9.76E+001	1.27E+001	2.67E+002	3.11E+001
Rastrigin	**4.38E+001**	1.01E+001	2.64E+002	1.98E+001	3.00E+002	8.27E+000
Schaffer2D	1.20E-004	3.20E-004	**1.33E-007**	2.32E-007	5.33E-004	3.85E-004
Griewank10D	**2.79E-002**	2.05E-002	4.94E+000	8.07E-001	1.85E+001	5.37E+000
f1	**-4.50E+002**	2.78E-014	4.08E+003	9.21E+002	1.77E+004	3.51E+003
f2	**-4.50E+002**	1.07E-013	7.65E+003	1.90E+003	2.98E+004	5.11E+003
f3	**-4.50E+002**	2.54E-014	3.95E+003	7.49E+002	1.72E+004	2.29E+003
f4	**-4.14E+002**	5.37E+001	9.05E+003	2.25E+003	3.50E+004	5.64E+003
f5	**2.23E+004**	1.44E+004	2.94E+004	4.09E+003	8.19E+004	8.63E+003
f6	**3.94E+002**	5.85E+000	1.37E+008	5.01E+007	2.58E+009	5.49E+008
f7	**-1.80E+002**	2.94E-002	1.62E+002	1.03E+002	8.61E+003	2.01E+003
f8	**-1.19E+002**	7.43E-002	-1.19E+002	4.72E-002	-1.19E+002	5.98E-002
f9	**-2.77E+002**	1.29E+001	-1.28E+002	1.27E+001	-7.31E+001	1.20E+001
f10	**-2.64E+002**	1.88E+001	-9.31E+001	1.43E+001	3.82E+000	2.43E+001
f11	**1.17E+002**	2.29E+000	1.23E+002	2.53E+000	1.29E+002	8.95E-001
f12	**5.90E+003**	7.19E+003	2.63E+005	5.29E+004	7.79E+005	1.44E+005
f13	**-1.25E+002**	1.45E+000	7.25E+001	8.46E+001	2.89E+004	1.84E+004
f14	**-2.90E+002**	7.35E-001	-2.89E+002	5.01E-001	-2.88E+002	2.24E-001

(continued)

Table 3 (continued)

Function	PSO AWL mean	PSO AWL std	NPSO1 mean	NPSO1 std	NPSO2 mean	NPSO2 std
f15	**1.20E+002**	0.00E+000	1.20E+003	2.39E+001	1.37E+003	1.90E+001
f16	**1.20E+002**	0.00E+000	1.28E+003	2.23E+001	1.45E+003	2.63E+001
f17	**1.20E+002**	0.00E+000	1.34E+003	3.14E+001	1.50E+003	2.93E+001
f18	**1.39E+002**	2.30E+002	1.20E+003	3.09E+001	1.41E+003	4.54E+001
f19	**2.63E+002**	1.72E+000	1.20E+003	2.38E+001	1.42E+003	3.42E+001
f20	**5.50E+002**	5.40E+000	1.03E+003	2.15E+001	1.27E+003	2.45E+001
f21	**6.33E+002**	2.84E+002	1.66E+003	5.29E+001	1.90E+003	5.07E+001
f22	**1.00E+003**	1.51E+002	1.52E+003	4.10E+001	1.66E+003	3.19E+001
f23	**7.29E+002**	2.79E+002	1.65E+003	4.14E+001	1.90E+003	4.63E+001
f24	**8.70E+002**	3.91E+002	1.48E+003	3.64E+001	1.58E+003	3.17E+001
f25	**6.71E+002**	4.19E+002	1.39E+003	3.81E+001	1.94E+003	3.47E+001
Best	31		1		0	
Worst	0		0		32	
PSO AWL statistically better			31		32	
PSO AWL statistically worse			1		0	
PSO AWL statistically equal			0		0	

5.2 PSO AWL Vs Standard PSO

When compared to the standard PSO, PSO AWL performs best most often and worst least often. This is illustrated in Table 2. When compared individually against the global SPSO, the PSO AWL performs statistically better on 17 functions and worse on 3 functions. Similarly, the PSO AWL performs statistically better on 17 functions and worse on 1 function when compared to the ring SPSO. The PSO AWL is closer in performance to the von Neumann SPSO due to each being implemented with a von Neumann topology. Statistically, the PSO AWL performs better on 9 functions and worse on 4 functions. Overall it performs best on 11 functions compared to 8 for the von Neumann SPSO, thus showing an improvement on the PSO and addressing research question 1.

5.3 Function Type Performance

The PSO AWL performs better on functions with certain properties. This is highlighted in Table 2. Functions f15 through to f25 are composite functions which have many local minima and are more complex. For these 10 functions the PSO AWL performs best on 5, and the same on 3. The PSO AWL never performs the worst on any of these 11 functions. This demonstrates that the PSO AWL is better able to optimise complex functions than the SPSO.

Functions where the optimum is at the boundary favour the PSO AWL. Of the 3 functions where this is the case (f5, f8 and f20), the PSO AWL performs best on 2 (f5 and f8). This is due to the particles having less velocity as they move away from the worst position. This better enables them to converge on an optimum at the boundary than faster moving particles of the SPSO.

The performance of PSO AWL improves on problems where the optimum is shifted. The PSO AWL has average performance on the initial 7 functions which have their optima at the origin. For the subsequent 25 functions, the PSO AWL performs exclusively the best on 11 functions, equally optimal on 6 functions, average on 7 functions and the worst on 1 function. The optimum is shifted away from the origin in each of these functions. PSO AWL is the second best performing PSO on the rosenbrock function but is the best performing PSO on function f6 (shifted rosenbrock). This further illustrates a preference for shifted functions. Similar results are seen for rotated functions. PSO AWL performs second best on f9 but performs best on f10 (f10 is f9 rotated).

PSO AWL performs worse on functions which have no boundaries. This is due to the particles having less velocity as they move away from the worst position. This causes the particles to converge faster. If the optimum is far outside of the initialisation region, the particles will not have enough velocity to reach this optimum and converge on a local minimum as a result. This is the case for functions f7 and f25. Function f7 is the only function which the PSO AWL performs worst on. On function f25, PSO AWL performs worse than the von Neumann SPSO.

5.4 PSO AWL Vs Previous Implementations

PSO AWL has superior performance compared to both previous implementations that include worst fitness locations. This gives a clear answer to research question 3. Table 3 outlines the performance of each PSO. PSO AWL performed best out of the 3 PSO algorithms on 31 out of the 32 functions tested. On the function where PSO AWL didn't perform best, it was not the worst performing PSO algorithm. The results show that PSO AWL performs better than previous attempts to take into account of worst fitness locations.

6 Conclusion

The experiments conducted in this paper have addressed the three research questions asked in Sect. 1. Consequently it is clear that:

1. Using poor locations can significantly improve upon the SPSO, as demonstrated by the PSO AWL.
2. The best implementation of worst locations is to influence the particles in a subtle way. A small influence from the location of the worst positions is best so that the particles' convergence is not negatively affected. It is vital to the success of the PSO that t_1 and t_2 are still the dominating factors in determining the motion of the particles.
3. The PSO AWL improves upon previous attempts of implementing worst fitness locations. This is because the effects of the PSO AWL velocity update equation are more subtle than previous attempts which improves convergence.

This paper has examined a novel and very promising extension to existing PSO research. Therefore, in future research, it is hoped to explore other possible velocity update equations that might better implement worst fitness locations than PSO AWL. It is also hoped that further insights can be gained by refining the influence that the worst locations have on the particles in order to better balance exploration and convergence.

References

1. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: Swarm Intelligence Symposium, SIS 2007, pp. 120–127. IEEE (2007)
2. Broderick, I., Howley, E.: Particle swarm optimisation with enhanced memory particles. In: Dorigo, M., Birattari, M., Garnier, S., Hamann, H., Montes de Oca, M., Solnon, C., Stützle, T. (eds.) ANTS 2014. LNCS, vol. 8667, pp. 254–261. Springer, Heidelberg (2014)
3. Clerc, M., Kennedy, J.: The particle swarm–explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 58–73 (2002)

4. Helwig, S., Wanka, R.: Theoretical analysis of initial particle swarm behavior. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 889–898. Springer, Heidelberg (2008)
5. X. Hu, Eberhart, R.C., Shi, Y.: Particle swarm with extended memory for multiobjective optimization. In: Swarm Intelligence Symposium, SIS'03. Proceedings of the 2003 IEEE, pp. 193–197. IEEE (2003)
6. Jayabarathi, T., Kolipakula, R.T., Krishna, M.V., Yazdani, A.: Application and comparison of PSO, its variants and hde techniques to emission/economic dispatch. Arab. J. Sci. Eng. **39**(2), 967–976 (2014)
7. Kennedy, J.: Particle swarm optimization. In: Encyclopedia of Machine Learning, pp. 760–766. Springer (2010)
8. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, vol. 2, pp. 1671–1676. IEEE (2002)
9. Liu, H., Howely, E., Duggan, J.: Particle swarm optimisation with gradually increasing directed neighbourhoods. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 29–36. ACM (2011)
10. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intell. **1**(1), 33–57 (2007)
11. Robinson, J., Rahmat-Samii, Y.: Particle swarm optimization in electromagnetics. IEEE Trans. Antennas Propag. **52**(2), 397–407 (2004)
12. Selvakumar, A.I., Thanushkodi, K.: A new particle swarm optimization solution to nonconvex economic dispatch problems. IEEE Trans. Power Syst. **22**(1), 42–51 (2007)
13. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S.: Problem defnitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL Report 2005005, 2005 (2005)
14. Xu, S., Rahmat-Samii, Y.: Boundary conditions in particle swarm optimization revisited. IEEE Trans. Antennas Propag. **55**(3), 760–765 (2007)
15. Yang, C., Simon, D.: A new particle swarm optimization technique. In: 18th International Conference on Systems Engineering, ICSEng 2005, pp. 164–169. IEEE (2005)

Two-Stage Stochastic Programming for Transportation Network Design Problem

Dušan Hrabec, Pavel Popela, Jan Roupec, Jan Mazal and Petr Stodola

Abstract The transportation network design problem is a well-known optimization problem with many practical applications. This paper deals with demand-based applications, where the operational as well as many other decisions are often made under uncertainty. Capturing the uncertain demand by using scenario-based approach, we formulate the two-stage stochastic mixed-integer linear problem, where the decision, which is made under uncertainty, of the first-stage program, is followed by the second-stage decision that reacts to the observed demand. Such a program may reach solvability limitations of algorithms for large scale real world data, so we refer to the so-called hybrid algorithm that combines a traditional optimization algorithm and a suitable genetic algorithm. The obtained results are presented in an explanatory form with the use of a sequence of figures.

Keywords Two-stage stochastic programming · Scenario-based approach · Transportation model · Network design problem · Genetic algorithm · Hybrid algorithm

D. Hrabec (✉) · P. Popela · J. Roupec
Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, 616 69 Brno, Czech Republic
e-mail: hrabec.dusan@gmail.com

P. Popela
e-mail: pavel.popela@fme.vutbr.cz

J. Roupec
e-mail: jan.roupec@fme.vutbr.cz

J. Mazal · P. Stodola
Faculty of Military Leadership, University of Defence in Brno, Kounicova 65, 662 10 Brno, Czech Republic
e-mail: jan.mazal@unob.cz

P. Stodola
e-mail: petr.stodola@unob.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_2

1 Introduction

The transportation network design problem (TNDP) deals with optimization of transportation networks under specific constraints [1]. Specifically, TNDP is an important topic in transportation planning and development, where the objective is to minimize transporting costs or to maximize achieved profit. The discrete case of the network design problem (NDP) focuses on addition of new roads to a transportation network while the continuous case deals with capacity expansion of existing links. A mixture of both is referred to as the mixed NDP (see [2, 3]). See also [4, 5] for recent overviews of the NDP.

The decisions on production and transportation/supply planning must often be made without (exact) knowledge of the customer's demand. We will further emphasize the importance of uncertainty in problems of demand based management, that we interpret as a decision-making under uncertainty. Many real-world complex systems include these uncertainties, and they can be modeled in different ways. One of them is stochastic programming (SP), see [6]. The beginning of SP, and in particular stochastic linear programming, dates back to the 50's and early 60's of the last century. Although many ways have been proposed to model uncertain quantities, stochastic models have proved their flexibility and usefulness in diverse areas of science. Finally, integer programming deals with models where some of the variables are discrete, see [7] for general concepts.

The paper deals with two-stage stochastic TNDP. Two-stage programs have been named in various ways by different authors (e.g., two-stage programs, programs with recourse, two-stage programs with recourse). Such problem was firstly introduced by Dantzig [8] and Beale [9], independendtly, and has been studied extensively in the years since (see [6] or [10]). As a typical and famous example of the two-stage stochastic linear problem we have identified the so-called STORM model, which appears in paper by Mulvey and Ruszczyński [11]. It was used by the US Military to plan the allocation of aircrafts to routes during the Gulf War of 1991 (see also paper by Higle and Sen [12] for more details). In this original two-period problem, routes are scheduled to satisfy a set of demands at first-stage, demands occur, and unmet demands are delivered at higher costs in second-stage to account for shortcomings [13]. The STORM served well for tests of algorithms and evaluation of the solution quality (see, e.g., VSS fundamental concept in SP).

Various approaches have been taken to solve NDP as well as TNDP. For a detailed review of solution techniques see, e.g., [14, 15]. However, suitable algorithms for large scale problems are still under development as a challenging optimization field. This paper presents a hybrid algorithm for the solution of a two-stage scenario-based stochastic mixed integer linear program.

2 Two-Stage Stochastic Transportation Network Design Model

The important property of the problem considered in this paper is based on the fact that the decisions (at least some of them) must be made under uncertainty. In general, uncertainty may be included in the model in many ways. By computational purposes, we prefer to deal with the case of discrete random variable ξ with a finite support [16]. We model the situation by using a scenario-based (SB) approach to two-stage stochastic programs (see [16]) that helps us to capture the evolution of demand, and so, we can develop SP model for determining the operational and network design decisions.

In our two-stage problem, we will distinguish two different decisions that have to be made at two different time stages: (a) amount to be transported (b) new links to be added to the network. The first-stage decisions are made before the demand is observed, and so, we decide due to demand given by all considered scenarios. When the demand is observed, the second stage decisions are made to satisfy real customers' demand (with a higher transporting/penalty cost), see Fig. 1 for an illustrative example.

Then, we define the following two-stage SB stochastic mixed-integer linear program (SMILP):

$$
\begin{aligned}
\min \sum_{e} c_e x_e + \sum_{e_n} d_{e_n} \delta_{e_n} + \sum_{s} p_s \left(\sum_{e^*} q_{e^*} y_{e^*,s} + \sum_{i} (r^+_{i,s} w^+_{i,s} + r^-_{i,s} w^-_{i,s}) \right) \\
\text{s.t.} \ \sum_{e} A_{i,e} x_e + \sum_{e^*} A^*_{i,e^*} y_{e^*,s} = b_{i,s} - w^+_{i,s} + w^-_{i,s}, \ \forall i \in I, \forall s \in S, \\
x_{e_n} \le \delta_{e_n} M, \qquad & \forall e_n \in E_n, \\
w^+_{i,s} \le b_{i,s}, \qquad & \forall i \in I, \forall s \in S, \\
x_e \ge 0, \qquad & \forall e \in E, \\
y_{e^*,s} \ge 0, \qquad & \forall e^* \in E^*, \forall s \in S, \\
w^+_{i,s}, w^-_{i,s} \ge 0, \qquad & \forall i \in I, \forall s \in S, \\
\delta_{e_n} \in \{0, 1\}, \qquad & \forall e_n \in E_n,
\end{aligned}
\tag{1}
$$

with the first-stage decision variables:

x_e	:	amount of a product to be transported on edge e,
δ_{e_n}	:	1 if new edge e_n is built, 0 otherwise,

the second-stage decision variables:

$y_{e^*,s}$	:	amount of a product to be transported on edge e^* in scenario s,
$w^+_{i,s}$	:	shortages in a node i in scenario s,
$w^-_{i,s}$	:	leftovers in a node i in scenario s,

the sets of indices:

E	:	set of edges, $e \in E$,
E_n	:	set of new (built) edges, $e_n \in E_n$, $E_n \subset E$,
E^*	:	set of edges for the second-stage decision, $e^* \in E^*$, depending on the problem, e.g., $E^* \subset E$ or $E^* = E$,
I_1	:	set of customers (or places with a non-zero demand), $i_1 \in I_1$,
I_2	:	set of production places (or warehouses), $i_2 \in I_2$,
I_3	:	set of traffic nodes, $i_3 \in I_3$,
I	:	set of all nodes in the network, $i \in I, I = I_1 \cup I_2 \cup I_3$,
S	:	set of all possible scenarios, $s \in S, s = 1, 2, \dots, m$,

and parameters:

$A_{i,e}$	:	incidence matrix, $A_{i,e} \begin{cases} 1 & \text{if edge } e \text{ from a node to node } i \text{ exists,} \\ -1 & \text{if edge } e \text{ from node } i \text{ to a node exists,} \\ 0 & \text{otherwise,} \end{cases}$
A^*_{i,e^*}	:	incidence matrix for the second stage,
$b_{i,s}$	:	the demand in a node i for a scenario s (alternatively denoted by $\xi_{i,s}$),
c_e	:	unit transporting cost on edge e,
d_{e_n}	:	cost of building of new edge e_n,
q_{e^*}	:	unit cost for the second-stage transporting on edge e^*,
$r^+_{i,s}$	:	unit penalty cost for shortages at node i in scenario s,
$r^-_{i,s}$	:	unit penalty cost for leftovers at node i in scenario s,
p_s	:	probability of observing a scenario s, $\sum_s p_s = 1$,
M	:	a large number, e.g. $M = \sum_{i_2}(-b_{i_2})$.

Analysing principal ideas from references introduced in the previous paragraphs, we can see that we have turned from static programs discussed in previous papers, e.g. [17, 18], to the programs having two decision stages. So, the decision $\mathbf{x}$, obtained as the solution of the first-stage program (master program), is followed by the decision $\mathbf{y}(\xi)$ that solves the second-stage program (see [16]). The network design decisions (when to make them, respectively), δ's, can be modified due to needs of the particular problem. We illustrate the decision sequence in Fig. 1.

3 Computational Example

In this section, we introduce our testing network example that was previously used in [17–20] (see Fig. 2) and solve the two-stage SMILP given by (1). We use next modification of our hybrid algorithm that we previously used for simpler cases with recourse in [17, 18]. We also shortly demonstrate the main idea of the algorithm (see Sect. 3.1). For an exhaustive description of the interface and further details see [17, 18].

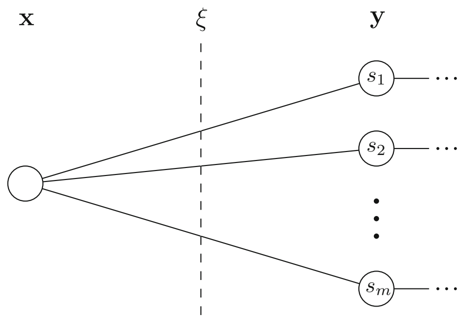


Fig. 1 Two-stage scenario tree illustrating sequence of transportation decisions

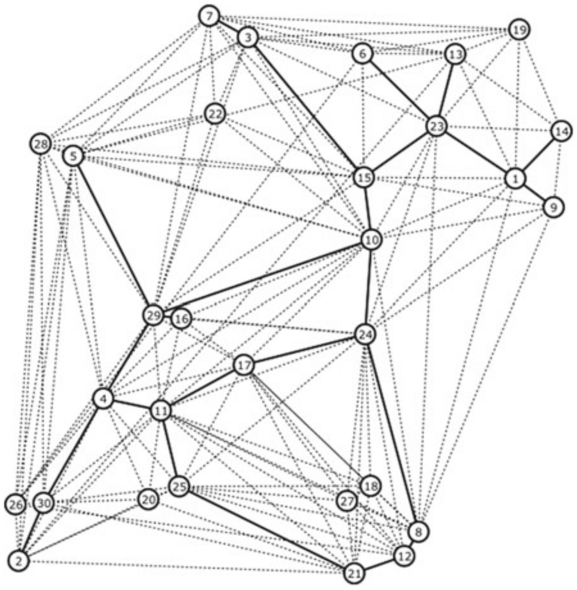


Fig. 2 Example of the network: $m = 30$, solid lines present existing network while dash lines are the potential network design variables/connections; moreover $b(i_1) > 0, I_1 = 1, 2, \dots 14$, $b(i_2) < 0, I_2 = 15, 16$, and $b(i_3) = 0, I_3 = 17, 18, \dots 30$

3.1 Hybrid Algorithm

We have implemented our model (1) in GAMS and we have solved it by the use of BARON, CONOPT, and CPLEX solvers for small test instances obtaining acceptable results (see [17, 18]). The attempt to solve larger test problems in the same way led to significant increase of computational time. Thus, we have modified and utilized our original hybrid computational technique that combines the GAMS code with genetic algorithm (GA). The algorithm is efficiently implemented in C++ with focus on GAMS-GA interface features.

Hybrid algorithm description

$\boxed{\textbf{1}}$ Initialization of parameters for all procedures and memory allocation.

$\boxed{\textbf{2}}$ Set up the scenario-based GAMS model (read model and data in *.gms files) for each scenario. Set up control parameters for the GA.

$\boxed{\textbf{3}}$ Create an initial population for each GA instance, so initial values of $0-1$ variables must be generated and placed into the so called `$INCLUDE` files, where they can be read-in by GAMS. Several runs of random generation are needed, corresponding to the population size and number of scenarios.

$\boxed{\textbf{4}}$ Repeatedly run the GAMS model by using the CPLEX solver. Each run solves the two-stage stochastic linear program for the fixed values of $0-1$ variables. Profit function values are calculated, also for new individuals created by means of the genetic operators, initially in $\boxed{\textbf{3.}}$ and then in $\boxed{\textbf{8.}}$

$\boxed{\textbf{5}}$ Save the best results obtained from GAMS in $\boxed{\textbf{4.}}$ for comparisons.

$\boxed{\textbf{6}}$ Test the algorithm termination rules and stop in case of their satisfaction. Otherwise continue till the last scenario solution is obtained.

$\boxed{\textbf{7}}$ Generate input values for the GA from GAMS results, see step $\boxed{\textbf{4.}}$ Specifically, the profit function values for each member of population of the GA are obtained from results of the GAMS runs in $\boxed{\textbf{4.}}$

$\boxed{\textbf{8}}$ Run GA to update the set of $0-1$ variables (population). Switch of binary variables belonging to edges having zero flow and recalculate the objective function (by using GAMS). Return to step $\boxed{\textbf{3.}}$

3.2 Results

In this subsection, we present results of the computations on Figs. 3 and 4. The first-stage decisions are realted to the network flow (Fig. 3a) and to the network design (Fig. 3b). When the real demands are known, we make the second-stage decisions: see the network flow (Fig. 4a) and the network design (Fig. 4b). Practically, if the second-stage decisions must be made as well as realized very quickly (e.g. in military applications, see similarities to aforementioned STORM ideas) then the network design decision from the second-stage should be realized in advance, i.e. in the first-stage. Then, the second stage decisions are restricted to only the transportation (network flow).

3.3 Conclusions and Further Research

The paper presents principle ideas behind the development of the original modification of hybrid algorithm involving GA and GAMS for the solution of the large-

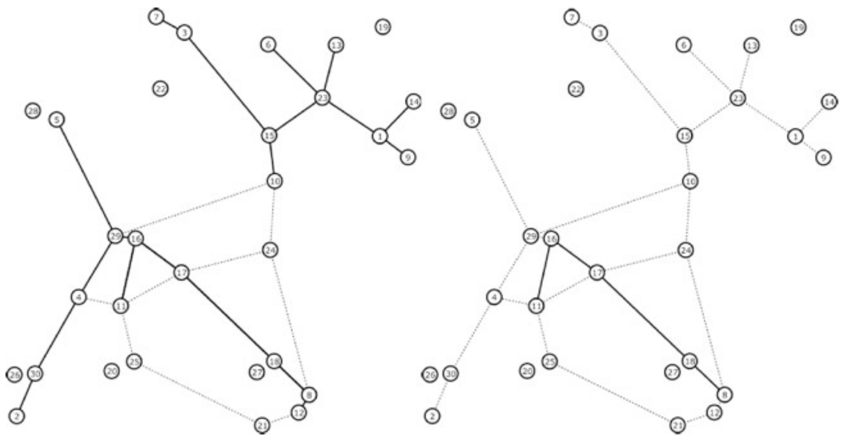

(a) Network flow in the first-stage. (b) Network design in the first stage.

Fig. 3 First-stage decisions: network flow (x) and network design (δ).

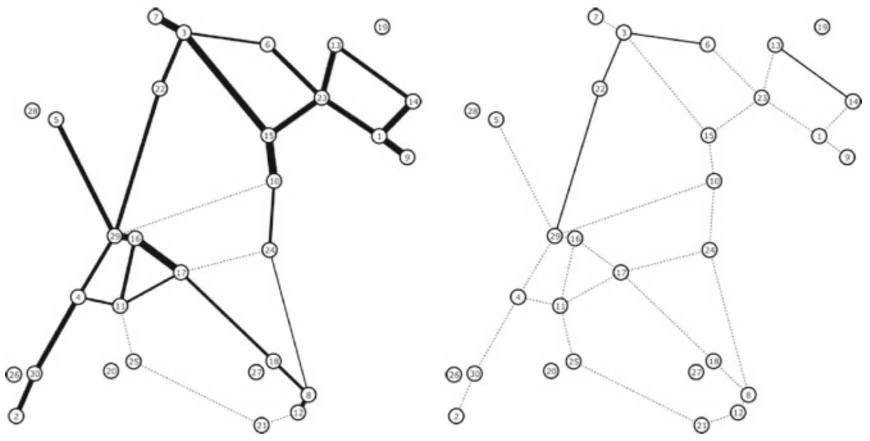

(a) Network flow in the second stage: line thickness captures number of uses of the edge in $n = 100$ scenarios. (b) Network design in the second-stage.

Fig. 4 Second-stage decisions: network flow (y) and network design (δ).

scale stochastic TNDP. The introduced hybrid algorithm can be further applied to engineering optimization of design parameters in civil engineering applications and continuous casting, where both integer and continuous variables may appear in the large scale instances of modeled problems. The idea is to test these conclusions more carefully for large test cases and real world applications in, e.g., waste management problems especially in the related transportation network design. The approach is also portable to other problems leading to nonlinear integer programming formulations. The hybrid algorithm can be further tested comparing with some other evolutionary approaches (see, e.g., [21]).

Acknowledgments The present work has been supported by the specific research project "Modern Methods of Applied Mathematics for the Use in Technical Sciences", no. FSI-S-14-2290, id. code 25053. We would like to acknowledge the help of Petr Jindra with the visualization of achieved results.

References

1. Ghiani, G., Laporte, G., Musmanno, R.: Introduction to Logistic Systems Planning and Control, Wiley-interscience series in systems and optimization. Wiley, Chichester (2004)
2. Magnati, T.L., Wong, R.T.: Network design and transportation planning: models and algorithms. Transp. Sci. **18**(1), 1–55 (1984)
3. Yang, H., Bell, M.G.H.: Models and algorithms for road network design: a review and some new developments. Transp. Rev. **18**(3), 257–278 (1998)
4. Liu, Ch.: A Stochastic Programming Approach for Transportation Network Protection. University of California, Davis, Research Report, Institute of Transportation Studies (2009)
5. Chen, A., Zhou, Z., Chootinan, P., Ryu, S., Yang, Ch., Wong, S.C.: Transport network design problem under uncertainty: a review and new developments. Transp. Rev. **31**(6), 743–768 (2011)
6. Kall, P., Wallace, S.W.: Stochastic Programming, 2nd edn. Wiley, Chichester (1994)
7. Wolsey, L.A.: Integer programming. Wiley-interscience Series in Discrete Mathematics and Optimization (1998)
8. Dantzig, G.B.: Linear programming under uncertainty. Manag. Sci. **1**, 197–206 (1955)
9. Beale, E.M.L.: On Minimizing a convex function subject to linear inequalities. J. Roy. Stat. Soc. **17b**, 173–184 (1955)
10. Glynn, P.W., Infanger, G.: Simulation-based confidence bounds for two-stage stochastic programs. Math. Program. **138**(1–2), 15–42 (2013)
11. Mulvey, J.M., Ruszczyński, A.: A new scenario decomposition method for large-scale stochastic optimization. Oper. Res. **43**(3), 477–490 (1995)
12. Higle, J.L., Sen, S.: Stochastic Decomposition. A Statistical Method for Large Scale Stochastic Linear Programming. Kluwer Academic Publishers, Dordrecht (2004)
13. Holmes, D.: A Collection of Stochastic Programming Problems. Technical report, University of Michigan (1994)
14. Babazadeh, A., Poorzahedy, H., Nikoosokhan, S.: Application of particle swarm optimization to transportation network design problem. J. King Saud Univ. Sci. **23**, 293–300 (2011)
15. Poorzahedy, H., Rouhani, O.M.: Hybrid meta-heuristic algorithms for solving network design problem. Eur. J. Oper. Res. **182**, 578–596 (2007)
16. Popela, P., Novotný, J., Roupec, J., Hrabec, D., Olstad, A.: Two-stage stochastic programming for engineering problems. Eng. Mech. **21**(5), 335–353 (2014)
17. Roupec, J., Popela, P., Hrabec, D., Novotný, J., Olstad, A., Haugen, K.K.: Hybrid algorithm for network design problem with uncertain demands. In: Proceedings of the World Congress on Engineering and Computer Science, WCECS 2013, pp. 554–559. San Francisco, USA (2013)
18. Hrabec, D., Popela, P., Roupec, J., Jindra, P., Haugen, K.K., Novotný, J., Olstad, A.: Hybrid algorithm for wait-and-see network design problem. In: 20th International Conference on the Soft Computing MENDEL 2014, pp. 97–104. Brno, Czech Republic (2014)
19. Hrabec. D.: Stochastic programming for engineering design. Diploma thesis, Department of Mathematics, Faculty of Mechanical Engineering, BUT, 2011

20. Hrabec, D., Popela, P., Novotný, J., Haugen, K.K., Olstad, A.: The stochastic network design problem with pricing. In: 18th International Conference on the Soft Computing MENDEL 2012, pp. 416–421. Brno, Czech Republic (2012)
21. Stodola, P., Mazal, J., Podhorec, M., Litvaj, O.: Using the ant colony optimization algorithm for the capacitated vehicle routing problem. In: 16th International Conference on Mechatronics - Mechatronika (ME), pp. 503–510 (2014)
22. Matousek, R.: HC12: the principle of CUDA implementation. In Proceedings of 16th International Conference on Soft Computing MENDEL 2010, Mendel series, vol. 2010, pp. 303–308, Brno (2010). ISSN: 1803–3814

A Novel Hyper-Heuristic Approach for Channel Assignment in Cognitive Radio Networks

Emrullah Gazioglu, A. Sima Etaner-Uyar and Berk Canberk

Abstract Wireless networks communicate with each other using radio spectrum bands which are assigned to license owners. Due to the fixed spectrum assignment policy, a large portion of the spectrum stays unused. The aim of cognitive radio is enabling users which do not hold a license to be able to access the spectrum assigned to license owners. In the channel assignment problem, the objective is to assign channels to unlicensed users in order to maximize channel utilization without causing any interference to licensed users. In this study, we propose a hyper-heuristic approach to solve the channel assignment problem in cognitive radio networks. Results show that our approach gives high channel utilization rates by allowing unlicensed users to access the channels owned by licensed users. The results are promising and promote further study.

1 Introduction

Wireless networks communicate with each other using radio spectrum bands which are assigned to license owners. Traditionally, there is a fixed spectrum assignment policy issued by the governments or the communication companies [1]. Because of the fixed spectrum assignment policy, a large portion of the spectrum stays unused. To utilize these unused spectrum portions, the idea of Cognitive Radio (CR) was proposed by J.Mitola [2, 3]. A CR is defined by the The Federal Communications Commission (FCC) as [4]: “A CR is a radio that can change its transmitter parameters

E. Gazioglu (✉) · A.S. Etaner-Uyar · B. Canberk
Faculty of Computer and Informatics, Istanbul Technical University,
34469 Istanbul, Turkey
e-mail: egazioglu@itu.edu.tr
http://www.cs.itu.edu.tr/en

A.S. Etaner-Uyar
e-mail: etaner@itu.edu.tr

B. Canberk
e-mail: canberk@itu.edu.tr

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_3

based on its interaction with the environment in which it operates. This interaction may involve active negotiation or communications with other spectrum users and/or passive sensing and decision making within the radio." The aim of CR is maximizing spectrum utilization in an intelligent way. By doing that, CR enables the unlicensed users to also access the spectrum. However, this process should be done in such a way that when an unlicensed user attempts to use a spectrum, it should not interfere with the licensed users.

In CR Networks (CRN) terminology, *a primary user (PU)* is a licensed user which has been assigned a spectrum, while *a secondary user (SU)* is an unlicensed user which does not have a license to access a spectrum. For this reason, SUs use the CR technology to access the spectrum when the spectrum is not occupied by the PU. This unoccupied spectrum portion is represented by *a spectrum hole* or *white space*. In CRN, PUs are known as the owners of a spectrum. They can use the spectrum whenever they want. On the other hand, SUs can use the spectrum if and only if the spectrum is not currently occupied by the PU. However, in [5], a new data transmission protocol, which allows a PU and a SU to work cooperatively, was proposed. In this method, a PU releases a portion of the bandwidth (BW) to the SU to transmit its own data in exchange for making the SU to also relay the PU's data.

A CRN fundamentally includes primary networks (PN) and secondary networks (SN) [1]. Normally, PNs do not share the spectrum with the SUs. On the other hand, SNs have a Base Station (BS) to assign channels to the SUs that are requesting spectrum access. However, before such an an access can be allowed, the BS must have some information on the PU's spectrum usage in the form of time slots. To collect this information from the environment, SUs sense (Spectrum Sensing) the PUs and they transmit the collected data to the BS. After that, the BS uses different data fusion techniques to put this data together and makes decisions (Spectrum Decision) to assign (Channel Assignment) available channels to the SUs. In this study, we only deal with the channel assignment stage of the problem.

Channel Assignment (CA) is a fundamental technique to control interference in the CRNs. The aim of CA is to assign channels to SUs in order to maximize channel utilization. In CA, the percentage of channel utilization can be increased by transmitting SU data and PU data simultaneously as given in [5].

In this study, we use hyper-heuristics (HH) to solve the CA problem. In literature, CA algorithms try to optimize various objectives such as utilization optimization, interference minimization, network overhead minimization, throughput maximization, etc [6]. Our objective is to maximize channel utilization. Hyper-heuristics are methods that work on the search space of low level heuristics rather than on the search space of solution candidates [7]. Single point based search heuristics or population based metaheuristics can serve as hyper-heuristics. In this study we use a single point based search heuristic, namely *Adaptive Iterated Construction Search (AICS)* [8]. In [9] and more recently in [6], a taxonomy of solution techniques to the CA problem is provided. One classification is based on the solution technique used. Based on this, our aproach as HH falls under the category of *heuristics* [6].

The rest of the paper is organized as follows: In Sect. 2, background information on hyper-heuristics and the AICS method used as a HH in this study is given. The problem definition and the proposed method is described in detail in Sect. 3. Section 4 shows the results. Section 5 concludes the paper and provides possibilities for future work.

2 Background

2.1 *Hyper-Heuristics*

In [10], hyper-heuristics are defined as heuristics that select heuristics. A more recent definition of HHs is given in [11] as: "A hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems". HHs work on a search space of low-level heuristics (LLH) as opposed to working on the search space of solutions. There are two types of HHs, i.e. *Selection HHs* and *Generation HHs* [11]. Selection HHs choose from existing heuristics defined for a problem, while generation HHs generate new heuristics for a given problem by using components of existing heuristics. In this study we work with selection HHs, so for the rest of the paper we will use HHs to denote selection HHs.

As demonstrated in Fig. 1, in a HH framework the HH is isolated from the problem domain. It can only access the LLHs provided for that problem domain. This implies that once a HH has been developed, it can be applied to different problem domains by changing the set of LLHs and the evaluation function [12].

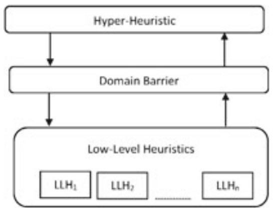


Fig. 1 Hyper-heuristic framework

Single point based search approaches as well as population based metaheuristics, such as ant colony optimization algorithms, can be used as HHs. In this study we use a single point based search approach, namely *Adaptive Iterated Construction Search (AICS)* [8]. AICS can be considered as being a simple Ant Colony Optimization (ACO) algorithm [13] which works using a single ant.

2.2 Adaptive Iterated Construction Search

ACO [13], is a well-known metaheuristic approach inspired from the behavior of real ants. One of the key concepts of ACO is the use of pheromone trails. Real ants release pheromones along the path from their nest to the food source. In ACO, pheromone values are associated with the edges between the nodes of a construction graph. The pheromone values in the pheromone matrix are updated when a new ant chooses to use the corresponding edges. The choosing process is accomplished with a probability p calculated using the pheromone level value (τ_{ij}) between the nodes and some heuristic information (η_{ij}). At the beginning of the algorithm, all values in the pheromone matrix are initialized with a small value τ_0. Each ant adds selected solution components into its own solution candidate set by walking on the construction graph. At the end of its walk, it constructs a complete solution candidate. In AICS, a single ant constructs a complete solution candidate and updates the pheromone matrix for the next iteration. AICS can be used as a HH mechanism like proposed in [14], where the probability p for choosing the next component is calculated as given in Eq. 1:

$$p_{ij} = \begin{cases} \frac{\tau_{ij}^{\alpha}}{\sum_{l \in N_i} \tau_{il}^{\alpha}} & \text{if } j \in N_i \\ 0 & \text{if } j \notin N_i \end{cases} \tag{1}$$

where τ_{ij} is the pheromone level between components i and j respectively, α is a parameter used to determine the effect of the pheromone level, N_i is the allowed neighborhood of the ant when it is at node i. Note that, the heuristic information η_{ij} is not used in this equation for calculating the probability values because there is no available heuristic information for the CA problem we addressed in this study.

In each iteration, after a complete solution candidate is constructed, the pheromone levels are updated. First, all the pheromone values in the pheromone matrix are evaporated with a constant factor. Then the pheromone values which are between the components that were used by the ant are increased. Pheromone evaporation and update are calculated as in Eqs. 2 and 3 respectively,

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} \tag{2}$$

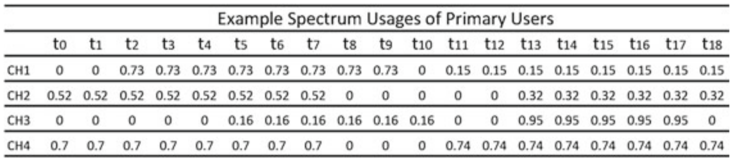
$$\tau_{ij} \leftarrow \tau_{ij} + \Delta(i, j, s) \quad \text{if } edge(i, j) \in T \tag{3}$$

where $\Delta(i, j, s)$ is the amount of pheromone to be added between the corresponding solution components. It is set to $Q/fitness(s)$ if the $fitness(s) > 0$ (Q is a constant), otherwise it is set to a very big number. The constant evaporation variable ρ is a value between 0 and 1. Finally, T is the set of edges which have been visited.

3 The Channel Assignment Problem

3.1 Problem Definition

In our CA problem model, both PUs and SUs can transmit their data simultaneously on the same channel. However, we should remember that a SU can use the spectrum as far as the PU permits it in the form of BW. In this study, we assume an underlying network architecture. More specifically, a SU can utilize the vacant channel as long as it does not disturb the PU. For example, if we assume that the possible total channel utilization is 1.0, and a PU is using the channel with a rate of 0.4, this means that the SU can use it with a rate of 0.6. In Fig. 2, a sample channel usage of PUs can be seen. The rows represent the channels and the columns represent time. The values in the cells represent the PUs' BW usage rate for each channel at each point in time, e.g. at time t_0 the PUs use CH_2 and CH_4 with the rates of 0.52 and 0.7 respectively and use no BW on channels CH_1 and CH_3. This means that at time t_0 SUs can use the full BW on channels CH_1 and CH_3, while they can use channels CH_2 and CH_4 with the rates of 0.48 and 0.3 respectively.



Example Spectrum Usages of Primary Users

	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18
CH1	0	0	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
CH2	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0	0	0	0	0	0.32	0.32	0.32	0.32	0.32	0.32
CH3	0	0	0	0	0	0.16	0.16	0.16	0.16	0.16	0.16	0	0	0.95	0.95	0.95	0.95	0.95	0
CH4	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0	0	0	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74

Fig. 2 Sample spectrum usage

Our main aim in solving the CA problem is to maximize the total channel utilization at each point in time by allowing the SUs to use the remaining BW on the channels. For our model of the CA problem, the objective function can be defined as given in Eq. 4.

$$maximize \sum_{j=1}^{n} \left[req(N_j) + \sum_{i=1}^{r} (usg(M_i).x_{ji}) \right] \tag{4}$$

subject to

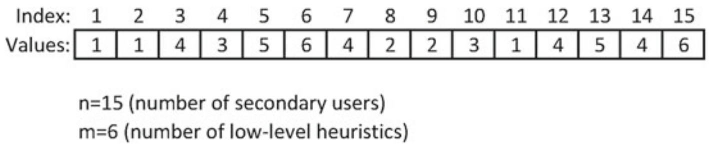
$$0 \leq req(N_j) + usg(M_i) \leq 1 \tag{5}$$

where n is the number of SUs; r is the number of channels (i.e. the number of PUs), $req(N_j)$ is the j_{th} SU's BW request rate, $usg(M_i)$ is the i_{th} PU's channel usage rate, x_{ji} is the 0/1 decision variable that shows whether channel M_i is assigned to SU N_j or not.

3.2 Solution Approach

To solve the CA problem, we use a HH approach based on AICS. Given a traffic matrix, i.e. the PUs' spectrum usage information for each channel and for each time slot (e.g. as shown in Fig. 2), the HH aims to assign a channel to the SUs for each time slot. To accomplish this, first, the HH constructs a solution candidate which includes a set of LLHs. Then, each LLH in the solution candidate is invoked in the order given by the solution candidate, to assign a channel to a SU. Next, the fitness value of the resulting assignments is calculated.

Solution Candidate: In Fig. 3 a possible solution candidate is illustrated. It can be seen from this figure, that a LLH can be used in a solution candidate more than once. The values in the solution candidate are integers between 1 and the total number of LLHs. Each number shows which LLH is used to choose the next SU for channel assignment. Since there are n SUs which require a channel assignment, the length of the solution candidate is equal to n.



Index:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Values:	1	1	4	3	5	6	4	2	2	3	1	4	5	4	6

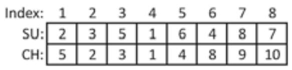
n=15 (number of secondary users)
m=6 (number of low-level heuristics)

Fig. 3 Example solution candidate

At each step of the assignment process, a LLH chooses a SU. For example in Fig. 3, the first LLH is used to choose the first and the second SUs while the fourth LLH is used to choose the third SU, etc. For each chosen SU, the channel assignment is done based on a *best-fit* policy. The aim of *best-fit* policy is to assign a channel to a SU based on its BW request so as to maximize channel utilization.

Assignment Table: Once the assignment is finished, the fitness value of the created assignment table is evaluated. In Fig. 4, a sample assignment table is given. The table shows channel assignments for 8 SUs. Each entry in the table denotes the channel number which was assigned to the corresponding SU, e.g. the fifth channel is assigned to the 2nd SU.

Fig. 4 Example assignment table



Index:	1	2	3	4	5	6	7	8
SU:	2	3	5	1	6	4	8	7
CH:	5	2	3	1	4	8	9	10

SU Properties: Each SU has some properties which define its behavior. In this study we used three different properties for the SUs: the BW *request* which is a number between 0 and 1; a *priority level* for which there are three commonly used settings as Gold, Silver and Bronze; the *degree* of the node corresponding the the SU in the graph representing the underlying network.

The Fitness Function: The objective of the CA problem is to maximize channel utilization. To achieve this we designed a fitness function which consists of three parts.

– Unused BW (c_1): Amount of total BW that was not used. This is to be minimized, therefore, it forces the algorithm to use as much spectrum BW as possible.
– Overflowed BW (c_2): Amount of total BW that is assigned to SUs which is more than the available amount on each channel after the corresponding PUs utilization is subtracted. This is to be minimized because it causes a solution candidate to be infeasible due to the constraint defined in Eq. 5. However, to let the algorithm be able to search the whole search space freely, infeasible solutions are allowed during the search stage but not as a final solution.
– Unassigned SU (c_3): Number of SUs that have a request for that time slot but are not assigned to any channels if there are any available channels.

Each part of the fitness function is normalized based on worst case scenarios as shown in Eqs. 6, 7 and 8 respectively.

$$c_1 = c_1 / number\ of\ assigned\ SU \tag{6}$$

$$c_2 = c_2 / number\ of\ assigned\ SU \tag{7}$$

$$c_3 = c_3 / (G.(number\ of\ SUs - number\ of\ available\ channels)) \tag{8}$$

where G is a number denoting the value used for the gold level priorities. Based on these three parts, the total fitness function is calculated as given in Eq. 9.

$$fitness(x) = \alpha_1.c_1 + \alpha_2.c_2 + \alpha_3.c_3 \tag{9}$$

where α_1, α_2 and α_3 are weight coefficients for each component of the fitness function. The algorithm tries to minimize this fitness function for each time slot in order to maximize the channel utilization.

Low-level Heuristics: In this study we used six LLHs for the CA problem. Each LLH selects a SU when it is invoked. For all LLHs, the channel assignment is done using the *best-fit* policy.

– The *Maximum/Minimum Degree Heuristic* selects the SU with the maximum/minimum degree based on a graph with SUs as vertices.
– The *Maximum/Minimum Request Heuristic* selects the SU with the maximum/minimum BW request.
– The *Maximum Priority Heuristic* selects the SU with the maximum priority.
– The *Random Heuristic* selects a SU randomly.

3.3 Proposed Algorithm

In this study, we proposed a solution approach to the CA problem in CRNs which uses the AICS algorithm as a HH to select the SUs at each step of the CA. The pseudo-code of the whole proposed solution approach is given in Algorithm 1.

```
$fitness \leftarrow calculate\ a\ random\ solution$;
for each time slot $t$ do
    $s \leftarrow AICS(t, fitness)$;
    $assignmentTable \leftarrow invokeSelectedLLHs(s)$;
    $fitness \leftarrow calculateFitness(assignmentTable)$;
end
```

Algorithm 1: Pseudo code of the proposed solution approaches.

The AICS approach in [8] is used in this study as a HH. In AICS, once the probabilities are calculated, the well known roulette-wheel selection technique used in evolutionary algorithms [15] is used for selecting the next LLH based on the calculated probabilities from the pheromone values. As noted before, since there is no heuristic information, probabilities are based only on pheromone values [14].

4 Experiments

4.1 Experiment Design

To test the proposed approach we generated several test instances based on four parameters as follows:

– Traffic matrix: We generated the traffic matrix using the Poisson distribution to determine the arrival times of the packets. We used three different λ values for the Poisson distribution, which are $\lambda = 1, 5, 10$. As λ gets bigger, the arrivals of the packets become sparse and vice-versa. To determine the size of the packets we used the uniform distribution. While doing that, we made sure there were no overlapping packets.
– Packet counts: We used two different packet counts in the experiments as 10 and 20. The BW of the packets are generated randomly between 0 and 1. These represent the PUs channel usage rates.
– Number of SUs: Four different settings were used as $(80, 100, 200, 300)$. The BW requests for the SUs are generated randomly between 0 and 1. The priority for each SU is determined randomly as 0.5, 0.3 or 0.2 corresponding to Gold, Silver and Bronze respectively.
– Number of channels: Three different settings were used as $(60, 80, 100)$.

By using the above parameters we created 72 different test instances. In AICS, for ρ and for the constant Q we used fixed values as 0.1 and 1000 respectively as suggested in [14]. The coefficients $\alpha_1, \alpha_2, \alpha_3$ used in the fitness calculation are set to 10, 1000, 10 respectively. These values were determined experimentally.

We evaluate the approaches tested in this study based on two criteria:

– *Channel Assignment Success Rate (CASR)* shows the success rate of an approach. It is calculated as the percentage of feasible assignments to the total number of channels. A feasible assignment is one where the constraint given in Eq. 5 is satisfied for all time slots.
– *Channel Utilization Rate (CUR)* shows the utilization rate of the channels at time t, weighted by the CASR as shown in Eq. 10.

$$CUR(t) = \frac{\sum_{i=1}^{r} util_i(t)}{r} * CASR \tag{10}$$

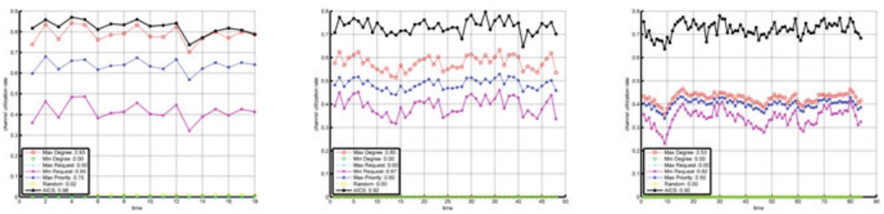
$util_i(t)$ is the channel utilization rate for channel i at time t and r is the total number of channels.

4.2 Results and Discussions

To show the effectiveness of the HH approach which uses six LLHs, each of the 72 test instances was solved with the proposed HH (AICS) as well as each LLH separately. Since AICS is a stochastic algorithm, we ran the AICS algorithm 20 times independently for each test instance. For this reason, in our result plots, we show the average value of 20 runs for AICS. The results are shown as *Weighted Channel Utilization Rates over Time (CUR(t))*. Due to lack of space, here we only show sample plots[1] for seven different test instances. The samples are chosen so as to show the effect of each parameter, i.e. λ used in generating the traffic matrix, the number of channels (CH) and the number of SUs. Note that the packet count is kept fixed at 20 since it only affects the length of the time axis and does not have an important effect on performance. In all plots, the results for the six LLHs and the proposed approach denoted as AICS are given, where the x-axis shows the time and the y-axis shows CUR.

The Effect of λ In this part, we want to analyze the effect of the traffic density on performance, so we fix the number of channels and the number of SUs as $CH = 60$ and $SU = 300$ respectively. In Fig. 5, we provide the plots for different λ settings. $\lambda = 1$ denotes dense traffic where the interarrival time between the packets is small. As λ increases, packet traffic becomes more sparse.

[1]The plots for all 72 instances can be seen on the web page at http://web.itu.edu.tr/egazioglu/cr.

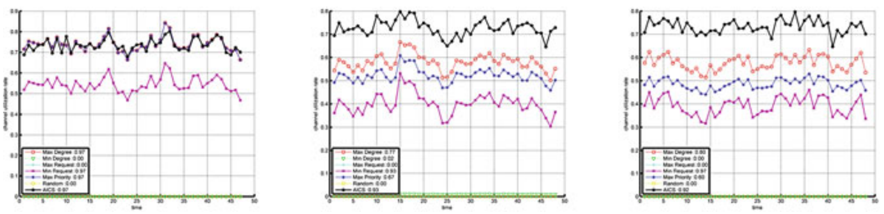


(a) λ: 1 (dense) (b) λ: 5 (medium) (c) λ: 10 (sparse)

Fig. 5 Example CUR plots for different λ settings where $CH = 60$ and $SU = 300$

It can be seen in Fig. 5 that the drop in CUR and CASR are much higher for the individual LLHs than for AICS. As the traffic density decreases, the performance of the LLHs drops. However, AICS is not affected from the change in traffic density.

The Effect of the Number of SUs In this part, we want to analyze the effect of the number of SUs in the system that want to use the available channel BW, so we fix the traffic density and the number of channels as $\lambda = 5$ and $CH = 60$ respectively. In Fig. 6, we provide the plots for different SU settings. We chose $SU = 100, 200, 300$ to denote a low, medium and high number of SUs respectively.

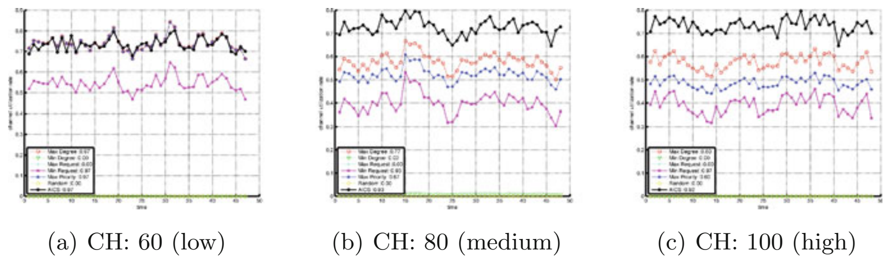


(a) SU: 100 (low) (b) SU: 200 (medium) (c) SU: 300 (high)

Fig. 6 Example CUR plots for different SU settings where $\lambda = 5$ and $CH = 60$

It can be seen in Fig. 6 that there is a high drop in CUR and CASR for the individual LLHs. This is to be expected because as the number of SUs increases, it becomes harder to assign the limited amount of channel BW efficiently. There is a slight decrease in CASR for the AICS but its CUR is not affected by the number of SUs in the system. This shows that AICS is able to provide good performance with regard to CASR and CUR for a range of SU counts.

The Effect of the Number of Channels In this part, we want to analyze the effect of the number of channels (CH) (i.e. number of PUs) in the system, so we fix the traffic density and the number of SUs as $\lambda = 5$ and $SU = 300$ respectively. In Fig. 7, we provide the plots for different CH settings. We used $CH = 60, 80, 100$ to denote a low, medium and high number of available channels respectively.



(a) CH: 60 (low) (b) CH: 80 (medium) (c) CH: 100 (high)

Fig. 7 Example CUR plots for different *SU* settings where $\lambda = 5$ and $CH = 60$

It can be seen in Fig. 7 that all approaches are affected similarly from the changes in the channel counts and AICS is the best performer in all three cases with regard to CASR and CUR.

Overall Evaluation In Figs. 5–7 it can be seen that among the LLHs, *Max. Degree* and *Max. Priority* LLHs are the best performers with regard to CUR in all cases. Similarly, *Min. Request* is the best performer among all LLHs with regard to CASR. However, the CUR for this LLH is very low. The reason for this behavior is that this LLH chooses the SUs in the increasing order of their requests. It is easier to find a channel assignment for smaller requests, therefore the CASR of this LLH is high. However, since it assigns channels to those SUs with lower requests first, the CURs of the assigned channels do not increase much after the assignments. *Min. Degree*, *Max. Request* and *Random* are the worst performing LLHs with $CASR = 0$. In the plots given in the paper, AICS is either the best performer or it has a performance of similar quality to the well performing LLHs.

These observations are generally true also when the results for all 72 instances are considered. However, there are a very few cases where *Max. Request* achieves a $CASR > 0$. However, even then, its performance is not comparable to the best performers. Over all instances, there are some cases where AICS is outperformed by one or more of the LLHs but this is to be expected since HHs are designed to be general solvers which should be applicable to all instances without requiring specific information about the nature of a specific instance. In this sense, the results show that AICS is very successful with regard to both CASR and CUR over all instances.

5 Conclusion and Future Work

In this work, we studied the channel assignment problem in CRNs. We applied the AICS algorithm as a HH which uses six LLHs. Experiments were performed to see the effects of different parameters on the performance of the proposed approach. Test instances that exhibit various properties were generated and each LLH was also tested on each test instance separately. The approaches were evaluated based on two criteria, namely their success rates in making feasible channel assignments and the channel utilization rates they provided over time.

The results show that for some types of instances one LLH is better whereas another LLH may be better for others. So overall, as each LLH favors one aspect of the instance (e.g. one LLH uses the requests of the SUs to select a SU), it performs better in related cases. However, HHs harvest the advantages of the LLHs and benefit from being able to utilize all of them in each instance. The learning mechanism incorporated in the AICS in the form of pheromone trails helps the HH to learn over time which LLHs are good. However, as is expected HHs allow for good and acceptable results for all instances, without needing to know the properties of each instance, but they may not be the best performing approach in all instances.

The experiments provided in this study suggest that using Hyper-Heuristics allows unlicensed users (SUs) to access spectrum with a high channel utilization rate under many different types of traffic and channel and SU counts. Overall, AICS as a HH is a very promising technique for the channel assignment problem used here. As future work, other types of heuristics and metaheuristics may be considered as a HH.

References

1. Haykin, Simon: Cognitive radio: brain-empowered wireless communications. IEEE J. Sel. Areas Commun. **23**(2), 201–220 (2005)
2. Mitola, J., Maguire, G.Q.: Cognitive radio: making software radios more personal. IEEE Pers. Commun. **6**(4), 13–18 (1999)
3. Mitola, J.: Cognitive radio–an integrated agent architecture for software defined radio (2000)
4. FCC: Notice of proposed rule making and order, et docket no. 03–322 (2003)
5. Su, W., Matyjas, J.D., Batalama, S.: Active cooperation between primary users and cognitive radio users in heterogeneous ad-hoc networks. IEEE Trans. Signal Process. **60**(4), 1796–1805 (2012)
6. Ahmed, E., Gani, A., Abolfazli, S., Yao, L.J., Khan, S.U.: Channel assignment algorithms in cognitive radio networks: taxonomy, open issues, and challenges. IEEE Commun. Surv. Tutor. (99), 1–1 (2014)
7. Peter Ross: Hyper-heuristics. In: Search methodologies, pp. 529–556. Springer (2005)
8. Hoos, H.H., Stützle, T.: Stochastic Local Search: Foundations & applications. Elsevier (2004)
9. Tragos, E.Z., Zeadally, S., Fragkiadakis, A.G., Siris, V.A.: Spectrum assignment in cognitive radio networks: a comprehensive survey. IEEE Commun. Surv. Tutor. **15**(3), 1108–1135 (2013)
10. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Practice and Theory of Automated Timetabling III, pp. 176–190. Springer (2001)
11. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A classification of hyper-heuristic approaches. In: Handbook of Metaheuristics, pp. 449–468. Springer (2010)
12. 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. Springer (2003)
13. Stützle, M.E., Dorigo, T.: Ant colony optimization (2004)
14. Ergin, F.C., Uyar, A., Yayimli, A.: Investigation of hyper-heuristics for designing survivable virtual topologies in optical wdm networks. In: Applications of Evolutionary Computation, pp. 1–10. Springer (2011)
15. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer Science & Business Media (2003)

Improved Bees Algorithm for Protein Structure Prediction Using AB Off-Lattice Model

Nanda Dulal Jana, Jaya Sil and Swagatam Das

Abstract Protein Structure Prediction (PSP) using sequence of amino acids is a multimodal optimization problem and belongs to *NP* hard class. Researchers and scientists put their efforts to design efficient computational intelligent algorithm for solving this kind of problem. Bees Algorithm (BA) is a swarm intelligence based algorithm inspired by the foraging behaviour of honey bees colony, already exhibits its potential ability for solving optimization problems. However, it may produce premature convergence when solving PSP like problems. To prevent this situation, Adaptive Polynomial Mutation based Bees Algorithm (APM-BA) has been proposed in this paper for predicting protein structure in 2D AB off-lattice model. In this strategy, each of best scout bees are mutated with adaptive polynomial mutation technique when their performances are no more improve during execution phase. The experiments are conducted on artificial and real protein sequences and numerical results show that the proposed algorithm has strong ability for solving PSP problem having minimum energy.

Keywords Protein structure prediction · Bees algorithm · Adaptive polynomial mutation · Artificial and real protein sequences

N.D. Jana (✉)
Department of IT, National Institute of Technology, Durgapur 713209, India
e-mail: nanda.jana@gmail.com

J. Sil
Department of CST, Indian Institute of Engineering Science & Technology, Shibpur 711103, India
e-mail: js@cs.iiests.ac.in

S. Das
ECS Unit, Indian Statistical Institute, Kolkata 700108, India
e-mail: swagatam.das@isical.ac.in

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_4

1 Introduction

Proteins are the primary building blocks in all living organisms and represented by a sequence of 20 different amino acids. Biological functions of proteins are closely related with their structures which play an important role in drug design, disease prediction and many more [1, 2]. Therefore, protein structure prediction is an important research area in computational biology. Anfinsen's thermodynamic hypothesis [3] states that the native structure of a protein corresponds to the global minimum of the free energy surface of the protein. So, the protein structure prediction problem can be treated as a global optimization problem. This problem is of NP-hard [4] and its multimodality characteristics increase with the protein sequence length [5].

Experimental methods like X-ray crystallography and Nuclear Magnetic Resonance (NMR) are time consuming and expensive to predict the structure of proteins. Moreover, due to strict laboratory requirements and heavy operational burdens, it is not always feasible to determine the protein structure experimentally. Therefore, researchers focused on predicting protein structure from the given amino acid sequences by computational methods [6]. A successful study of computational methods in PSP reveals two facts. The first part is the consideration of physical model which corresponds to a potential energy function. The second part involves searching of global minimum of the potential energy function. In the literature, most of the physical models are grouped into two classes of residues: hydrophobic (non-polar or H) and hydrophilic (polar or P) instead of considering 20 different amino acids individually. The most widely used simplified model is HP lattice models [7]. An off-lattice model is a generalization of HP lattice model known as AB off-lattice model proposed by Stillinger et al. [8] where the hydrophobic and the hydrophilic residues are labelled by 'A' and 'B' respectively. Many computational intelligent algorithms have been applied on AB off-lattice model for finding global minimum energy to predict protein structure [9–18].

The Bees Algorithm (BA) [19, 20] is a swarm intelligence based algorithm inspired by the food foraging behaviour of honey bee colonies developed in 2005 by D. T. Pham. It performs a kind of exploitative neighbourhood search combined with random explorative search, successfully applied to many engineering optimization problems. However, it has the limitation of premature convergence due to lack of diversity in the search space when solving multimodal optimization problems. In this paper, to prevent premature convergence and improve the performance of BA, we have proposed, adaptive polynomial mutation based bees algorithm (APM-BA) for solving PSP problem in 2D AB off-lattice model. Adaptive polynomial mutation is applied on best scout bees which do not improve their visited site in a predefined limit, known 'trial' counter of inefficient search. As a result, there is a high chance to jump out from visited site to unvisited site and made exploration on the search space. Experiments are conducted on artificial and real protein sequences using 2D AB off-lattice model. The numerical results show that the proposed algorithm is suitable for solving PSP problem having minimum energy. Moreover, results are compared with other algorithms demonstrating efficiency of the proposed method.

The paper is organized as follows: In Sects. 2 and 3, basic principle of 2D AB off-lattice model and bees algorithm are describe, respectively. The details of APM-BA are presented in Sect. 4, followed by experimental setups and results in Sect. 5. Finally, the conclusion is drawn and future works are highlighted in the Sect. 6.

2 AB Off-Lattice Model

AB off-lattice model, known as toy model proposed by Stillinger et al. in 1993 [8] and widely used for predicting the structure of a protein sequence due to its simplicity. In this model, 20 amino acids are classified into hydrophobic and hydrophilic residues, labelled as 'A' and 'B' respectively. Two residues are linked by rigid unit-length bonds and the angle between two bonds can change freely in two dimensional Euclidean space. An n length protein sequence is represented by (n-2) bend angles $\theta_2, \theta_3, ..., \theta_{n-1}$ at each of the non-terminal residues. Each bend angles θ_i are in the range $-180°$ to $180°$ and $\theta_i = 0$ represents linearity in the successive bonds. The bend angel, $\theta_i \in [-180°, 0)$ and $\theta_i \in (0, 180°]$ represent rotation of amino acids in clockwise and counter clockwise direction respectively. A 2D off-lattice model of a protein sequence with length 9 shown in Fig. 1. The AB off-lattice model represents the intra-molecular potential energy for each molecule with backbone bend potentials V_1 and nonbonded interactions V_2. Amino acids along the backbone can be conveniently encoded by a set of bipolar variables ξ_i. If ξ_i =1, the i^{th} amino acid is A while ξ_i = -1, it is B. Hence, the total potential energy function $\boldsymbol{\Phi}$ for any n length protein sequence is expressed using Eq. 1.

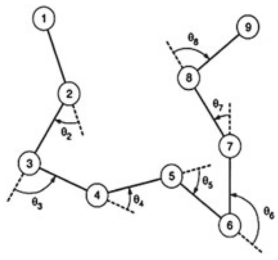


Fig. 1 2D off-lattice model of a protein sequence with length 9

$$\boldsymbol{\Phi} = \sum_{i=2}^{n-1} V_1(\theta_i) + \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} V_2(r_{ij}, \xi_i, \xi_j) \tag{1}$$

Where V_1 is the bending potential, independent of protein sequence as defined by Eq. 2

$$V_1(\theta_i) = \frac{1}{4}(1 - cos\theta_i) \tag{2}$$

The nonbonded interactions V_2 have a species-dependent Lennard-Jones 12, 6 form, described in Eqs. 3 and 4 respectively.

$$V_2(r_{ij}, \xi_i, \xi_j) = 4[r_{ij}^{-12} - C(\xi_i, \xi_j) r_{ij}^{-6}] \tag{3}$$

$$C(\xi_i, \xi_j) = \frac{1}{8}(1 + \xi_i + \xi_j + 5\xi_i\xi_j) \tag{4}$$

Where r_{ij} denotes the distance between i^{th} and j^{th} residue of the chain. For an AA pair, $C(\xi_i, \xi_j) = 1$ regarded as strongly attracting for an AB or BA pair while $C(\xi_i, \xi_j) = -0.5$, regarded as weakly repelling and for a BB pair, $C(\xi_i, \xi_j) = 0.5$, regarded as weakly attracting. Our objective is to find the minimum value of Eq. 1, representing lowest free energy of the structure of a protein.

3 Bees Algorithm (BA)

Bees Algorithm [19] is a swarm intelligence based algorithm inspired by the foraging behaviour of honey bees used for finding global optimum solution for a given optimization problem. Scout bees i.e. candidate solutions are randomly generated in the search space and the quality of the visited locations depend on the fitness value. The generated solutions are ranked and other bees are recruited from the neighbourhood having the highest ranking locations on the search space. This algorithm locates the most promising solutions and selectively explores their neighbourhoods looking for the global minimum of the fitness function.

The population contains N number of scout bees which are randomly scattered with uniform probability across the search space. Therefore, the j^{th} element of the i^{th} solution X_i is expressed using Eq. 5.

$$X_i^j = X_{min}^j + \left(X_{max}^j - X_{min}^j\right) \times rand(0, 1), j = 1, 2, ..., D \tag{5}$$

Where X_{min}^j and X_{max}^j denote the lower and upper bound of the j^{th} element and D denotes the dimension of any X_i. Each scout bees evaluates using the fitness function. After initialization of the scout bees, BA enters into a cycle which is composed of four phases [19]. Following phases are executed sequentially until stopping condition is met.

3.1 Waggle Dance

Say, N number of visited sites are ranked based on fitness information and B number of sites with highest fitness (i.e. minimum measure) are selected for local search. The local search is performed by other bees (foragers) that are directed to the neighbourhood of the selected sites. Each scout bees that are returned from one of the B best sites performs the 'waggle dance' to recruit nest mates for local search. The scout bees visit the first E elite (top-rated) sites among the B sites by recruiting E_r bees for neighbourhood search. The remaining $(B - E)$ sites that visited by the scouts recruit $B_r \leq E_r$ bees for neighbourhood search.

3.2 Local Search

For each of the B selected sites, the recruited bees are randomly placed in a neighbourhood of the high fitness location marked by the scout bees. This neighbourhood is defined as an D-dimensional hyper box of sides $a_1, a_2, ..., a_D$, centred on the scout bee. For each neighbourhood, the fitness is evaluated by the recruited bees. If one of the recruited bees lands in a position of higher fitness than the scout bee then the recruited bee is treated as the new scout bee. At the end of the local search, only the fittest bee is retained. The fittest solution visited so-far is therefore, considered as a representative of the whole neighbourhood.

3.3 Global Search

In the global search phase, $(N - B)$ number of bees are placed according to Eq. 5 across the search space for new solution. Random scouting represents the exploration capability of the BA.

3.4 Population Update

At the end of each cycle, the population is updated from two groups. The first group comprises the B scout bees which are associated with the best solution of each neighbourhood and represents the results of the local exploitative search. The second group is composed of the $(N - B)$ no. of scout bees associated with a randomly generated solution and represents the results of the global explorative search.

4 Adaptive Polynomial Mutation Based BA (APM-BA)

BA was originally designed as a numerical optimization technique based on foraging behaviour of honey bees and proved its robustness and efficiency to solving non-linear, real-valued function optimization problems. However, when dealing with multimodal problems, quality of solution is affected as the number of iterations increases and it suffers from premature convergence. The situation occurs when all the best visited sites converge in a small region of the search space, forcing them to converge to the global best point found so far which is not a global optima. BA sometimes suffers from premature convergence due to many local minima in the search space. In general, convergence is a desirable property that recruited bees of best visited sites and allowed to search near the global minimum as time progresses. Unfortunately, in the context of many local minima, the scout bees of the best visited sites are trapped in one of the local minima and fail to explore more promising neighbouring minima. To enhance the exploration capability and to avoid being trapped into local optima, a mutation strategy is necessary to increase the diversity of the best scout bees in the search space. With this observation, an adaptive polynomial mutation based bees algorithm (APM-BA) has been proposed in this paper to prevent premature convergence. We define a neighbourhood structure of each of the say, B selected site in the local search processes of APM-BA. For example, the j^{th} component of i^{th} selected site creates neighbourhood by Eq. 6.

$$X_i^j = X_i^j + rand(-2, 2) \tag{6}$$

We assume a parameter *trial* representing the number of iterations lead to inefficient search before better position is derived. If the i^{th} best scout bee finds a better site, *trial(i)* is set to zero; otherwise, it is incremented by one for the next iteration. However, the searching competence of a best scout bee should not be evaluated by the quality of its current site i.e. the fitness value but by the efficiency of current search i.e. by the *trial* counter. Finally to avoid premature convergence, we employed adaptive polynomial mutation on i^{th} best scout bee when a specific number of times the i^{th} best scout bee cannot improve its current position.

4.1 Adaptive Polynomial Mutation (APM)

In adaptive polynomial mutation strategy [21], the j^{th} dimension of a i^{th} candidate solution X_i is mutated with polynomial mutation as expressed in Eq. 7.

$$X_i^j(t+1) = X_i^j(t) + \left(X_{max}^j - X_{min}^j\right) \times \delta_j \tag{7}$$

Where t represents current iteration number, X_{max}^j and X_{min}^j are the upper and lower bound of j^{th} component of X_i while δ_j represents the polynomial function calculated using Eq. 8.

$$\delta_j = \begin{cases} (2r_j)^{\frac{1}{(\eta_m+1)}} - 1 & r_j < 0.5 \\ 1 - [2(1-r_j)]^{\frac{1}{(\eta_m+1)}} & r_j \geq 0.5 \end{cases} \tag{8}$$

η_m is the polynomial distribution index and r_j represents uniformly distribute random number in (0,1). The probability of δ_j is calculated using Eq. 9.

$$P(\delta_j) = 0.5(\eta_m + 1)(1 - |\delta_j|)^{\eta_m} \tag{9}$$

By varying η_m, the perturbation can be varied in the mutated solution. If the value of η_m is large, a small perturbation of a variable is achieved. To achieve gradually decreasing perturbation in the mutated solutions, the value of η_m is gradually increased. The following rule presented in Eq. 10 is applied to achieve the proposed adaption policy known as adaptive polynomial mutation.

$$\eta_m = (80 + t) \tag{10}$$

To improve the performance of BA, we used adaptive polynomial mutation on ith best scout bees if the *trial(i)* > *D*. The mutated solution mX_i^j is obtained by Eq. 11.

$$mX_i^j = X_i^j + (X_k^j - X_i^j) \times \delta_j \tag{11}$$

In Eq. 11, the i^{th} best scout bee exchange information with the k^{th} one in its j^{th} component where $k \neq i$. If $f(mX_i) \leq f(X_i)$, then i^{th} best scout bee X_i is replaced by mX_i and the *trial* counter is set to 0 i.e. *trial(i)* = 0.

The pseudo-code of APM-BA for protein structure prediction is given in Algorithm 1. Since, protein structure prediction is a minimization problem, fitness values are ranked in ascending order.

5 Experiments and Results

In order to evaluate the performance of the proposed algorithm, the experiments are performed on both artificial and real protein sequences for protein structure prediction in 2D AB off-lattice model.

5.1 Artificial Protein Sequence

Fibonacci sequence known as artificial protein sequence considered usually as benchmark for the protein structure prediction problem in AB off-lattice model [22]. A Fibonacci sequence is defined recursively by

$S_0 = A,\ S_1 = B,\ S_{i+1} = S_{i-1} * S_i$

Where $'*'$ is the concatenation operator. The first few sequences are $S_2 = AB$, $S_3 = BAB$, $S_4 = ABBAB$ and so on. Hydrophobic residue *'A'* occurs in isolation along the chain, while hydrophilic residue *'B'* occurs either isolated or in pairs and the

Algorithm 1 APM-BA Algorithm

```
1: Initialize the parameters of BA, Maximum iterations $G$, Number of candidate solutions $N$,
   Dimensions $D$ and the inefficient trial counter $trial(i) = 0$.
2: Initialize Population $X$ by Eq. 5 and evaluate fitness $f(x)$
3: Ranked fitness values $f(x)$ and $X$
4: for $iter = 1$ to $G$ do
5:    for $i = 1$ to $E$ do
6:       for $j = 1$ to $E_r$ do
7:          Generate neighbourhoods according to Eq. 6
8:       end for
9:       Select best neighbourhood, $X_{ngh}$
10:      if $f(X_{ngh}) < f(X_i)$ then
11:         update $X_i$ and $f(X_i)$
12:         $trial(i) = 0$
13:      else
14:         $trial(i) = trial(i) + 1$
15:      end if
16:      if $trial(i) > D$ then
17:         generate $mX_i$ according to Eq. 11
18:         if $f(mX_i) < f(X_i)$ then
19:            update $X_i$ and $f(X_i)$
20:            $trial(i) = 0$
21:         end if
22:      end if
23:   end for
24:   for $i = 1$ to $B - E$ do
25:      for $j = 1$ to $B_r$ do
26:         repeat lines from 7 to 22
27:      end for
28:   end for
29:   for $i = 1$ to $N - B$ do
30:      generate according to Eq. 5 and evaluate fitness $f(X_i)$
31:   end for
32:   Ranked fitness values $f(x)$ and $X$
33: end for
34: Output the best solution
```

molecules have a hierarchical string structure. Artificial protein sequence lengths are obtained through the Fibonacci numbers $n_{i+1} = n_{i-1} + n_i$. In the experiment, we have considered artificial protein sequence given in Table 1 with different lengths, say 13, 21 and 34.

Table 1 Artificial protein sequences

Artificial Protein	Sequence	Length
S_{13}	*ABBABBABABBAB*	13
S_{21}	*BABABBABABBABBABABBAB*	21
S_{34}	*ABBABBABABBABBABABBABABBABBABABBAB*	34

5.2 Real Protein Sequence

In order to measure the effectiveness of the APM-BA algorithm in predicting real protein structures, we select protein sequences from Protein Data Bank (PDB, http://www.rcsb.org/pdb/home/home.do). The PDB ID of these protein sequences are 1BXP, 1BXL, 1EDP and 1EDN respectively. The sequence information of these real proteins are given in Table 2. In the experiment, the same K-D method used in the literature [23] is adopted to distinguish the hydrophobic and hydrophilic residues of 20 amino acids in real protein sequences. The amino acids I, V, L, P, C, M, A, G are considered as hydrophobic *(A)* residues and D, E, F, H, K, N, Q, R, S, T, W, Y are hydrophilic *(B)* residues.

Table 2 Real protein sequences

Real Protein	Sequence	Length
1BXP	MRYYESSLKSYPD	13
1BXL	GQVGRQLAIIGDDINR	16
1EDP	CSCSSLMDKECVYFCHL	17
1EDN	CSCSSLMDKECVYFCHLDIIW	21

5.3 Parameter Settings and Initialization

The proposed APM-BA algorithm is compared with simple bees algorithm [20] and the algorithms which are already used for protein structure prediction such as CPSO [10], EPSO [11], IF-ABC [18] in 2D off-lattice model. All experiments are implemented in MATLAB R2010a and executed on an Intel Core (TM) I7-2670 QMCPU running at 2.20 GHz with 8 GB of RAM with Windows XP. The independent experiments of each algorithm is repeated 30 times with same initial population. The population size (N) for all approaches is fixed at 50 but the dimension D is different based on the respective length of protein sequences. The stopping criteria is same for all algorithms based on number of iterations (G). The number of iterations is defined [24] by $G = (D \times 10000)/N$. The parameters of the proposed algorithm are given in Table 3. The best of scout bees are going through adaptive polynomial mutation when *trial* counter of each of these scout bees are greater than D.

Table 3 APM-BA learning parameters

Parameters	Values
Number of scout bees (N)	50
Number of elite sites (E)	10
Number of best sites (B)	20
Number of recruited bees for elite sites (E_r)	10
Number of recruited bees for remaining best sites (B_r)	5

5.4 Results for Artificial Protein Sequences

Table 4 list the mean and standard deviation (Std.Dev) of 30 runs obtained by the APM-BA along with the results obtained by CPSO, EPSO, IF-ABC and BA for comparison. From Table 4, we can see that minimum energy obtained by the proposed algorithm dominates all other algorithms for every artificial protein sequences. Strong significant improvement has been observed in case of protein sequence length 21 and 34 by APM-BA. Three artificial protein sequences are placed in 4^{th}, 3^{rd} and 2^{nd} position with respect to standard deviation by the APM-BA which measures robustness of the minimum energy, as shown in Table 4. Therefore, the proposed algorithm out performs than other algorithms as increase the length of artificial protein sequence. The convergence characteristics of each algorithm on artificial protein sequences of lengths 13, 21 and 34 are shown in Fig. 2. The APM-BA exhibits better convergence than other approaches.

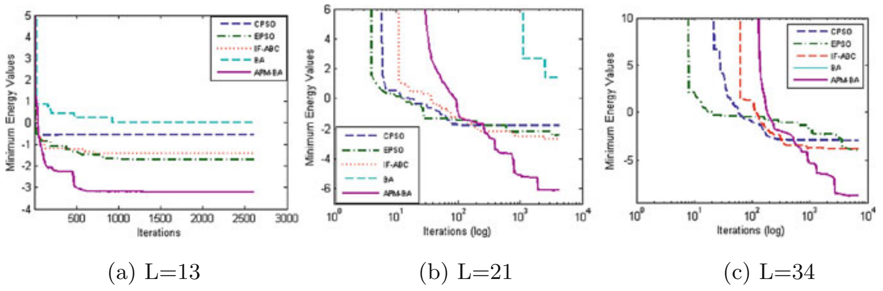


Fig. 2 Convergence graph of artificial protein sequence with different length

5.5 Results for Real Protein Sequence

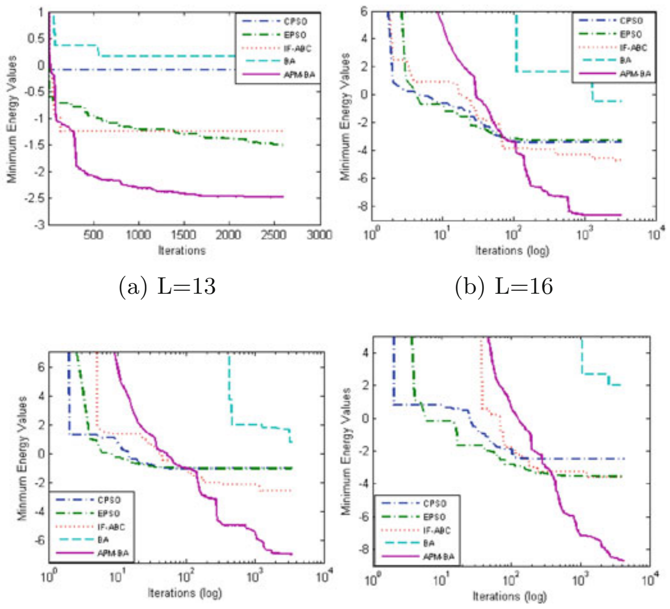
The results obtained by the APM-BA for real protein sequences are summarized in Table 5, along with the results of other algorithms used for comparison. It has been observed that lowest energy obtained by the proposed algorithm is significantly better than that of other algorithms like CPSO. EPSO, IF-ABC and BA. Therefore, APM-BA is superior in solving real protein sequences. Based on standard deviation for all real protein sequences compare to other algorithm, the proposed approach is highly robust compare to other algorithms. The convergence characteristics of the algorithms on real protein sequences are plotted in Fig. 3.

Table 4 Results for artificial protein sequence

Artificial protein	CPSO		EPSO		IF-ABC		BA		APM-BA	
	Mean	Std.Dev	Mean	Std.Dev	Mean	Std.Dev	Mean	Std.Dev	Mean	Std.Dev
S_{13}	−1.653	0.518	−1.974	0.346	−1.846	0.190	0.020	0.228	−2.596	0.370
S_{21}	−3.352	0.741	−3.400	0.432	−3.062	0.264	1.491	0.850	−4.964	0.601
S_{34}	−5.284	1.302	−5.671	0.910	−4.586	0.363	2397.448	2845.384	−6.694	0.701

Table 5 Results for real protein sequence

Real protein	CPSO		EPSO		IF-ABC		BA		APM-BA	
	Mean	Std.Dev	Mean	Std.Dev	Mean	Std.Dev	Mean	Std.Dev	Mean	Std.Dev
1BXP	-1.893	0.432	-2.110	0.250	-1.517	0.162	-0.118	0.349	-2.335	0.085
1BXL	-6.296	1.062	-6.255	1.161	-5.333	0.359	-0.696	1.087	-8.002	0.592
1EDP	-3.092	1.593	-3.773	1.438	-3.093	0.386	0.689	0.614	-6.230	0.518
1EDN	-4.232	1.325	-5.005	0.812	-4.232	0.482	2.135	1.058	-7.275	1.078



(a) L=13 (b) L=16

(c) L=17 (d) L=21

Fig. 3 Convergence characteristics of real protein Sequence with different length

6 Conclusions

In this paper, an improved BA revised by adaptive polynomial mutation strategy is introduced to optimize protein structure prediction in 2D AB off-lattice model. In APM-BA, adaptive polynomial mutation is applied to the best scout bees based on their inefficiency during the search processes. The proposed strategy is able to preventing stuck at local optima and made exploration on the search space. Experimental results confirm that APM-BA is significantly more effective for protein structure prediction problem with respect to artificial and real protein sequences. It should be noted that our study concerns few number of protein sequences with smaller lengths.

Predicting structure of more real protein sequences with larger lengths in 3D AB off-lattice model and investigations on the neighbourhood structure as well as the selection procedure of best visited sites of BA will be our future work.

References

1. Freitas, A.A., Wieser, D.C., Apweiler, R.: On the importance of comprehensible classification models for protein function prediction. IEEE/ACM Trans. Comput. Biol. Bioinform **7**(1), 172–182 (2010)
2. May, A., Pool, R., Dijk, E.V., Bijlard, J., Abeln, S., Heringa, J., Feenstra, K.A.: Coarse-grained versus atomistic simulations: realistic interaction free energies for real proteins. Bioinformatics **30**(3), 326–334 (2014)
3. Anfinsen, C.B.: Principles that govern the folding of protein chain. Science **181**(4096), 223–230 (1973)
4. Pierce, N.A., Winfree, E.: Protein design is np-hard. Protein. Eng. **15**(10), 779–782 (2002)
5. Rossi, G., Ferrando, R.: Searching for low-energy structures of nanoparticles: a comparison of different methods and algorithms. J. Phy. Condens. Matter **21**(8), 84208 (2009)
6. Dorn, M., e Silva, M.B., Buriol, L.S., Lamb, L.C.: Three-dimensional protein structure prediction: methods and computational strategies. Comput. Biol. Chem. **53**, 251–276 (2014)
7. Dill, A.K., Bromberg, S., Yue, K., Fiebig, K.M., Yee, D.P., Thomas, P.D., Chan, H.S.: Principle of protein folding: a perspective from simple exact models. Protein Sci. **4**(4), 561–602 (1995)
8. Stillinger, F.H., Head-Gordon, T., Hirshfel, C.L.: Toy model for protein folding. Phys. Rev. **48**(2), 1469–1477 (1993)
9. Kim, S.Y., Lee, S.B., Lee, J.: Structure optimization by conformational space annealing in an off-lattice protein model. Phys. Rev. **72**(1), 011916 (2005)
10. Liu, J., Wang, L., He, L., Shi, F.: Analysis of toy model for protein folding based on particle swarm optimization algorithm. In: Wang, L., Chen, Ke, S. Ong, Yew (eds.) ICNC 2005. LNCS, vol. 3612, pp. 636–645. Springer, Heidelberg (2005)
11. Zhu, H., Pu, C., Lin, X., Gu, J., Zhang, S., Su, M.: Protein structure prediction with epso in toy model. In: Second International Conference on Intelligent Networks and Intelligent Systems, 2009. ICINIS '09, pp. 673–676 (2009)
12. Liu, J.F., Xue, S.J., Chen, D.B., Geng, H.T., Liu, Z.X.: Structure optimization of the two-dimensional off-lattice hydrophobichydrophilic model. J. Biol. Phys. **35**(3), 245–253 (2009)
13. Cheng-yuan, L., Yan-rui, D., Wen-bo, X.: Multiple-layer quantum-behaved particle swarm optimization and toy model for protein structure prediction. In: 2010 Ninth International Symposium on Distributed Computing and Applications to Business Engineering and Science (DCABES), pp. 92–96 (2010)
14. Kalegari, D.H., Lopes, H.S.: A differential evolution approach for protein structure optimization using a 2d off-lattice model. J. Bio-Inspired Comput. **2**(3), 242–250 (2010)
15. Mansour, R.: Applying an evolutionary algorithm for protein structure pre-diction. Am. J. Bioinf. Res. **1**(1), 18–23 (2011)
16. Jana, N.D., S, Jaya: Hybrid particle swarm optimization technique for protein structure prediction using 2D off-lattice model. In: Panigrahi, Bijaya Ketan, Suganthan, Ponnuthurai Nagaratnam, Das, Swagatam, Dash, Shubhransu Sekhar (eds.) SEMCCO 2013, Part II. LNCS, vol. 8298, pp. 193–204. Springer, Heidelberg (2013)
17. Jingfa, L., Sun, Y., Li, G., Song, B., Huang, W.: Heuristic based tabu search algorithm for folding two-dimensional ab off-lattice model proteins. Comput. Biol. Chem. **47**, 142–148 (2013)
18. Li, B., Li, Y., Gong, L.: Protein secondary structure optimization using an improved artificial bee colony algorithm based on ab off-lattice model. Eng. Appl. Artif. Intell. **27**, 70–79 (2014)
19. Pham, D., Castellani, M.: The bees algorithmmodelling foraging behaviour to solve continuous optimisation problems. J. Bio-Inspired Comput. **223**(12), 2919–2938 (2009)

20. Pham, D., Castellani, M.: Benchmarking and comparison of nature-inspired population based continuous optimization algorithms. Soft comput. **18**, 871–903 (2014)
21. Saha, A., Datta, R., Deb, K.: Hybrid gradient projection based genetic algorithms for constrained optimization. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2010)
22. Stillinger, F.H.: Collective aspects of protein folding illustrated by a toy model. Phys. Rev. E **52**(3), 2872–2877 (1995)
23. Mount, D.W.: Bioinformatics: Sequence and Genome Analysis. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York (2001)
24. Liang, J.J., Qu, B.Y., Suganthan, P.N., Hernandez-Diaz, A.G.: Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. Technical report DAMTP 2000/NA10, Nanyang Technological University, Singapore (2013)

Limited Randomness Evolutionary Strategy Algorithm

Tomas Brandejsky

Abstract Herein presented paper is denoted to study of real requirements of evolutionary algorithm to random number generator properties. In the past years novel studies occurred. These studies pointed that in some situations random number generator might be replaced by deterministic chaos system. The goal of presented paper is to point the significant properties of number generator, to extend the class of systems to use on its place. During preparation of the paper experiments with Evolutionary Strategy algorithm were done and as the test- bed problems of identification of parameters of two deterministic chaos systems were used. Namely, these systems were Lorenz and Rabinovich-Fabricant ones. The conclusion of the paper is, that periodic functions might be used if proper parameters and sampling period of number generating function replacing random number generator are chosen. This result is not so interesting from practical viewpoint, because the application of *sin(x)* function is slower than standard *rand()* function of C and C++ language, but it points that evolutionary algorithms do not require randomness as the source of its capabilities.

Keywords Genetic algorithm • Evolutionary strategy • Random number generator • Efficiency • Optimization

1 Introduction

For many years researchers discusses influence of random number generator into speed of evolutionary process. There it is hard to recognize significant behaviors and features of the optimal random number generator [1–3]. The work [1] studies influence of choice of different pseudo-random generators onto performance of selected genetic algorithms. The works [2, 3] discuss impact of the Random Number Generator quality on Particle Swarm Optimization Algorithm. PSO algorithms are not GA or ES

T. Brandejsky (✉)
Faculty of Transportation Sciences, CTU in Prague, Prague, Czech Republic
e-mail: brandejsky@fd.cvut.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_5

algorithms, but they also strongly depend on randomness. On the opposite side, they are not applying natural selection principle. There is accessible study of random number generator to genetic algorithm produced by another authors group too in [4].

Studies concluding the possibility to replace the random number generator by deterministic chaos system [15, 16] are now in works [5–8]. But the pseudo-random number generators are only long period functions with specific properties; especially they have constrained magnitudes and large number of crossing of any value in the output interval during one period. Typically, this requirement is known in the much stronger form as requirement of uniformity.

There is also the second significant requirement – requirement of independence, which tells that the generated numbers has no correlation with each other. Thus there is correct to form question if it is possible to replace random number generator by any non periodic or long period function?

It is also interesting to mention that any periodical continuous function might be transformed into non periodic discrete one if the length of period is not integer multiple of sampling period. This is the alternative way how to satisfy the third requirement – maximal cycle length. Maximal cycle length is significant especially in situations when large populations, big number of evolutionary steps and highly dimensional problems occurs. This fact might be demonstrated e.g. by increasing popularity of Mersenne-Twister algorithm [9]. This algorithm has extremely long period, e.g. its implementation MT19937 has period $2^{19937} - 1$. Applicability of this algorithm in evolutionary algorithms is presented e.g. in [10].

Problem of non periodic functions created from periodic ones is that they satisfy requirement of independence only partially, in contrary to some deterministic chaos systems.

On the base of this background, the idea of evolutionary system not using random number generator nor even deterministic chaos system was formulated. As the test bed, simple evolutionary strategy algorithm described in the Chap. 2 was used. The random number generator was replaced by $\sin(kx)$ function, as it is described in the Chap. 3. As test cases, the identification of parameters of equations describing Lorenz attractor and Rabinovich-Fabrikant system is applied. The results are outlined in the Chaps. 4 and 5.

2 Used Evolutionary Strategy Algorithm

The simple evolutionary strategy algorithm is outlined at Fig. 1. It is used as test bed in herein presented experiments for its simplicity and well known behavior.

n	number of individuals
S	maximal number of iterations
X	vector of individuals
X'	vector of new population candidates
$f(x)$	fitness function
Errlimit	Required error magnitude

```
Given $n \in N_+$ , s
Initialize $x_k$ , $k = 0$
For $k \in \{1....,\lambda\}$ Do
  Evaluate $f(x_k)$
End For
Sort $\{x_k\}$
Cycle=0;
While $f(x_1)$>Errlimit AND k<s Do
  For $k \in \{1....,\lambda\}$ with step 2 do
    Randomly select between mutation and crossover
      Do $(x_k^/, x_{k+1}^/) \leftarrow mutate(x_k), mutate(x_{k+1})$
      Or
      Do $(x_k^/, x_{k+1}^/) \leftarrow crossover(x_k, x_{k+1})$
  End For
  For $k \in \{1....,\lambda\}$ Do
    Evaluate $f(x_k^/)$
    If $f(x_k^/) < f(x_k)$ Then $x_k \leftarrow x_k^/$
  End For
  Cycle=Cycle+1
End While
```

Fig. 1 Used evolutionary strategy algorithm

3 Replacement of Random Number Generator by Non Periodic or Long-Periodic Functions

Random number generators, both natural and artificial, described by specific equations should satisfy the following properties:

- *Uniformity:* The numbers generated appear to be distributed uniformly on interval <0, 1>;
- *Independence:* The numbers generated show no correlation with each other within interval of given cycle length;
- *Cycle length:* It should take long before numbers start to repeat.

Above mentioned works, especially [5–8] demonstrate that it is possible to replace random number generator by deterministic chaos system equations. These results are significant, because studied systems do not guarantee the property of uniformity in common. So, there occurs question, if it is needed to use deterministic chaos system equations or if it is possible to extend the group of applicable equations? Before answering of this question, it is useful to reason why evolutionary algorithms need to use random number generators?

The evolutionary algorithms use the random numbers in two different situations – initialization population formulation and processing of evolution. The first case points especially uniformity of generated numbers. The second one uses random numbers to achieve diverse modifications, to test big number of different possible solutions if the number of evolutionary cycles is high. Thus, the cycle length is significant, because it is the parameter giving the chance to test more values, more solution candidates.

Probably, there is no reason that avoids us to replace random number generator by any long periodic or non-periodic function. Even it is possible to sample any periodic continuous function defined on domain of real numbers and analogous infinite co-domain such way, that resulting discrete sequence of numbers will have endless period. E.g. if $sin(x)$ with period of 2π will be sampled with period 1, we obtain such infinite non-periodic sequence because there does not exist any integers n and m different from 0, that $n\pi = m$. The next chapters will be denoted to testing of applicability of two such functions (1) and (2) in above described evolutionary strategy algorithm on the place of standard random number generator. As test examples, the identification of parameters of non-linear differential equations of deterministic chaos will be used. These equations will represent Lorenz attractor (3) described by [11] and Rabinovich-Fabrikant system (4) described in [12]. Identification means in the case of Lorenz attractor determining of parameters σ, ρ, *and* β magnitudes to fit the modeled data. In the case of Rabinovich-Fabricant system these parameters are $\alpha\, and\, \gamma$. Training data set consisted in the case of Lorenz attractor 400 samples and 2000 samples in the case of Rabinovich-Fabricant system. These two systems have been chosen for its nonlinearity, which forces the need of uniformity and independence of the tested generator and the transformation of Rabinovich-Fabricant system into more complicated form (6) increases the significance of cycle length requirement. Bigger number of parameters with similar influence to final behavior of the identified equation complicated identification and thus increases needed number of evolutionary algorithm cycles. Increasing of this number forces the need of longer cycle of number generator together with the larger genes (more parameters are optimized simultaneously, thus the larger number of magnitudes must be generated by the generator).

$$\sin(kn) \tag{1}$$

$$\sin(a\ \ \sin(nx)) \tag{2}$$

The property of this function is illustrated by Fig. 2.

$$\begin{aligned} x' &= \sigma(\mathrm{y} - \mathrm{x}), \\ y' &= \mathrm{x}(\rho - \mathrm{z}) - \mathrm{y}, \\ z' &= \mathrm{xy} - \beta\mathrm{z} \end{aligned} \tag{3}$$

$$\begin{aligned} x' &= y(z-1+x^2)+\gamma x \\ y' &= x\left(3z+1-x^2\right)+\gamma y \\ z' &= -2z(\alpha+xy) \end{aligned} \tag{4}$$

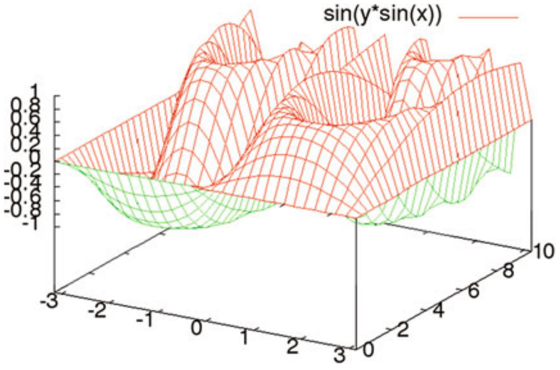


Fig. 2 Property of sin(y sin(x)) function

4 Identification of Lorenz Attractor Equations Parameters

Average number of the used Evolutionary Strategy algorithm in the case of population of 1000 individuals (this number of individuals is used in all experiments presented in this contribution) for identification of Lorenz equations parameters σ and β are outlined in Table 1 for each variable x, y and z, when standard C++ *rand()* and *sin()* number generators are used. All average results were obtained by 1000 times repetition of experiments:

Table 1 The average number of cycles of Lorenz attractor equations parameter identification depending on different number generators

Function category variable	Rand()	sin(kn)	sin(sin(kn))	sin(10sin(kn))
X	14.4	42.1	11	13.3
Y	14	60	12.2	16.3
Z	14.6	58.3	18.3	14.3

The Fig. 3 presents numbers of iterations when standard *rand()* function is replaced by *sin*(k x) function. These numbers are also similar for all three variables x, y and z, but they are approximately five times worse than for standard rand function.

Figures 4 and 5 then display the analogous dependency for random number generator in the form of Eq. (2) with parameter a equal to 1 and 10 respectively.

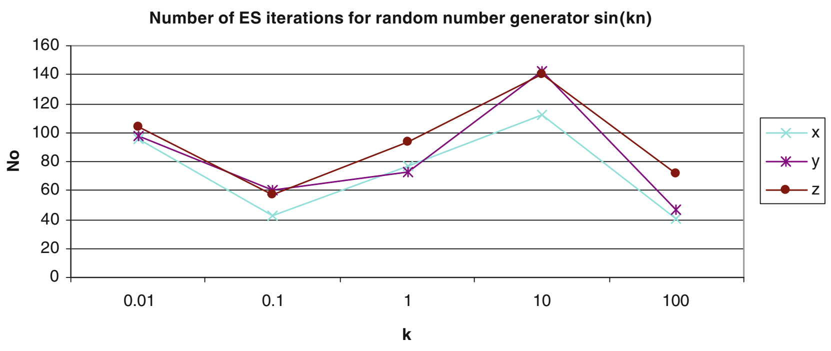


Fig. 3 Number of iterations for x, y and z variables, *sin(k x)* is used on the place of random number generator

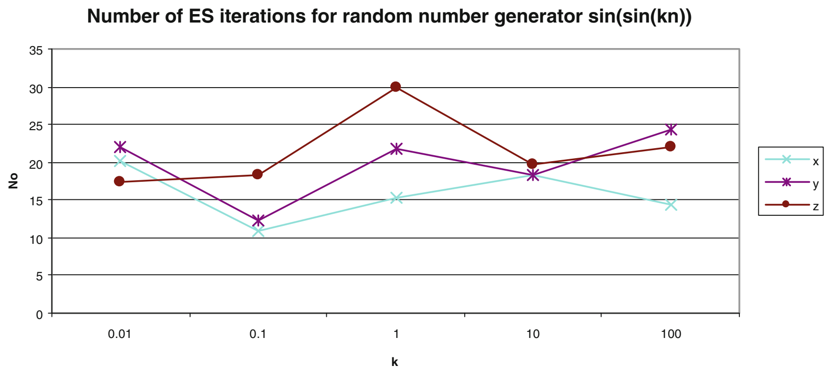


Fig. 4 Number of iterations for x, y and z variables, $sin(sin(k\ x))$ is used on the place of random number generator

Above presented results demonstrate that function (2) gives better results than (1), as it is summarized in Table 1. This result is in relation to self-correlation of time series produced by rand function and $sin(t)$ function (rand gives much smaller than $sin(kn)$, but $sin(sin(t))$ and especially function $sin(sin(10t))$ gives competitive results), but the Lorenz system consist of equations where is only one parameter to identify. Self-correlation is measure of independence condition of number generator.

The $sin(a\ sin(kn))$ function gives results comparable to *rand()* function while $sin(kn)$ produces significantly worse convergence of ES algorithm. It is probably given by higher self-correlation of $sin(x)$ function in comparison to the rest ones.

Table 2 presents correlations of $sin(akt)$ and $sin(ak(t + 1))$ functions and Table 3 outlines correlations of $sin(sin(kt))$ and $sin(sin(k(t + 1))$ or $sin(sin(10\ kt))$ and $sin(sin(10\ k(t + 1))$ functions.

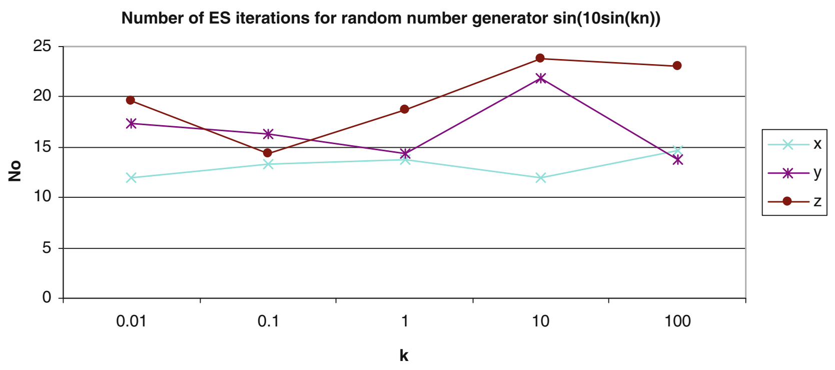


Fig. 5 Number of iterations for *x*, *y* and *z* variables, *sin*(10*sin*(*k x*)) is used on the place of random number generator

Table 2 The self-correlation of *sin*(*akt*) and *sin*(*ak*(*t* + *1*)) functions

Function	sin(0.01t)	sin(0.1t)	sin(t)	sin(10t)	sin(100t)
correlation	0.9999973	0.99536378	0.54634896	−0.839216	0.86546618

Table 3 The correlations of *sin*(*sin*(*kt*)) and *sin*(*sin*(*k*(*t* + 1)) or *sin*(*sin*(10 *kt*)) and *sin*(*sin*(10*k* (*t* + 1)) functions

Functions	sin(sin(kt)) and sin(sin(k(t + 1)))	sin(sin(10kt)) and sin(sin(10k(t + 1)))
correlation	0.54388688	−0.126774177

5 Identification of Rabinovich-Fabrikant System Attractor Parameters

Problem of n-dimensional system is the problem of independence of random time sub-series influencing each particular parameter. In the case on multidimensional system the independence is required in more complex manner than in the case of one dimensional one.

Def. 1: Let is given reasoned number of samples *n*. Let used Evolutionary algorithm optimizes *m* dimensional problem and it need 1 random number to each parameter to be optimized. Symbol r_i denotes i-th number of time series generated by random number generator or alternative function. Then there is need to investigate mutual independence and self-independence (independence of data taken in *t* and *t* + *d*) of all following data sub-series separated from the original series r:

$$\forall k \in \langle 1, m \rangle, \forall l \in \langle 0, m-1 \rangle, \forall i \in \langle 1, n\ div\, m \rangle, \quad s_{k,l,i} = k + l, \ldots ik + l. \tag{5}$$

Rabinovich-Fabrikant system is described by three equations as Lorenz one, but these equations are strongly non-linear, where e.g. the first equation contains expression x^2y and the second one contains x^3 one. These equations were transformed into the form (6), which is much complicated to ES algorithm (they contains additional parameters $\alpha_1, \ldots, \alpha_4$):

$$\begin{aligned} x' &= y(z - \alpha_1 + x^2) + \gamma x \\ y' &= x\left(\alpha_2 z + \alpha_3 - x^2\right) + \gamma y \\ z' &= -\alpha_4 z(\alpha + xy) \end{aligned} \tag{6}$$

It means that these equations contain 2, 3 and 2 parameters respectively in contrary to Eq. (3), where each differential equation contains one parameter to be identified only. Such situation is avoided by GPA algorithms frequently because these algorithms have no preventive mechanism against creation of overcomplicated structures. The solution of this problem is difficult and it is partially solved e.g. by GPA-ES algorithm [13].

For each number generator it was the most difficult to identify the second equation (equation describing variable y) with tree parameters. The best results for each category of functions are summarized in Table 4.

Table 4 The average number of cycles of Rabinovich – Fabrikant equations parameters identification depending on different number generators

Function category variable	rand()	sin(kn)	sin(sin(kn))	sin(10sin(kn))
X	24.6	69.1	62.4	46.9
Y	24017	7776.2	16052.4	27679.5
Z	35.9	79.3	69.8	75.6

These results point interesting fact that for identification of the first and last equations, the *rand()* function is the best random number generator, but from the viewpoint of the second one $sin(k\ n)$ or $sin(sin(k\ n))$ are better, non looking that both the first and the second equations both contain nonlinearities in the form of multiplication of three variables and two or three parameters respectively. To colorate this paradox, it is need to mention previous work [14], where it is presented, that symbolic regression of expression x^3 is much more complicated than regression of other expression occurring in Eq. (5). In fact, in herein discussed case the increase of needed computational complexity is given by presence of three parameters which masks each other.

6 Conclusion

Presented paper discusses the possibility to replace standard random number generator by continuous periodic function. This replacement is possible, when the requirements of uniformity, independence and cycle length are satisfied. Such functions as $sin(kt)$ or $sin(a\ sin(kt))$, which solve as examples in this paper might be carefully applied, but the results are applicable to much wider group of functions. Analysis of the next types of functions and the next function features influencing the speed of convergence of evolutionary algorithms will be subjects of future works.

The conclusion of above presented work is that nonstandard number generators as $sin(kt)$ function and composed $sin(a\ sin(kt))$ are not significantly worse than standard *rand()* random number generator in presented situation. There is chance, that the random number generators may be replaced by any continuous aperiodic function satisfying conditions of uniformity, independence and (maximal) cycle length in application in evolutionary algorithms. Future tests with other functions with smaller self-correlation will give information about its influence to evolutionary algorithm behaviors. Presented short work is not able to give complete answer on the question, which random number generator parameters are significant to evolutionary algorithms, but it points to above mentioned tree conditions. On the opposite side, while in the case of single parameter equations describing Lorenz attractor, the $sin(k\ sin(nx))$ functions were comparable to *rand()* generator, in the case of Rabinovich-Fabrikant system these functions were worse than *rand()* generator. Thus there is probably undiscovered some more deep, or it is possible to say more complicated, property than above discussed three conditions.

References

1. Cantú-Paz, E.: On random numbers and the performance of genetic algorithms. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 9–13. New York, USA, Morgan Kaufmann (July 2002)
2. Bastos-Filho, C.J.A., Oliveira Junior, M.A.C., Nascimento, D.N.O., Ramos A.D.: Impact of the random number generator quality on particle swarm optimization algorithm running on graphic processor units. In: Conference: 10th International Conference on Hybrid Intelligent Systems (HIS 2010), pp. 23–25. Atlanta, GA, USA (August 2010)
3. Rodgers, M.: Random Numbers and Their Effect on Particle Swarm Optimization. On_Line paper, http://ncre.ucd.ie/COMP30290/Crc2006/rodgers.pdf. Accessed 15 March 2015
4. Meysenburg, M.M., Foster, J.A.: The quality of pseudorandom number generators and simple genetic algorithm performance. In: Proceedings of the Seventh International Conference on Genetic Algorithms, pp. 276–281. Morgan Kaufmann (1997)
5. Senkerik, R., Davendra, D.D., Zelinka, I., Pluhacek, M., Kominkova-Oplatkova, Z.: Chaos driven differential evolution with lozi map in the task of chemical reactor optimization. Lecture Notes in Computer Science, vol. 7895, pp. 56–66. Springer (2013)
6. Senkerik, R., Davendra, D.D., Zelinka, I., Pluhacek, M., Kominkova-Oplatkova, Z.: On the differential evolution driven by selected discrete chaotic systems: extended study. In: Mendel

2013: 19th International Conference on Soft Computing, June, pp. 26–28, Brno, Czech Republic, Brno University of Technology, pp. 137–144 (2013)

7. Senkerik, R., Pluhacek, M., Davendra, D.D., Zelinka, I., Kominkova-Oplatkova, Z.: Chaos driven evolutionary algorithm: a new approach for evolutionary optimization. In: Proceedings of the 2013 International Conference on Systems, Control and Informatics (SCI 2013). Recent Advances in Systems, Control and Informatics, September, pp. 28–30, 2013, Venice, Italy, WSEAS Press, pp. 117–122 (2013)
8. Senkerik, R., Pluhacek, M., Kominkova-Oplatkova, Z.: Simulationof time-continuous chaotic systems for the generating of random numbers. In: Proceedings of the 18th International Conference on Systems (part of CSCC'14). Latest Trends on Systems—Volume II. Santorini Island, Greece, July, pp. 17–21, ISSN: 1790-5117, ISBN: 978-1-61804-244-6. pp. 557–561
9. Matsumoto, M.; Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**(1), 3–30 (1998)
10. Meysenburg, M.M., Hoelting, D., Mcelvain, D., Foster, J.A.: How random generator quality impacts genetic algorithm performance, pp. 480–487. In: Langdon, W.B., et al. (eds) GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan-Kaufmann, ISBN: 1-55860-878-8 (2002)
11. Lorenz, E.N.: Deterministic nonperiodic flow. J. Atmos. Sci. **20**(2), 130–141 (1963)
12. Rabinovich, M.I., Fabrikant, A.L.: Stochastic self-modulation of waves in nonequilibrium media. Sov. Phys. JETP **50**, 311 (1979)
13. Brandejsky, T., The use of local models optimized by genetic programming algorithm in biomedical-signal analysis. Handbook of optimization from Classical to Modern Approach. Springer, Heidelberg, pp. 697–716 (2012). ISBN: 978-3-642-30503-0
14. Brandejsky, T., Zelinka, I: Specific bahaviour of GPA-ES evolutionary system observed in deterministic chaos regression. In: Roesler, O.E., et al. (eds.) Nostradamus: Modern Methods of Prediction, Modeling and Analysis of Nonlinear Systems. Springer, Heidelberg, pp. 73–82 (2012)
15. Matousek, R., Minar, P.: Stabilization of chaotic logistic equation using HC12 and grammatical evolution. In: Zelinka, I., Chen, G., Rössler, O.E., Snasel, V., Abraham, A. (eds.) Nostradamus 2013: Prediction, Modeling and Analysis of Complex Systems, Advances in Intelligent Systems and Computing. Springer International Publishing, vol. 210, pp. 137–146. doi:10.1007/978-3-319-00542-3, ISSN: 2194- 5357
16. Matousek, R., Dobrovsky, L., Minar, P., Mouralova, K.: A note about robust stabilization of chaotic Hénon system using grammatical evolution. In: Zelinka, I., Suganthan, P.N., Chen, G., Snasel, V., Abraham, A., Rössler, O.E. (eds.) Nostradamus 2014: Prediction, Modeling and Analysis of Complex Systems, Advances in Intelligent Systems and Computing. Springer International Publishing, vol. 289, pp. 219–228. doi:10.1007/978-3-319-07410-6, ISSN: 2194- 5357

Data Mining Application on Complex Dataset from the Off-Grid Systems

Tomas Vantuch, Jindrich Stuchly, Stanislav Misak and Tomas Burianek

Abstract The Off-Grid systems with renewable sources of stochastic behavior's nature are facing to the issue of keeping the power quality parameters in national and international defined limits. This article aims on this problem by the application of data-mining. There is an attempt to uncover the influence between the variables of the system by synthesis of the classification rules. Next task that has to be solved and it is partly started in this article is to gain the robust strategy to keep this Off-grid systems stable and safety.

Keywords Power quality parameters · Genetic programming · Decision tree

1 Introduction

The Off-Grid Systems are defined as the independent energy units which are independent of the power supply from external grids. These systems are operated under different conditions than the regular energy units are have their specific characteristics. The low and variable short-circuit power as the one of their important parameters becomes an appreciable issue in this context [8, 9]. Its behavior is determined mainly by a character of a source part of Off-Grid systems because Off-Grid systems

T. Vantuch (✉) · T. Burianek
Department of Computer Science, VSB-Technical University of Ostrava,
17. Listopadu 15 708 33, Ostrava-poruba, Czech Republic
e-mail: tomas.vantuch@vsb.cz

T. Burianek
e-mail: tomas.burianek.st1@vsb.cz

J. Stuchly · S. Misak
Department of Electrical Power Engineering, VSB-Technical University of Ostrava,
17. Listopadu 15 708 33, Ostrava-poruba, Czech Republic
e-mail: jindrich.stuchly@vsb.cz

S. Misak
e-mail: stanislav.misak@vsb.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems
and Computing 378, DOI 10.1007/978-3-319-19824-8_6

use mainly renewable sources (RESs). These renewable energy sources have stochastic character of a power supply [13] and together with energy storage device are these power sources the only one source of a short-circuit power. The problem with their stochastic behavior is possible to solve by the energy storage device, whereas the power management can be controlled by using of the methods based on an artificial intelligence for example the DSM – demand side management [21].

Due to the variable and a low short-circuit power, these above mentioned algorithms are employed mainly to handle the power quality parameters (PQP) in requested limits, which are defined by the national and international standards and norms [1, 14].

There are no problems to keep the PQPs in requested limits in regular On-Grid systems, because the short-circuit power of On-Grid system is ca. 1000 times higher than in Off-Grid systems. The short-Circuit power in On-Grid systems is defined by a rotating machines and transformers operated at mega or gigawatts power level, whereas in Off-Grid system is defined by a energy storage devices (mostly battery bank) and RESs [17].

Due to the low short-circuit power can any change of source part's operation state affects the part of the power consumption. On the other hand any change of the operational state of a power consumption part can affects the PQPs of the source part too. By previously mentioned statements, it is clearly evident that Off-Grid system with all the variables and all the external and internal influences becomes to a complex issue to handle.

Because of this complexity, the data mining, analysis and forecasting methods are arising as the possible application. First of all, there is requirement to preprocess the data set by a subset selection technique [19] to maximize the data mining efficiency [3].

This paper deals with the Correlation Based Feature Selection [11, 12, 26] as an attribute's subset selector, because of its reasonable results [10]. For the task of the data mining there was employed two algorithms, the C4.5 implementation of the Decision Tree and the genetic programming. Both of them were focused to synthesize the hidden rules for the classification of the PQP's, which could confirm the original idea of the authors. The idea is to reveal the evidence of dependencies between meteorological variables, variables of the electric power consumption and power and power quality parameters.

The secondary motivation of this paper is to compare the accuracy and suitability of both applied data mining algorithms.

2 Experiment Design

The experiment was performed on data set contains more than one hundred attributes that were possibly related to previously mentioned PQPs. The part of the attributes is described in the appendix. The total count of the attributes was 118 and their values represents one minute measurements. The number of all observations was

8640 (minutes) which are in real conditions six days of continual measurement in the Off-Grid system.

The work-flow of this experiment consist of two phases, preprocessing and data mining.

As a simple preprocessing step, there was performed feature's sub set selection [19] in faith to chose the most suitable subset [3] for each of the PQPs. All the referenced attributes were firstly converted into nominal variables by their limits defined by the national and international standards. By this approach, there was created only two classes for each of the referenced variable, so first class "in" means that the value is not exceed the limits and the second class "out" means that the value of the variable exceed the limits.

In the second step, there were employed two widely known algorithms for data mining. Both of them were focused to synthesize the rules that could forecast the value of the power quality parameter by the values of the dataset's attributes. Of course there was available the final comparison to find out, which of the classifications was the most accurate.

2.1 Feature's Subset Selection

The quality of the data is one of the most important factors in data mining and decision making area [3, 19], because many data sets deals with keeping of redundant, noisy or irrelevant data. There are many approaches to solve this issue [15] and one of the most used is the features subset selection [10].

In this case, there was used Correlation-Based Feature Selection (CBFS) [11, 12] driven by Genetic Algorithm (GA) [4]. The combination of these two models met with success in previous studies [26].

Them main focus of the CBFS is to evaluate the subset by correlation analysis. The attributes that are correlated or strongly related to each other can bring redundant information for the data mining. On the other hand the uncorrelated variables with the referenced attributes are probably not caring enough information or can be noisy. The idea of this algorithm is that the ideal subset will contain only high correlated attributes to the referenced variables and the less possible correlation between each other in the subset.

The M_s is the heuristic "merit" of the subset and consists of the k as a number of features, the $\overline{r_{cf}}$ is the average feature-class correlation and $\overline{r_{ff}}$ means the average feature-feature inter-correlation.

$$M_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{1}$$

As it was mentioned before, there was applied the Genetic algorithm [4] as a search engine, to gain the best possible subset. The idea of Genetic algorithm is simply inspired by Darvin's theory of evolution and as a framework it is widely used.

GA consists of two major phases, breeding and selection. The primary entity in this process is an individual, representing one of the possible solution to the given problem. In this case the individual consists of the vector of attribute's IDs and represents one subset. The algorithm goes through iterations, where population of such individuals are firstly breed by cross-overs (one-point cross-over simple splits chosen individual's vector into two parts and one part is switched with other individual's part) and mutation (one id in the vector is randomly changed to any other valid id, that is not already contained) of the previous best individuals. On the next step there is a selection phase to choose the best candidates for coming crossovers and mutations or for the final output. For the selection purpose, there is a fitness function based on CBFS's merit to gain the best subset after defined count of iterations.

2.2 C4.5 Decision Tree

Decision tree is an induction algorithm widely described and used by machine-learning and applied statistic [24]. C4.5 is an extension of ID3 algorithm, that brings the ability of dealing with continuous attributes [5]. The model of building the tree is supervised learning where the input data (observations) and their classification (nominal variable) are given. The result of the algorithm is a set of conditions in the tree structure that is able to categorize each record of the observations to the given classes. Once the tree is obtained, it can be used for predicting the classification of the measurements by the given values, and this approach meets with a lot of applications [6, 16].

There are a several steps of constructing the decision tree, that are repeated until all observations are not correctly classified:

1. For each attribute of the data set do an evaluation and compute the information gain and select best splitting attribute X.
2. If X is categorical with x values:
   - Split the dataset's observations into x distinct nodes according to X
3. If X is numerical:
   - Split the dataset's observations into n distinct nodes according to computed threshold values on X
4. For each node N_x :
   - If each observation of the N_x refers to the same class:
     - N_x is a leaf with the resulted class.
   - otherwise
     - Repeat the algorithm for observations of the N_x.

The information gain is calculated from the entropy and conditional entropy (information), where entropy of the attribute is is a simple measure of disorder of data.

$$E(S) = -\sum_{j=1}^{n} P(s_j)log(P(s_j)) \tag{2}$$

$$I(S|A) = \sum_{i} \frac{|S_i|}{|S|} \times E(S_i) \tag{3}$$

$$G(S|A) = E(S) - I(S|A) \tag{4}$$

By this algorithm the DT is always attempting to fit the training data for 100 %, which brings the possible issue of over fitting [24]. This usually happens when DT grows so much that the number of its nodes is close to the size of the training data. In that case it has a lot of specific leafs conditions for every specific observation and only few leafs that are able of general classification decisions. This kind of the tree can have lower accuracy on the testing data.

There is an approach known as pruning [7] to make DT able of more general classifications and there are more available pruning algorithms [24]. In this case there was used reduced-error pruning [25] in post-pruning phase. This approach requires to divide the data set into training and testing set and pruning is performed on inducted tree by the validation on the testing data. The procedure evaluates each node, if it can be replaced by most frequent class without harmful impact on the tree accuracy. This operation is terminated when any further pruning will cause the decreasing of the tree accuracy.

2.3 Genetic Programming

GP is a set of evolutionary based algorithms inspired by biological evolution [18] and they are very popular due to their widely applications [2]. From defined set of individuals based on given grammar (Backus-Naur form) there is a synthesized solution to the defined problem. Genetic programming based on grammar is called Grammatical evolution [22].

The process comes through more steps. First the individual is represented by binary vector and using splitting is converted into codons - the vector of integers. This integers are substituted by the members of the given grammar. The mapping is performed by modulation of the codon's values by the number of the replacing rules of the nonterminal symbol. The mapping starts from the starting terminal of the grammar and ends until all the nonterminals are not replaced by the terminals. In case, that the mapping went more time through all the codons values and the individual is still not mapped into valid form, the process can be stopped and the individual is evaluated with the zero fitness value. The example of the valid individual with all substituted nonterminals in a tree structure is in Fig. 1.

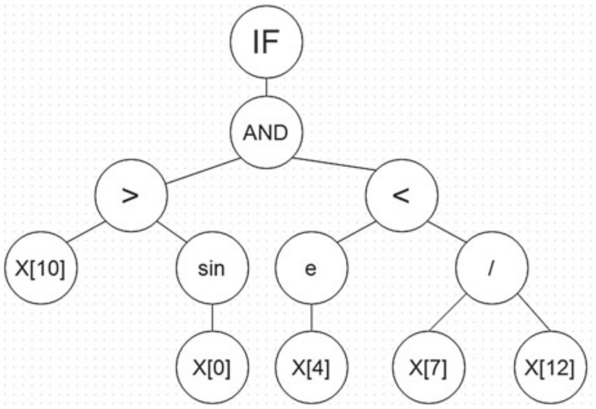


Fig. 1 The tree interpretation of the GP's individual

The next phase is the evaluation of all the valid individuals (Boolean forms) by adjusted fitness function. In this case it was computing of the percentage of the correct classified observations. On the subset of the best candidates there is performed the crossover [23] and mutation to create new generation of the population. The crossover is simply cutting the tree into two or three sub-trees (based on the number of points of crossover), mapping back to the codon values and replacing the genotype with the codon's parts of the other crossovered individual. The mutation is only modification of a single codon value, which has an impact to the resulted Boolean form. After defined number of iterations (generations), the best individual should represent the ideal solution.

The implementation of this algorithm was based on ECJ toolkit [20, 27]. Each individual was defined by the grammar that contains operators like $+$, $-$, $\times$, $\div$, $sin(x)$, $cos(x)$, e^x, etc. The behavior of some of the operations was modified. The absolute values serve as the inputs for the square root, as well as for the logarithm. In case of zero input it returns zero the same way as division by zero. All this restrictions was used to make the process of GP more stable.

Breeding of the individual was performed by MultiBreedingPipeline [20], where the options like List-Crossover, duplication and mutation of genes were simply adjusted. Details of the settings and results are described in next chapter.

3 Adjustments and Results

In this section there are described settings and results from the all of the steps of the experiment. First step was choosing the most suitable subset for each of the referenced variables (Thdi, Thdu, Plt, Pst, Freq) described in the appendix. As it was mentioned before, this task was performed by Correlation based filter selection driven

Table 1 Filtered subsets for each referenced attribute with their merit and sizes

Attribute	Criteria	Value	Subset
Thdu	Total m	0.41479	S_SI, THDu_max_1, THDi_max_1, freq, Urms_1_avg, Pst_1,
	Subset size	12	Uharm7_1, Uharm9_1, Uharm11_1, Uharm15_1, Uharm17_1,
	Subsets m	0.94246	Uharm25_1
Thdi	Total m	0.25476	U_Generator, f_SI, Q_Tracker, I_DC, S_F2, THDu_1,
	Subset size	20	THDu_max_1, THDi_max_1, Urms_1_avg, Q_1_avg, S_1_avg,
	Subsets m	0.4286	cos_1_avg, PF_1_avg, Uharm5_1, Uharm7_1, Uharm9_1,
			Uharm19_1, Uharm21_1, Uharm23_1
Pst	Total m	0.2209	f_SI, I_BAT, S_Tracker, Q_G1, S_G2, U_DC, Q_F1,
	Subset size	24	P_F3, Smer_Vetru, Relativni_Vlhkost, THDu_1,
	Subsets m	0.4453	THDu_max_1, cos_1_avg, PIt_1, Uharm7_1, Uharm11_1,
			Uharm13_1, Uharm15_1, Uharm17_1, Uharm19_1,
			Uharm21_1, Uharm23_1, Uharm25_1
Plt	Total m	0.23338	U_SI, f_SI, U_BAT, S_G2, M_GEN, I_F3, P_F3,
	Subset size	21	Atmosfericky_Tlak, Teplota_horniho_cidla, THDu_1,
	Subsets m	0.36156	THDi_max_1, freq, Urms_1_avg, Pst_1, Uharm7_1,
			Uharm9_1, Uharm15_1, Uharm19_1, Uharm21_1, Uharm23_1
Freq	Total m	0.19996	S_SI, I_Tracker, Q_Tracker, I_G1, RV_Meteo,
	Subset size	18	Irms_1_avg, Q_1_avg, S_1_avg, cos_1_avg, Pst_1,
	Subsets m	0.65152	Uharm3_1, Uharm7_1, Uharm9_1, Uharm11_1, Uharm13_1,
			Uharm15_1, Uharm19_1

by Genetic Algorithm. The amount of the subset's merit M_s was the fitness function of the GA. Every subset's selection obtained the same adjustment (number of generations: 1000, number of individuals: 500, crossover prob.: 0.6, mutation prob.: 0.01, crossover type: one-point). Resulted subsets are summarized in Table 1.

For each attribute, is evaluated the "Total m" which means the merit value of the entire given data set (118 attributes) to the referenced attribute. The subset size is the amount of chosen attributes for the referenced attribute due to the highest gained merit value and the "Subset m" holds that value.

In the next phase, the subsets were duplicated 7 times and modified by the replacement of the resulted class of the observation of n-ticks (0–6) back. This was proceeded to force the algorithms to create the forecast rules for the classification (1st. dataset for 0 tick forecast, 2nd. dataset for 1 tick forecast, 3rd. for 2 ticks forecast,...). This was possible to attend only with the confirmed evidence of auto-correlation in all of the variables for all of the subsets, which was confirmed due to the natural character of all of the attributes and of course by statistical test that was performed.

After the data preprocess phases, there comes the phase of the DT classification creation and measuring its accuracy. DT was obtained in the WEKA software as J48 classifier with adjusted reduced error pruning. The accuracy of all DT based classifiers is shown in Table 3. The average time to build a single DT was approximately 4 seconds on the standard notebook (2 thread CPU, 6GB RAM).

On the same datasets in the next phase was the synthesize the boolean formulas by the GP. For each of the dataset, there was created 5 populations with different adjustments for crossover, mutation, tournament size, etc. These multiple population computing was used in faith to obtain different but equally good results. Settings for all the populations are shown in Table 2. The accuracy of the best individuals from the five populations were averaged, so in the Table 3 there are approximated numbers to the results that were achieved, because the nature of the algorithm is stochastic and the random seed or mutations can not guarantee any of the results. The average time to run the GP for all of the 5 populations (they ran parallel) was approximately 12 min on the server station (30 thread CPU, 1TB RAM).

As an example there are two equally good synthesized rules for classification of THDu in time one tick ahead. The first is made by GP (listing 1.1) and the second is a product of C4.5 (Fig. 2). Both of them are easy to understand and apply.

Listing 1.1 GP synthesized classification condition for Thdu attribute

```
if (THDu_max_1 <
Urms_1_avg * (cos(THDu_max_1) + Urms_1_avg)) * (Uharm15_1 *
    sin(Uharm11_1))
   THDu[t+1] = "in";
else
   THDu[t+1] = "out";
```

As we can see, the GP is able to synthesize much shorter classification rule, due to the usage of more complex grammar. The differences were obtained in all of the classifiers. The table of average number of nodes of the classifies is presented in Fig. 3.

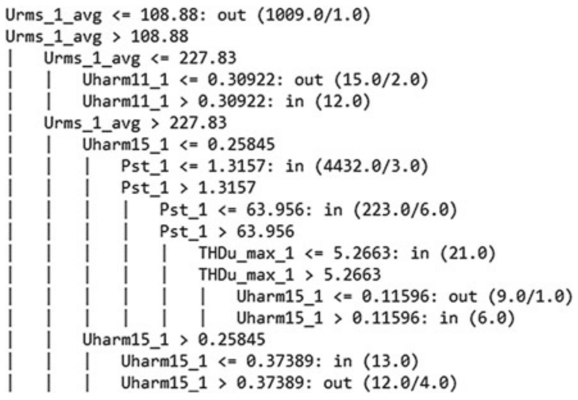
The quality of all of the obtained results was measured by their classification's accuracy, first on the training and second on the testing dataset. This numbers was compared to the results from C4.5 algorithm. The comparison of all of the classifiers is shown in the Table 3. The GP accuracy is the average of five population's best individual accuracy on a given dataset. GP means result of Grammatical evolution algorithm synthesis on training data and GP* is the same result but on testing data. Each line of the table represents the classification for $< 0, 6 >$ ticks ahead.

4 Conclusions

The outcomes of the data mining methods of this paper corresponds with the original idea of the authors, that the measured variables has a non-zero impact to each other. There is strongly evidence of dependencies between meteorological variables, power and power quality parameters. The results of this experiment become a corner stone for forecasting of PQP of electric energy as a part of Active-Demand Side Management for complex handling of energy distribution in Off-Grid systems.

Table 2 Adjustment of GE sub-populations

Settings	pop. 01	pop. 02	pop. 03	pop. 04	pop. 05
Candidates	1024	1024	1024	1024	1024
Generations	2500	2500	2500	2500	2500
Breed elite	7	10	1	25	50
List crossover prob.	0.1	0.6	0.6	0.5	0.9
Crossover type	one-point	one-point	two-point	two-point	one-point
Gene duplication prob.	0.25	0.35	0.05	0.2	0.05
Vector mutation prob.	0.65	0.05	0.35	0.3	0.05



```
Urms_1_avg <= 108.88: out (1009.0/1.0)
Urms_1_avg > 108.88
|   Urms_1_avg <= 227.83
|   |   Uharm11_1 <= 0.30922: out (15.0/2.0)
|   |   Uharm11_1 > 0.30922: in (12.0)
|   Urms_1_avg > 227.83
|   |   Uharm15_1 <= 0.25845
|   |   |   Pst_1 <= 1.3157: in (4432.0/3.0)
|   |   |   Pst_1 > 1.3157
|   |   |   |   Pst_1 <= 63.956: in (223.0/6.0)
|   |   |   |   Pst_1 > 63.956
|   |   |   |   |   THDu_max_1 <= 5.2663: in (21.0)
|   |   |   |   |   THDu_max_1 > 5.2663
|   |   |   |   |   |   Uharm15_1 <= 0.11596: out (9.0/1.0)
|   |   |   |   |   |   Uharm15_1 > 0.11596: in (6.0)
|   |   Uharm15_1 > 0.25845
|   |   |   Uharm15_1 <= 0.37389: in (13.0)
|   |   |   Uharm15_1 > 0.37389: out (12.0/4.0)
```

Fig. 2 Pruned tree for Thdu's one minute ahead classification

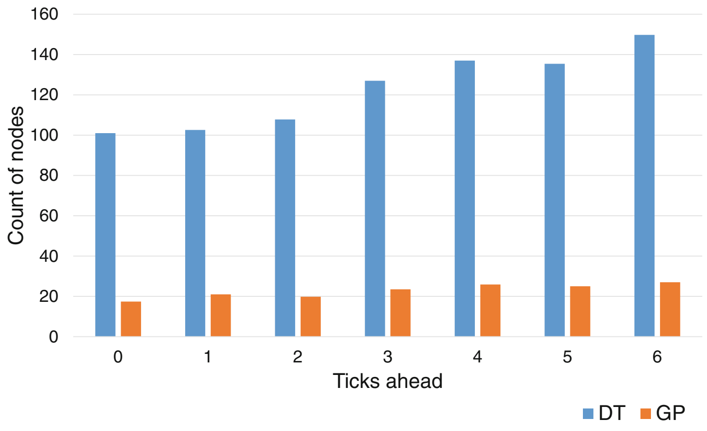


Fig. 3 Average numbers of nodes of classifiers on power quality attributes

Table 3 The tables of classification's accuracy

	Freq			THDi			THDu		
class. for	C4.5	GP	GP*	C4.5	GP	GP*	C4.5	GP	GP*
C_t	99.897 %	99.307 %	98.687 %	96.966 %	98.263 %	84.094 %	99.863 %	99.942 %	99.343 %
C_{t+1}	99.591 %	99.114 %	95.252 %	95.058 %	97.118 %	81.725 %	99.216 %	99.522 %	99.114 %
C_{t+2}	99.079 %	98.781 %	98.159 %	93.931 %	96.316 %	75.810 %	98.909 %	99.057 %	98.756 %
C_{t+3}	98.704 %	98.364 %	97.400 %	90.897 %	95.548 %	76.108 %	98.364 %	98.912 %	97.861 %
C_{t+4}	99.113 %	98.272 %	97.988 %	92.360 %	95.140 %	68.394 %	98.806 %	98.785 %	96.863 %
C_{t+5}	98.602 %	97.934 %	95.814 %	91.985 %	94.566 %	73.807 %	98.636 %	98.618 %	97.682 %
C_{t+6}	97.954 %	97.820 %	97.834 %	90.450 %	94.294 %	72.068 %	98.397 %	98.491 %	96.224 %
		Plt			Pst				
	class. for	C4.5	GP	GP*	C4.5	GP	GP*		
	C_t	98.091 %	88.767 %	84.376 %	95.023 %	92.491 %	82.245 %		
	C_{t+1}	98.330 %	89.360 %	83.140 %	96.149 %	92.838 %	84.461 %		
	C_{t+2}	98.364 %	90.592 %	83.369 %	94.784 %	92.912 %	82.126 %		
	C_{t+3}	98.193 %	89.675 %	77.685 %	95.363 %	93.092 %	84.103 %		
	C_{t+4}	98.090 %	89.605 %	69.928 %	95.259 %	92.996 %	83.933 %		
	C_{t+5}	97.681 %	89.276 %	75.725 %	95.123 %	93.289 %	78.878 %		
	C_{t+6}	98.158 %	89.083 %	80.310 %	95.157 %	92.636 %	83.771 %		

As it turns out, one of the weak spots can be considered not enough suitable data set for the demand of the experiment. The filtering partly solve this issue by obtaining the subsets with much greater merit values, than merit value of the entire data set. This turns out as an improvement, because the correlation between the higher merit value (Table 1) and the accuracy of the classification (Table 3) are evident.

Classifications made from the C4.5 algorithm with reduced error pruning dealt with accuracy more than 90 percent for most times of the experiment. The grater time gap between observation values and the desired classification caused the lower evidence of accuracy. The other issue of increasing time gap appears. The trees become bigger and bigger, so they were loosing generality and became more and more specific (over-fitting problem). Even with used pruning, their size became more than 250 nodes and in comparison to the GP results, the trees from DT were at least three times bigger. That could be the problem in case of application on bigger data set.

The genetic programming faced to the same issues as the C4.5 algorithm. On the lower count of the observations it is easier for GP individual to converge into non-general solution, which will appear as a problem on testing data.

For the comparison of the data-mining algorithms, it appears that both of them dealt with the almost equal problems. The GP's results were in this experiment more promising due to ability of use of the complex grammar in the Boolean forms and due to the rapidly growing of the DTs in many cases (3).

Acknowledgments This paper was conducted within the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070, project ENET CZ.1.05/2.1.00/03.0069, Students Grant Competition project reg. no. SP2015/142, SP2015/146, SP2015/170, SP2015/178, project LE13011 Creation of a PROGRES 3 Consortium Office to Support Cross-Border Cooperation (CZ.1.07/2.3.00/20.0075) and project TACR: TH01020426.

Appendix

`freq`	System Frequency (Hz)
`THDi_1`	Total Current Harmonic distortion in 1 min interval (%)
`THDi_max_1`	Maximal Total Current Harmonic distortion (%)
`THDu_1`	Voltage total harminic distortion in 1 min interval (%)
`Uharm11_1`	11th harmonic order of Voltage
`Uharm15_1`	15th harmonic order of Voltage
`PIt_1`	flicker long term severity (−)
`Pst_1`	flicker short term severity (−)
`THDu_max_1`	Maximal Voltage total harminic distortion in 1 min interval (%)
`Urms_1_avg`	Loads Voltage (V)

References

1. Ieee recommended practice for monitoring electric power quality. IEEE Std 1159–2009 (Revision of IEEE Std 1159–1995) pp. c1–81 (June 2009)
2. Allen, F., Karjalainen, R.: Using genetic algorithms to find technical trading rules1. J. Finan. Econ. **51**(2), 245–271 (1999)
3. 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)
4. Beasley, D., Bull, D.R., Martin, R.R.: An Overview of Genetic Algorithms: Part 1, Fundamentals (1993)
5. Bhargava, D.N., Sharma, G., Bhargava, R., Mathuria, M.: Decision tree analysis on j48 algorithm for data mining. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 3 (June 2013)
6. Duan, F., Zhao, Z., Zeng, X.: Application of decision tree based on c4.5 in analysis of coal logistics customer. In: Third International Symposium on Intelligent Information Technology Application, IITA 2009, vol. 2, pp. 380–383 (Nov 2009)
7. Frank, E.: Pruning decision trees and lists. Tech. rep. (2000)
8. Goksu, O., Teodorescu, R., Bak-Jensen, B., Iov, F., Kjr, P.: An iterative approach for symmetrical and asymmetrical short-circuit calculations with converter-based connected renewable energy sources. application to wind power. In: Power and Energy Society General Meeting, 2012 IEEE, pp. 1–8 (July 2012)
9. Gugale, P., Wang, J., Alt, B., Muller, H., Monti, A.: Development of network dimensioning guidelines for renewable island. In: Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2014 IEEE PES, pp. 1–6 (Oct 2014)
10. Hall, M., Holmes, G.: Benchmarking attribute selection techniques for discrete class data mining. IEEE Trans. Knowl. Data Eng. **15**(6), 1437–1447 (2003)
11. Hall, M.A.: Correlation-based feature selection for machine learning. Disertation thesis, Department of Computer Science, Hamilton, New Zealand (1999)
12. Hall, M.A.: Correlation-based feature selection for discrete and numeric class machine learning. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 359–366. ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2000)
13. Inman, R.H., Pedro, H.T., Coimbra, C.F.: Solar forecasting methods for renewable energy integration. Prog. Energy Combust. Sci. **39**(6), 535–576 (2013). http://www.sciencedirect.com/science/article/pii/S0360128513000294
14. Ji, N.: Steady-state signal generation compliant with iec61000-4-30: 2008. In: 22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013), pp. 1–4 (June 2013)
15. Kantardzic, M.: Data Mining: Concepts, Models, Methods and Algorithms. Wiley, New York (2002)
16. Kim, J.W., Lee, B.H., Shaw, M.J., Chang, H.L., Nelson, M.: Application of decision-tree induction techniques to personalized advertisements on internet storefronts. Int. J. Electron. Commer. **5**(3), 45–62 (2001)
17. Kosmak, J., Vramba, J., Uher, M., Stuchly, J., Kubalik, P.: Short-circuit protection in off-grid system. In: 14th International Conference on Environment and Electrical Engineering (EEEIC), pp. 339–344 (May 2014)
18. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
19. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, Norwell, MA, USA (1998)
20. Luke, S.: The ECJ Owner‘s Manual—A User Manual for The ECJ Evolutionary Computation Library (2014)
21. Matallanas, E., Castillo-Cagigal, M., Gutirrez, A., Monasterio-Huelin, F., Caamao-Martn, E., Masa, D., Jimnez-Leube, J.: Neural network controller for active demand-side management with PV energy in the residential sector. Appl. Energy **91**(1), 90–97 (2012). http://www.sciencedirect.com/science/article/pii/S0306261911005630

22. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language, vol. 4. Springer Science & Business Media (2003)
23. Poli, R., Langdon, W.B.: Genetic programming with one-point crossover and point mutation. Soft Computing in Engineering Design and Manufacturing, pp. 180–189. Springer-Verlag (1997)
24. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
25. Rinkal Patel, R.A.: A reduced error pruning technique for improving accuracy of decision tree learning. Int. J. Eng. Adv. Technol. 3 (June 2014)
26. Singh, M.P., Tiwari, R.: Article:correlation-based attribute selection using genetic algorithm. Int. J. Comput. Appl. **4**(8), 28–34 (August 2010) (Published By Foundation of Computer Science)
27. White, D.: Software review: the ECJ toolkit. Genet. Program. Evolvable Mach. **13**(1), 65–67 (2012)

Population Size Reduction in Particle Swarm Optimization Using Product Graphs

Iztok Fister Jr., Aleksandra Tepeh, Janez Brest and Iztok Fister

Abstract Purpose of this paper is to introduce a population size reduction in particle swarm optimization algorithm, where the reduction is performed by selecting two particles (also donor particles) randomly and replacing these by a new particle with elements determined from a set of pair values obtained by the Cartesian product of both donor particles for each particular element randomly. Average values of each pair values from the donor particles are calculated for corresponding elements of the new particle. The proposed PSOGP was applied on a benchmark function suite consisted of four well-known functions and compared with the original PSO algorithm. The results are very promising and show the potential of the proposed idea.

Keywords Product graphs · Optimization · Particle swarm optimization · Population size reduction

1 Introduction

In the past decades, nature-inspired algorithms gained immeasurable attention from scientific public. Nature-inspired algorithms are a very efficient tool for solving the different kinds of optimization tasks. In line with this, many various nature-inspired algorithms were proposed so far. Since the number of these algorithms became so huge, there is impossible to classify them all. Anyway, few years ago, one proposed taxonomy was developed in [5]. In this taxonomy, nature-inspired algorithms were

I. Fister Jr.(✉) · A. Tepeh · J. Brest · I. Fister
Faculty of Electrical Engineering and Computer Science,
University of Maribor, Smetanova Ul. 17, 2000 Maribor, Slovenia
e-mail: iztok.fister1@um.si

A. Tepeh
e-mail: aleksandra.tepeh@um.si

J. Brest
e-mail: janez.brest@um.si

I. Fister
e-mail: iztok.fister@um.si

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_7

divided into four groups based on their primary inspirations. These four groups can be summarized as follows:

– swarm intelligence (SI) based algorithms,
– bio-inspired algorithms (without SI-based algorithm),
– physics and chemistry based algorithms,
– other algorithms (e.g. inspiration of which have roots in sociology, history and some even in sport)

Currently, SI-based algorithms seem to be one of the more popular in research area. They were used in solving discrete and numerical optimization problems as well as practical applications, e.g., in [7, 11, 13, 17–19]. According to the NFL theorem [16], average performances of different algorithm families are the same when they are applied to all classes of problems. Therefore, a lot of special techniques were developed in order to improve the results by solving the different problems [4, 6, 14, 15].

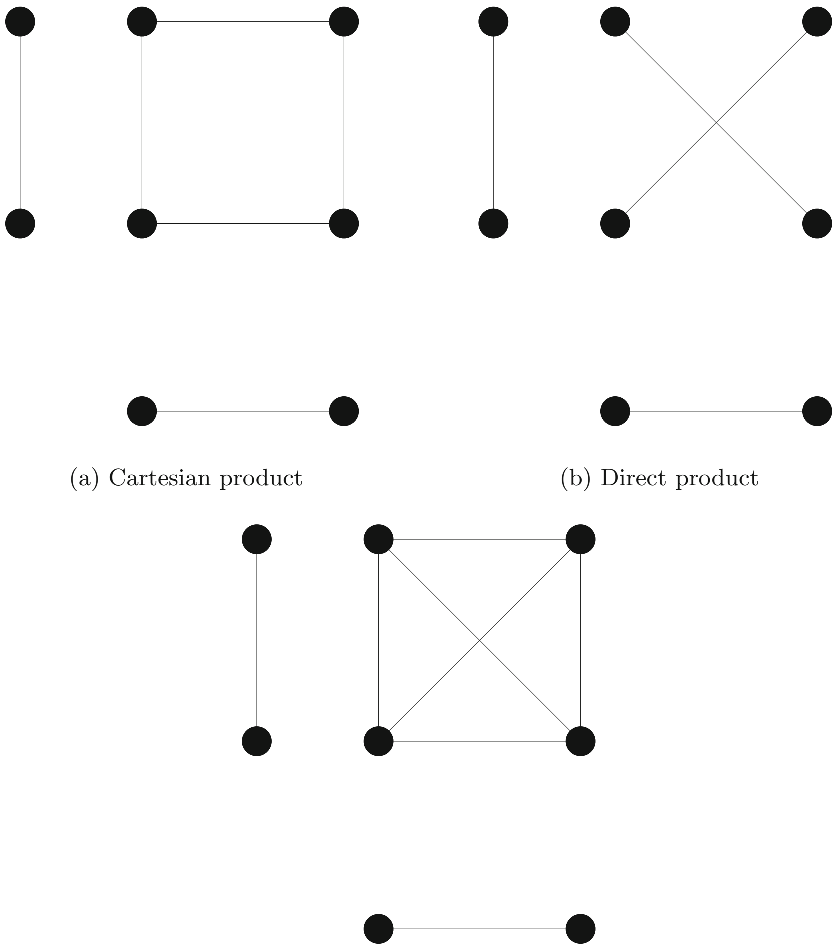
This paper proposes a population size reduction borrowed from Brest and Maučec [2], where authors justified an advantage of this mechanism as follows. A diversity of population is high at the start of optimization. The diversity becomes lower when a search process progresses and stepwise directs itself in promising regions of the search space. Thus, the search process can continue with a smaller number of population members because of wasting the number of fitness function evaluations. Typically, the population size is reduced proportionally to the number of generations.

Primarily, a candidate for elimination must be determined by the reduction. This paper proposes the PSOPG algorithm that selects two donor particles from the current population randomly for replacing them with the new particle. Here, these two donor particles are treated as sets of nodes $V(G)$ and $V(H)$ that are combined using the Cartesian product of these sets, $V(G) \times V(H)$. The new particle is calculated with elements as an average of pairs of elements (u_i, v_j), where the node v_j is selected randomly. The PSOPG algorithm was tested on a benchmark of four well-known functions from the literature. Interestingly, the PSOPG outperformed the results of the original PSO algorithm and showed the potential for the future work, where different type of graph products can be applied.

The remainder of the paper is structured as follows: in Sect. 2 a basic overview of graph products is given, and Sect. 3 presents a short overview of a swarm intelligence and particle swarm optimization. In Sect. 4 the new algorithm is proposed. Section 5 presents experiments and results, while the concluding section shows directions for possible future work.

2 Product Graphs

A graph product of two graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ is a new graph whose node set is the Cartesian product of sets $V(G) \times V(H)$ and where for any two nodes (g, h) and (g', h') in the product, the adjacency of those two nodes is determined entirely by the adjacency (or equality, or non-adjacency) of g and g', and that of h and h'. It turns out that there are 256 different types of graph products that can be defined. Among products that have been widely investigated and have many significant applications are the Cartesian product, the direct product, the strong product and the lexicographic product (the terminology is not quite standardized, so these products may actually be referred to by different names, however, we adopt the terminology from [8], where a comprehensive overview of graph products can be found).


(a) Cartesian product (b) Direct product

(c) Strong product

Fig. 1 Graph products

For the mentioned (so called standard) products, edges are defined as follows. In the *Cartesian product* $G \square H$ of graphs G and H two nodes (g, h) and (g', h') are adjacent when ($gg' \in E(G)$ and $h = h'$) or ($g = g'$ and $hh' \in E(H)$). In the *direct product* $G \times H$ nodes (g, h) and (g', h') are adjacent when $gg' \in E(G)$ and $hh' \in E(H)$. In the *strong product* $G \boxtimes H$ nodes (g, h) and (g', h') are adjacent whenever ($gg' \in E(G)$ and $h = h'$) or ($g = g'$ and $hh' \in E(H)$) or ($gg' \in E(G)$ and $hh' \in E(H)$). Note that $E(G \boxtimes H) = E(G \square H) \cup E(G \times H)$ and that the notation of these three products comes from multiplying two copies of K_2 (complete graph on two nodes), see Fig. 1. Finally, in the *lexicographic product* $G[H]$ nodes (g, h) and (g', h') are adjacent if either $gg' \in E(G)$ or ($g = g'$ and $hh' \in E(H)$).

It follows immediately from the definitions that the Cartesian, the direct and the strong product are commutative in the sense that the graph $G * H$ is isomorphic to $H * G$, where $*$ stands for any one of the three mentioned products. Note that the lexicographic product is not commutative (we want to emphasize this fact with the notation $G[H]$, although in the literature the notation $G \circ H$ is often used for this product). However, all four standard products are associative, i.e. for given graphs G_1, G_2, G_3 the graphs $(G_1 * G_2) * G_3$ and $G_1 * (G_2 * G_3)$ are isomorphic (here $*$ stands for any one of the four standard products). Associativity of all four products allows the easy extension of these products to arbitrarily many factors.

3 The Particle Swarm Optimization

The particle swarm optimization (PSO) is one of the older members of SI-based algorithm family. It was proposed by Kennedy and Eberhart [12] in 1995. The PSO algorithm mimics behavior of bird flocks. Therefore, it is a population-based algorithm, where the population consists of Np particles. Each particle represents a solution of the problem to be solved. Typically, the solution is represented as a vector $\mathbf{x}_i = \{x_{i,j}\}$ for $j = 1, \ldots, n$ with real-valued elements $x_{i,j} \in \mathbf{R}$, where n determines a dimensionality of the problem.

The pseudo-code of the original PSO algorithm is illustrated in Algorithm 1 that consists of the following components:

– initialization of population randomly (function init_particles),
– fitness function evaluation (function evaluate_the_new_solution),
– selection of the local best solution (lines 8–10),
– selection of the global best solution (lines 11–13),
– generation of the new solution (function generate_new_solution).

Additionally, the termination condition (function termination_condition_not_meet) needs to be defined in order to complete operating the PSO algorithm. Typically, the maximum number of fitness function evaluations MAX_FE is used as a termination condition. Interestingly, the PSO algorithm works with two sets of particles because it manages the local best solutions $\mathbf{p}_i^{(t)}$ for each particle $\mathbf{x}_i^{(t)}$. Moreover, the best solution in the population $\mathbf{g}^{(t)}$ is determined in each generation. The new particle position is generated as follows:

$$\begin{aligned}\mathbf{v}_i^{(t+1)} &= \mathbf{v}_i^{(t)} + C_1U(0,1)(\mathbf{p}_i^{(t)} - \mathbf{x}_i^{(t)}) + C_2U(0,1)(\mathbf{g}^{(t)} - \mathbf{x}_i^{(t)}),\\ \mathbf{x}_i^{(t+1)} &= \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)},\end{aligned} \tag{1}$$

where $U(0, 1)$ denotes a random value drawn from the uniform distribution, and C_1 and C_2 are learning factors.

4 The Proposed PSOPG Algorithm

The proposed algorithm PSOPG implements a population size reduction feature, where the population size Np is stepwise reduced by increasing the generation number t. This reduction is controlled by an additional parameter so-called *reductionRate* that determines when the reduction needs to be performed. The pseudo-code as presented in Algorithm 2 is added to a pseudo-code of Algorithm 1 after line 16 in order to implement this feature.

Algorithm 1 Tho original PSO algorithm

```
Input: PSO population of particles $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n})^T$ for $i = 1, \ldots, Np$, $MAX\_FE$.
Output: The best solution $\mathbf{g}$ and its corresponding value $f_{min} = \min(f(\mathbf{x}))$.
 1: init_particles;
 2: $t = 0$;   // generation counter
 3: $eval = 0$;   // counter of the fitness function evaluations
 4: while termination_condition_not_meet do
 5:    for $i = 1$ to $Np$ do
 6:       $f_i$ = evaluate_the_new_solution($\mathbf{x}_i^{(t)}$);
 7:       $eval = eval + 1$;
 8:       if $f_i \le pBest_i$ then
 9:          $\mathbf{p}_i^{(t)} = \mathbf{x}_i^{(t)}$; $pBest_i^{(t)} = f_i$;  // save the local best solution
10:       end if
11:       if $f_i \le f_{min}$ then
12:          $\mathbf{g}^{(t)} = \mathbf{x}_i^{(t)}$; $f_{min} = f_i$;  // save the global best solution
13:       end if
14:       $\mathbf{x}_i^{(t)}$ = generate_new_solution($\mathbf{x}_i^{(t)}$);
15:    end for
16:    $t = t + 1$;
17: end while
```

Algorithm 2 Proposed algorithm PSOPG

```
1: if (($t$ mod $reductionRate$) == 0) then
2:    $makeReduction$;   // performing the population size reduction
3: end if
```

In fact, the additional code launches the process of population size reduction via the *makeReduction* function call. However, the question is which particle to eliminate from the population by this reduction. In this paper, an idea from product graphs [1] is implemented.

In the proposed PSOPG algorithm, each particle in the population is treated as an empty graph G (recall that an empty graph on n nodes consists of n isolated nodes and contains no edges), where elements of a particle are represented by a set of nodes $V(G)$. Two randomly selected donor particles, i.e. empty graphs G and H, enter in the population reduction process to generate the new particle with characteristics of the both donors. Two elements of donor particles are identified by the pair (u_i, v_j) for the i-th element of the new particle (also trial particle), where the element v_j is selected randomly. The value of this element is obtained by calculating the average value of elements u_i and v_j.

We remark that to describe our idea, the Cartesian product of sets would be enough. However, we used the notion of the Cartesian product of graphs since we believe that our idea could be improved using graphs. More precisely, we believe that representation of particles as paths (a path P_k is a graph with $V(P_k) = \{u_1, \ldots, u_k\}$ and $E(P_k) = \{u_i u_{i+1} | i \in \{1, \ldots, k-1\}\}$), or even other simple graphs, and usage of different graph products would be worth to study (Fig. 2).

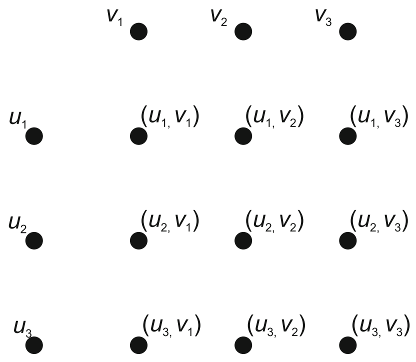


Fig. 2 The Cartesian product $G \square H$ of empty graphs G and H with $V(G) = \{u_1, u_2, u_3\}$ and $V(H) = \{v_1, v_2, v_3\}$. This product consists of isolated nodes, represented by the ordered pairs (u_i, v_j). In our case, the average value of corresponding nodes u_i and v_j in factors is calculated, that represents a potential value for the i-th element of the new particle

Motivation behind a selection of the candidate for elimination by the reduction is to preserve the particle in the next generation that transfers characteristics of both donors whereby both must be eliminated from the population. The selection operation acts as follows. The i-th element of the trial particle is selected randomly from the set $\{(u_i, v_j) | j \in \{1, \ldots, n\}\}$. As a result, the value for the i-th element of the trial particle is calculated as an average between a value of node u_i and v_j in the corresponding donor particles. In other words, this operation is expressed mathematically as

$$x_i^{(t+1)} = \frac{x_{u_i}^{(t)} + x_{v_j}^{(t)}}{2}. \tag{2}$$

As can be seen from Eq. 2, the trial particle becomes the new one i-th particle for the next generation. Thus, the j-th particle is eliminated by squeezing the current population according to the following equation

$$\mathbf{x}_k^{(t+1)} = \mathbf{x}_{k+1}^{(t)}, \quad \text{for } k = 1, \ldots, Np^{(t)} - 1. \tag{3}$$

Finally, the population size is decremented after this operation.

5 Experiments

The goal of our experimental work was to show that the population size reduction in the PSOPG algorithm using the product graphs outperforms the results of the original PSO by optimizing the benchmark test suite consisted of four well-known functions from literature. All algorithms were implemented in Python programming language, while the Cartesian products were obtained using NetworkX library. During the tests, the parameters of both PSO algorithms were set as follows: $C_1 = C_2 = 2.0$, $MAX\_FE = 50,000$ and the starting population size of the PSOPG $Np^{(0)} = 100$. The 25 independent runs were conducted, and the accumulated results were compared according to their best, median, worst and mean values, as well as standard deviation. In the remainder of the paper, the benchmark test suite is illustrated, the results of experiments are presented and discussed.

5.1 Benchmark Function Suite

Benchmark functions were downloaded from Neal Holtschulte website [10] and they are presented in Table 1.

Table 1 Summary of the benchmark functions

Tag	Function	Definition	Domain
F_1	Ackley's	$f(\mathbf{x}) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	$[-32, 32]$
F_2	Griewank	$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	$[-512, 512]$
F_3	Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$	$[-5.12, 5.12]$
F_4	Sphere	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	$[-5.12, 5.12]$

The purpose of this preliminary research was to show that an idea of using the product graphs by reducing the population size could be successfully applied to the original PSO algorithm. Therefore, the original benchmark functions without shifting and rotating were considered in our tests. Note that functions of dimension $n = 10$ were employed during the tests.

5.2 Results

Four variants of the PSOPG algorithm were taken into consideration in order to show how the parameter *reductionRate* influences the results of the optimization. Thus, this parameter was varied as $reductionRate = \{80, 60, 40, 20\}$ that corresponds to a reduction of the population sizes for $\Delta_{Np} = \{6, 8, 13, 29\}$ when the initial population size of $Np^{(0)} = 100$ is considered. Additionally, two different population sizes were used for the original PSO, i.e., $Np = \{100, 50\}$.

The results of the mentioned tests are presented in Table 2, where the characteristic measures are displayed for each variant of the PSO and PSOPG algorithms for each specific function. Here, the particular variant of the algorithm is denoted either by the population size of the PSO or the parameter *reductionRate* of PSOPG that are enclosed by parentheses following the algorithm names. The best results according to mean values are represented bold in the table.

As can be seen from Table 2, the best results according to the mean values are obtained by the PSOPG using the smaller values of *reductionRate* (i.e., $reductionRate = \{20, 40\}$). Interestingly, the original PSO with population size $Np = 100$ outperformed the results of the PSO with smaller population size $Np = 50$ by the same number of the fitness function evaluations $MAX\_FE = 50,000$.

5.3 Discussion

Two facts can be concluded from the results of our experimental work. Firstly, the population size reduction using the product graphs in the PSOPG algorithm outperformed the results of the original PSO. Secondly, the parameter *reductionRate* has the important influence on the results of the PSOPG. The lower this value is, the better the results. However, in our tests, the allowed maximum reducing the population size was $\Delta_{Np} = 29$ for the starting population size $Np_0 = 100$. This means that the original population size can be maximally reduced for near 30 % that is still far from the reasonable minimum of 90 %. Therefore, more tests should be performed in the future in order to determine the correct behavior of this parameter.

It seems the used product graphs is similar to arithmetic crossover in evolutionary algorithms [3] or in PSO [9]. The purpose of our study was not to outperform

Table 2 Comparison of the experimental results

		F_1	F_2	F_3	F_4
PSO (100)	Best	3.21	0.29	1.90E-06	0.00033
	Median	7.38	1.10	1.39	0.46
	Worst	10.52	2.49	78.18	3.18
	Mean	7.51	1.10	16.28	0.65
	Std	1.67	0.53	23.06	0.76
PSO (50)	Best	4.50	0.23	9.58E-07	0.0063
	Median	7.24	0.88	2.33	0.27
	Worst	12.10	3.28	53.78	4.52
	Mean	7.40	1.02	11.62	0.70
	Std	1.89	0.59	17.04	0.97
PSOGP (80)	Best	1.92E-05	5.38E-07	3.37E-14	3.61E-18
	Median	3.17	0.53	5.68E-10	2.70E-10
	Worst	7.08	0.96	77.65	1.75E-06
	Mean	2.68	0.51	9.97	1.53E-07
	Std	2.04	0.29	22.18	3.92E-07
PSOGP (60)	Best	1.59E-06	1.72E-11	0.0	7.76E-18
	Median	3.57	0.44	3.22E-10	1.43E-12
	Worst	5.62	0.93	55.44	1.65E-06
	Mean	3.38	0.42	15.95	6.72E-08
	Std	1.55	0.31	21.38	3.24E-07
PSOGP (40)	Best	9.19E-07	3.77E-15	0.0	3.72E-25
	Median	2.80	0.54	1.77E-15	1.50E-17
	Worst	7.87	0.83	3.85	2.55E-11
	Mean	2.54	0.47	**3.85**	1.74E-12
	Std	2.16	0.25	13.07	5.99E-12
PSOGP (20)	Best	5.01E-14	0.0	0.0	7.69E-44
	Median	1.63E-07	0.35	0.0	7.68E-33
	Worst	7.10	0.83	58.07	3.14E-24
	Mean	**1.07**	**0.37**	11.14	**1.31E-25**
	Std	1.78	0.22	19.35	6.14E-25

operating this kind of crossover, but to show that the neighbourhood of candidate solution can be defined as product graphs. However, a potential of this idea needs to be fully elaborated in the future work.

6 Conclusion

In this paper we presented a new variant of the PSO algorithm. i.e., the PSOGP that proposes the use of graph products in order to help to reduce the population size. Experiments were conducted on a benchmark functions and results show the potential of developed idea. However, this study leaves a lot of free space for further improvements. For instance, the other graph products like strong, lexicographic, direct could be applied in place of Cartesian product. Furthermore, in place of calculating the average values, the various operators between nodes should be also selected. Finally, there are also possibilities to test this idea in other nature-inspired algorithms.

Acknowledgments The second author is supported by ARRS Research Program P1-00383.

References

1. Bondy, J.-A., Murty, U.S.R.: Graph Theory, Graduate Texts in Mathematics. Springer, New York (2008)
2. Brest, J., Maučec, M.S.: Population size reduction for the differential evolution algorithm. Appl. Intell. **29**(3), 228–247 (2008)
3. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer Science & Business Media (2003)
4. Fister, I., Strnad, D., Yang, X.-S., Fister I.: Adaptation and hybridization in nature-inspired algorithms. In: Adaptation and Hybridization in Computational Intelligence, pp. 3–50. Springer (2015)
5. Fister I., Yang, X.-S., Fister, I., Brest, J., Fister, D.: A brief review of nature-inspired algorithms for optimization. Elektrotehniški vestnik **80**(3), 116–122 (2013)
6. Fister, I., Yang, X.-S., Ljubič, K., Fister, D., Brest, J., Fister, I.: Towards the novel reasoning among particles in PSO by the use of rdf and sparql. The Scientific World Journal **2014** (2014)
7. Gálvez, A., Iglesias, A.: Efficient particle swarm optimization approach for data fitting with free knot b-splines. Comput. Aided Des. **43**(12), 1683–1692 (2011)
8. Hammack, R.H., Imrich, W., Klavžar, S.: Handbook of Product Graphs. Discrete mathematics and its applications. CRC Press, Boca Raton (2011)
9. Hao, Z.-F., Wang, Z.-G., Huang, H.: A particle swarm optimization algorithm with crossover operator. In: 2007 International Conference on Machine Learning and Cybernetics, vol. 2, pp. 1036–1040. IEEE (2007)
10. Holtschulte, N., Moses, M.: Should every man be an island? (website). (2013)
11. Kaiwartya, O., Kumar, S., Lobiyal, D.K., Tiwari, P.K., Abdullah, A.H., Hassan, A.N.: Multi-objective dynamic vehicle routing problem and time seed based solution using particle swarm optimization. J. Sens. **2015** (2015)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings., IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
13. Olusanya, M.O., Arasomwan, M.A., Adewumi, A.O.: Particle swarm optimization algorithm for optimizing assignment of blood in blood banking system. Comput. Math. Methods Med. (2014)
14. Pluhacek, M., Senkerik, R., Zelinka, I., Davendra, D.: Gathering algorithm: A new concept of PSO based metaheuristic with dimensional mutation. In: 2014 IEEE Symposium on Swarm Intelligence (SIS), pp. 1–6. IEEE (2014)
15. Tvrdík, J., Poláková, R., Veselský, J., Bujok, P.: Adaptive variants of differential evolution: Towards control-parameter-free optimizers. In: Handbook of Optimization, pp. 423–449. Springer (2013)
16. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
17. Ye, Z., Wang, M., Hu, Z., Liu, W.: An adaptive image enhancement technique by combining cuckoo search and particle swarm optimization algorithm. Comput. Intell. Neurosci. **2015** (2015)

18. Zhang, Y., Wu, L.: Crop classification by forward neural network with adaptive chaotic particle swarm optimization. Sensors **11**(5), 4721–4743 (2011)
19. Zhang, Y., Wu, L., Dong, Z., Wang, S., Zhou, Z.: Face orientation estimation by particle swarm optimization. In: 2009 Second International Symposium on Information Science and Engineering (ISISE), pp. 388–391. IEEE (2009)

Cooperation of Evolutionary Algorithms: A Comparison of Several Hierarchical Models

Radka Poláková and Josef Tvrdík

Abstract The cooperation of evolutionary algorithms in a simple hierarchical model was proposed. Three adaptive variants of differential evolution and covariance-matrix-adaptation evolutionary strategy were chosen for cooperation. The performance of the model was tested on the learning-based CEC 2015 suite of 15 functions at levels of the dimension 10 and 30. The results showed that the proposed cooperation is beneficial. The proposed model was superior significantly compared to any of individual algorithms included in the cooperation. The cooperation of the selected four algorithms appeared beneficial especially in the problems of dimension $D = 30$, where the cooperative model outperformed all the algorithms including other cooperative models in comparison.

Keywords Evolutionary algorithms · Cooperation · Hierarchical model · Experimental comparison

1 Introduction

Various evolutionary algorithms differ in the performance on different optimization problems substantially. The search strategy used in an algorithm is suitable for some problems, whilst it fails on the other problems. Cases of bad convergence or stagnation of some algorithms are described in the literature [9]. This is reason for considering the concept of some cooperative models combining different search strategies. Such models have appeared in the design of parallel evolutionary algorithms, e.g. [3, 16]. Simple models of cooperation are also described in [4, 28].

We proposed a simple model of hierarchical cooperation of four evolutionary algorithms and applied it to CEC 2015 competition of algorithms for the boundary

R. Poláková (✉) · J. Tvrdík
Centre of Excellence IT4Innovations, Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, 30. Dubna 22, 701 03 Ostrava, Czech Republic
e-mail: radka.polakova@osu.cz

J. Tvrdík
e-mail: josef.tvrdik@osu.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_8

constrained problems [13]. The test suite of 15 functions at several levels of dimension was defined specially for this competition as well as the conditions for the run of algorithms and the way of presentation of experimental results [10].

The algorithms in proposed model [13] were selected more or less intuitively. The aim of this paper is to study this model in more detail, especially to get an insight into the role of the algorithms involved in cooperation. The rest of the paper is organized as follows. The proposed hierarchical model of cooperation is described in Sect. 2 and the algorithms selected for the cooperation in Sect. 3. Section 4 presents the setting of experiments and the results are shown in Sect. 5. The paper is concluded in Sect. 6.

2 Model of Cooperation

A simple hierarchical model for the cooperation of the selected algorithms was inspired by the topology of the parallel model used in [3]. A similar model has also appeared in Elsayed et al. [4], where *CMA-ES*, a real-value coded genetic algorithm, and an adaptive DE cooperate in the search.

The model uses a star topology with several islands and one mainland depicted in Fig. 1. The algorithms on islands work independently and they can work in parallel mode. Pseudo-parallel regime was used in our Matlab implementation. A very simple migration way was chosen: when the algorithms on islands finished the search, the populations are prepared to migrate. After stopping all the island processes, the current island populations are put together and the joint population containing all the individuals found on the islands is moved to the mainland, where the search continues until the stopping condition for the mainland is reached. Then the whole process is finished, i.e. there is no migration from the mainland to the islands. It is also possible to propose a bit more complicated cooperative model, in which the process continues by redistribution of the final mainland population into the islands and the search repeats cyclically with renewed populations containing the individuals evolved on the mainland.

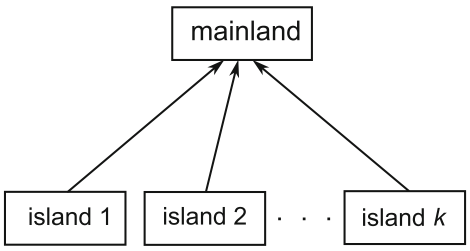


Fig. 1 Topology of cooperative model

In the versions of cooperative algorithms compared in this paper, the total number of function evaluations maximally allowed (*MaxFES*) is distributed uniformly among all $k + 1$ nodes of the cooperative model, k is the number of islands.

Another question which arises when we consider the hierarchical cooperative model, is the selection of algorithms for cooperation. We should select efficient algorithms using different search strategies. Thus, we choose algorithms which succeeded in CEC 2013 or CEC 2014 competitions. According these rather vague criteria, we choose the following four algorithms. One of them is the well-known covariance matrix adaptation evolutionary strategy, the remaining three algorithms are different adaptive DE versions:

– Covariance Matrix Adaptation Evolutionary Strategy (*CMA-ES*) [8], which was the base of all the algorithms with the best ranking in CEC competitions. Its aim in the cooperative model is to provide a rotation-invariant search substantially distant from any DE variant.
– *SHADE* [20] as the best DE version in CEC 2013 competition. It uses an archive of former solutions and the *current-to-pbest* mutation completely different from mutation commonly used in DE.
– *b6e6rl* variant of competition-adaptive DE [24] as base of the algorithms that achieved the second [14] and the third [2] best ranking among DE versions in CEC 2014 competition. The expected advantage of this DE variant in the cooperation is the exploitation of various DE strategies including the exponential crossover.
– *L-SHADE* [21], the best DE variant in CEC 2014. It was selected despite using search strategy like *SHADE* but it decreases the population size dynamically, which is suitable for its use as the mainland algorithm because it can cope with initial population of large size effectively. Thus, all the individuals evolved on the islands are used in the final search on the mainland.

3 Description of Algorithm Selected to Cooperation

The DE algorithm [17, 18] works with a population of individuals (*NP* points in the search domain) that are considered as candidates of solution. A parameter *NP* is called the size of the population. The population is developed by using evolutionary operators of selection, mutation, and crossover generation by generation. Applications of evolutionary operators in the old generation *P* create individuals for a new generation *Q*. After completing the new generation *Q*, the generation *Q* becomes the old generation *P* for next iteration. A new trial point is generated by using mutation (with control parameter *F*) and crossover (with control parameter *CR*).

3.1 CDE Algorithm, Version B6e6rl

Competitive adaptation of the DE strategies and the control parameters was originally proposed in [22]. There are *H* different DE strategies or settings of control parameters (*F*, *CR*) in a pool and we choose randomly among them with probabilities proportional to their success rate in preceding steps. This adaptive variant of DE is called competitive differential evolution (CDE) hereafter.

Several variants of CDE were compared in different benchmark tests [23, 24]. The CDE variant denoted *b6e6rl* was found as one of the most efficient among all the tested CDE variants. The variant uses twelve different DE strategies or parameter settings. This *b6e6rl* version appeared the most reliable and the second fastest algorithm in the comparison with the state-of-the-art adaptive DE algorithms [1, 12, 15, 26, 27] on the benchmark set of six shifted functions at three levels of dimension ($D = 10$, 30, and 100) [25].

3.2 SHADE

Success-History Based Parameter Adaptation for Differential Evolution (*SHADE*) algorithm [19, 20] was introduced by Tanabe and Fukunaga in 2013. This very efficient optimization method was derived from the Adaptive Differential Evolution with Optional External Archive (*JADE*) [27] proposed by Zhang and Sanderson. The main extension of *SHADE* compared to original *JADE* is in a history-based adaptation of the control parameters F and CR. Both *JADE* and *SHADE* variants use an efficient greedy *current-to-pbest* mutation strategy, archive A and the adaptation of parameters F and CR. The new mutant vector $\boldsymbol{u}$ is generated from four mutually different individuals – the current individual $\boldsymbol{x}_i$, $\boldsymbol{x}_{pbest}$ (which is randomly chosen individual from the $p \times 100\,\%$ best points of P), randomly selected point $\boldsymbol{x}_{r1}$ from P and randomly selected point $\boldsymbol{x}_{r2}$ from $P \cup A$ (A is archive), as it is formed in the equation:

$$\boldsymbol{u} = \boldsymbol{x}_i + F \cdot (\boldsymbol{x}_{pbest} - \boldsymbol{x}_i) + F \cdot (\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}). \tag{1}$$

Unlike *JADE*, the parameter of p (separately for each $\boldsymbol{x}_i$) for selecting the $\boldsymbol{x}_{pbest}$ is uniformly distributed random number from interval $p \in [2/NP,\ 0.2]$, then the point $\boldsymbol{x}_{pbest}$ is selected from the $p \times 100\,\%$ best points of P. After mutation, the binomial crossover operation is used for generating the trial point.

If the function value of the newly composed trial point $\boldsymbol{y}$ is better than the function value in $\boldsymbol{x}_i$, the trial point $\boldsymbol{y}$ replaces the $\boldsymbol{x}_i$ in population and the point $\boldsymbol{x}_i$ is inserted into archive A. At the beginning of the algorithm, the archive A is set to $A = \emptyset$. When the size of the archive A exceeds the size of the population NP, superfluous randomly chosen points are deleted from the archive A. The purpose of the archive is to store old good solutions from previous generations and use them for the mutation according to (1).

The successful values of the control parameters F and CR are stored into auxiliary memories S_F and S_C. At the start of computing of new generation, S_F and S_C equal to $\emptyset$. The historical circle memories M_F and M_C used for generating new values of the control parameters F and CR are updated using the successful values stored in auxiliary memories S_F and S_C after each generation. The size of these two circle memories is set to $|M_F| = |M_C| = H$. At the beginning of the algorithm, all values in memories M_F and M_C are set to 0.5. The new values of M_{F_ℓ} and M_{C_ℓ} are computed as weighted Lehmer mean of the current values from S_F and weighted arithmetic mean of the current values from S_C according to (2), respectively. The means (3) are weighted by the difference between function value in $\boldsymbol{y}_m$ and $\boldsymbol{x}_m$, see Eq. (4). The values of M_{F_ℓ}, M_{C_ℓ}, and ℓ remain the same if no successful point was created in the last generation. At the beginning $\ell = 1$, and ℓ is increased by 1 after each writing into M_F and M_C memories. If $\ell > H$, then ℓ is again set to $\ell = 1$.

$$M_{F_\ell} = mean_{WL}(S_F) \quad if\ S_F \neq \emptyset, \quad M_{C_\ell} = mean_{WA}(S_C) \quad if\ S_C \neq \emptyset, \tag{2}$$

$$mean_{WL}(S_F) = \frac{\sum_{m=1}^{|S_F|} w_m\, F_m^2}{\sum_{m=1}^{|S_F|} w_m\, F_m}, \quad mean_{WA}(S_C) = \sum_{m=1}^{|S_C|} w_m\, CR_m, \tag{3}$$

$$w_m = \frac{\Delta f_m}{\sum_{h=1}^{|S_C|} \Delta f_h}, \quad \Delta f_m = |f(\boldsymbol{x}_m) - f(\boldsymbol{y}_m)|. \tag{4}$$

The parameters F and CR are generated before each computing of a new trial point $\boldsymbol{y}$ as follows: for index r randomly chosen, $1 \leq r \leq H$, F is generated from Cauchy distribution with parameters M_{F_r} and 0.1 and CR is generated from normal distribution with parameters M_{C_r} and 0.1.

3.3 L-SHADE

L-SHADE [21] was derived from *SHADE* [19, 20] by Tanabe and Fukunaga in 2014. *L-SHADE* differs from *SHADE* above all in dynamic reduction of the population size. The population size is decreasing linearly generation by generation with the increasing number of the objective function evaluations (*FES*) during the search process from the initial value NP^{init} to the final value NP^{min} at the end of the search process, i.e. if allowed number of the function evaluations (*MaxFES*) is reached:

$$NP_{G+1} = round\left[\left(\frac{NP^{min} - NP^{init}}{MaxFES}\right) FES + NP^{init}\right], \tag{5}$$

where *FES* is the current number of the objective function evaluations. Whenever $NP_{G+1} < NP_G$, the $(NP_G - NP_{G+1})$ worst individuals (i.e. the individuals with the highest function values) are deleted from the population.

Unlike *SHADE*, the constant value p for the *current-to-pbest* mutation is used in *L-SHADE*. The size of archive of $2.6 \times NP$ is recommended by the authors. *L-SHADE* with this parameter setting was the best version of DE in CEC2014 competition [11].

3.4 CMA-ES

The evolutionary strategy (*ES*) with covariance matrix adaptation was proposed by Hansen a Ostermeier in [8]. From each current vector (point), λ new points can be generated according to

$$\boldsymbol{x}^{N_\ell} = \boldsymbol{x} + \delta \boldsymbol{B} \boldsymbol{z}_\ell \,, \tag{6}$$

where $\boldsymbol{x} = (x_1, \ldots, x_D)^T \in R^D$ is the current vector to be optimized, $\boldsymbol{z} = (z_1, \ldots, z_D)^T \sim N(\boldsymbol{0}, \boldsymbol{I})$, its elements $z_i \sim N(0, 1)$ are independent, $\boldsymbol{z}_\ell$, $\ell = 1, \ldots, \lambda$ are independent realizations of $\boldsymbol{z}$, and $\delta > 0$ is a parameter determining the step size. Covariance matrix $\boldsymbol{C}$ of the mutation distribution determines $\boldsymbol{B}$, so that $\boldsymbol{B}\boldsymbol{z} \sim N(\boldsymbol{0}, \boldsymbol{C})$, $\boldsymbol{C} = \boldsymbol{B}\boldsymbol{B}^T$ holds, $\boldsymbol{C}^{start} = \boldsymbol{I}$. Matrix $\boldsymbol{C}$ is adapted during the search process.

The source code of *CMA-ES* in Matlab used in our algorithms was downloaded from [7]. The *purecmaes.m* function was then modified for the cooperative algorithms.

4 Experiments

The aim of the paper is to investigate the role of individual algorithms cooperating in the proposed hierarchical model. Therefore, the cooperative algorithm with four optimizer mentioned above (denoted *HiCo* hereafter) was compared with each component running alone (abbreviated by *CMAES*, *SHADE*, *b6e6rl*, and *LSHADE* hereafter) and with hierarchical cooperative algorithms, in which only two islands are used. *LSHADE* is used as the mainland algorithm without any change. It results in the comparison with algorithms labeled *HiCoxx*, where *xx* stands for the first letter of the algorithms on the islands. The list of *HiCoxx* algorithms follows:

– *HiCoSb* – two islands, where *SHADE* and *b6e6rl* are used, *CMAES* is omitted. Only three different DE variants are used in this model.
– *HiCoCb* – two islands, where *CMAES* and *b6e6rl* are used, *SHADE* is omitted. Completely different strategies of the search are used on the islands.
– *HiCoCS* – two islands, where *CMAES* and *SHADE* are used, *b6e6rl* is omitted. Completely different strategies of the search are used on the islands in this model, too.

The comparison of results achieved by *HiCo* and *CMAES*, *SHADE*, *b6e6rl*, and *LSHADE* working alone gives us evidence if the hierarchical cooperation of four optimizers can outperform the best performing algorithm working alone. The comparison of *HiCo* with *HiCoxx* variants shows if three-island algorithm is better than the simpler variants with only two islands. This comparison brings also an insight into the role of the individual algorithms included in cooperation.

The algorithms are implemented in Matlab 2010a and this environment is used for experiments. The performance of the algorithms is tested on the CEC 2015 suite of 15 minimizations problems defined in the report [10]. Search range for all the test functions is $[-100, 100]^D$. The source code of functions in C was downloaded from the web page [10] and compiled for Matlab via mex command.

The tests were carried out at two levels of dimension, namely $D = 10$ and $D = 30$. 51 repeated runs were performed per the test function and the dimension of the problem except *HiCoxx* variants, where only 25 runs per problem were done. The run was stopped if the $MaxFES = D \times 10^4$ prescribed in [10] was reached. The best solution found in the run in form of function error $f_{min} - f(\boldsymbol{x}^*)$ was recorded (among other values), where f_{min} is the least function value and $f(\boldsymbol{x}^*)$ is the function value at the global minimum, which is known for the test problems.

The population size of the algorithms in the cooperation was set up as follows: $NP = 50$ for *b6e6rl* and *SHADE*, $NP = \mu = 25$ for *CMAES* and $NP^{init} = 125$ for *LSHADE*. The other control parameters were set up to their default values. In experiments with the algorithms running individually, the same size population was used except *LSHADE*, where $NP^{init} = 18 \times D$ was used as recommended in [21].

5 Results

The results of the experimental comparison of individual algorithms with *HiCo* for $D = 10$ are shown in Table 1, where the medians of function-error values are reported together with p values achieved by Kruskal-Wallis test of distribution-equality hypothesis. When $p < 0.05$, the null hypothesis (equivalent performance

Table 1 Medians of function error and results of Kruskal-Wallis test for individual algorithms and *HiCo*, $D = 10$

F	CMAES	SHADE	b6e6rl	LSHADE	HiCo	p
1	0	0	0	0	0	1.000
2	0	0	0	0	0	1.000
3	20	20.0169	20.0575	20.0073	20.0073	0.000
4	2.98488	2.99866	5.05762	3.97991	2.98488	0.000
5	133.558	129.581	146.234	15.5309	118.891	0.000
6	459.382	4.34509	0.41629	3.23841	1.57033	0.000
7	1.03306	0.25717	0.16884	0.23807	0.19322	0.000
8	75.0498	0.31807	0.37843	0.76884	0.42442	0.000
9	100	100.851	100.005	100	100	0.000
10	151.087	143.109	143.109	143.109	141.529	0.000
11	390.963	3.91969	3.16197	3.07095	3.08909	0.000
12	111.284	112.526	111.587	112.166	111.278	0.000
13	0.10857	0.09273	0.09273	0.09273	0.09273	0.000
14	6794	6677.01	6670.66	6670.66	6662.87	0.000
15	100	100	100	100	100	1.000

of the algorithms in comparison) is rejected. The results for $D = 30$ in the same structure are presented in Table 2. We can see that the performance of algorithms was different in most problems significantly. The equivalent performance was found only in three problems for $D = 10$ and in one problem for $D = 30$, where all the algorithms found the same solutions in all runs.

Based on the results of Kruskal-Wallis test, the ranks of performance were assigned to the algorithms in each problem. If the algorithms do not differ significantly, the same ranks were assigned to them corresponding to their average ranks. Then the sums of ranks on 15 problems were computed for each algorithm. The comparison of the ranks observed for *HiCo* and individual algorithms is shown in Table 3. From the sums of ranks is obvious that *HiCo* algorithm with hierarchical cooperation outperforms all the individual algorithms substantially. The medians of the function errors and the results of Kruskal-Wallis test for the comparison of the cooperative algorithms are shown in Table 4 for $D = 10$ and in Table 5 for $D = 30$, respectively. The performance of the algorithms does not differ significantly in about half problems of $D = 10$, while the significant difference in performance was observed in 13 out of 15 problems for $D = 30$.

Table 2 Medians of function error and results of Kruskal-Wallis test for individual algorithms and *HiCo*, $D = 30$

F	CMAES	SHADE	b6e6rl	LSHADE	HiCo	p
1	0	4632.65	14138.6	0	0	0.000
2	0	0	0.27878	0	0	0.000
3	20.9365	20.0747	20.2691	20.1047	20.0342	0.000
4	14.9244	44.3321	42.2026	25.0974	14.9244	0.000
5	981.276	1836.15	2066.05	1242.31	1064.97	0.000
6	1354.18	1579.09	2719.69	197.065	1151.27	0.000
7	7.24943	9.00941	7.01717	6.8496	5.87447	0.000
8	767.921	382.957	152.27	52.0179	196.2	0.000
9	106.046	109.138	106.037	107.076	105.965	0.000
10	710.835	868.635	665.241	616.434	660.477	0.000
11	400	597.514	416.695	556.304	400	0.000
12	107.082	111.226	110.132	109.363	107.143	0.000
13	0.01344	0.01081	0.01049	0.01073	0.01042	0.000
14	44872.2	43572.3	42786.9	42558.5	36775.4	0.000
15	100	100	100	100	100	1.000

Like in the previous comparison, the ranks of performance were assigned to the algorithms in each problem. If the algorithms do not differ significantly, the same ranks were assigned to them corresponding to their average ranks. Then the sums of ranks on 15 problems were computed for each algorithm. The comparison of the ranks observed for *HiCo* and *HiCoxx* variants is shown in Table 6.

Table 3 Ranks of the algorithms on test problems and the sum of ranks for each algorithm in comparison

	$D = 10$					$D = 30$				
F	CMAES	SHADE	b6e6rl	LSHADE	HiCo	CMAES	SHADE	b6e6rl	LSHADE	HiCo
1	3	3	3	3	3	2	4	5	2	2
2	3	3	3	3	3	2.5	2.5	5	2.5	2.5
3	1	4	5	2.5	2.5	5	2	4	3	1
4	1.5	3	5	4	1.5	1.5	5	4	3	1.5
5	4	3	5	1	2	1	4	5	3	2
6	5	4	1	3	2	3	4	5	1	2
7	5	4	1	3	2	4	5	3	2	1
8	5	1	2	4	3	5	4	2	1	3
9	2	5	4	2	2	3	5	2	4	1
10	5	3	3	3	1	4	5	3	1	2
11	5	4	3	1	2	1.5	5	3	4	1.5
12	2	5	3	4	1	1	5	4	3	2
13	5	2.5	2.5	2.5	2.5	5	4	2	3	1
14	5	4	2.5	2.5	1	5	4	3	2	1
15	3	3	3	3	3	3	3	3	3	3
Σ	54.5	51.5	46	41.5	31.5	46.5	61.5	53	37.5	26.5

Table 4 Medians of function error and results of Kruskal-Wallis test for *HiCo* and *HiCoxx* variants, $D = 10$

F	HiCo	HiCoSb	HiCoCb	HiCoCS	p
1	0	0	0	0	1.000
2	0	0	0	0	1.000
3	20.0073	20.0037	20.0056	20.0035	0.256
4	2.98488	3.98054	2.98488	2.98488	0.002
5	118.891	27.3323	130.33	22.0338	0.227
6	1.57033	0.46805	0.83673	6.82087	0.000
7	0.19322	0.11907	0.17329	0.14706	0.014
8	0.42442	0.32485	0.53236	0.45825	0.187
9	100	100.009	100	100	0.000
10	141.529	141.525	141.525	143.642	0.008
11	3.08909	2.9951	3.38306	3.4156	0.131
12	111.278	111.328	111.318	111.297	0.810
13	0.09273	0.09273	0.09433	0.09273	0.000
14	6662.87	6662.87	6670.66	6670.85	0.000
15	100	100	100	100	1.000

Table 5 Medians of function error and results of Kruskal-Wallis test for *HiCo* and *HiCoxx* variants, $D = 30$

F	HiCo	HiCoSb	HiCoCb	HiCoCS	p
1	0	19511.4	0	0	0.000
2	0	0.00958	0	0	0.000
3	20.0342	20.0185	20.0229	20.0194	0.000
4	14.9244	34.8353	18.9042	15.9193	0.000
5	1064.97	1550.18	1219.18	1016.31	0.000
6	1151.27	1678.02	1247.21	1328.02	0.010
7	5.87447	6.44536	6.247	6.81616	0.025
8	196.2	184.341	205.916	468.894	0.000
9	105.965	106.703	106.079	106.109	0.000
10	660.477	685.858	650.583	731.757	0.005
11	400	420.244	400	400	0.173
12	107.143	110.01	106.763	107.342	0.000
13	0.01042	0.01048	0.01053	0.01076	0.000
14	36775.4	42765.1	42665.6	43565.8	0.003
15	100	100	100	100	1.000

The sums of ranks are almost the same for the problems of $D = 10$, which means that the performance of the algorithms does not differ substantially. However, the performance of the algorithms differs apparently for the problems of $D = 30$. The winner is *HiCo* with three islands followed by *HiCoCb* variant, which is the second best performing algorithm. The performance of the other *HiCoxx* variants is worse.

Table 6 Ranks of the algorithms on test problems and the sum of ranks for each algorithm in comparison

	$D = 10$				$D = 30$			
F	HiCo	HiCoSb	HiCoCb	HiCoCS	HiCo	HiCoSb	HiCoCb	HiCoCS
1	2.5	2.5	2.5	2.5	2	4	2	2
2	2.5	2.5	2.5	2.5	2	4	2	2
3	2.5	2.5	2.5	2.5	4	1	3	2
4	2	4	2	2	1	4	3	2
5	2.5	2.5	2.5	2.5	2	4	3	1
6	3	1	2	4	1	4	2	3
7	4	1	3	2	1	3	2	4
8	2.5	2.5	2.5	2.5	2	1	3	4
9	2	4	2	2	1	4	2	3
10	3	1.5	1.5	4	2	3	1	4
11	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
12	2.5	2.5	2.5	2.5	2	4	1	3
13	2	2	4	2	1	2	3	4
14	1.5	1.5	3	4	1	3	2	4
15	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
Σ	37.5	35	37.5	40	27	46	34	43

6 Conclusion

The cooperation of evolutionary algorithms in the solution of the global minimization problem with boundary constraints was studied. A simple hierarchical model with one mainland and several islands was implemented and tested on the CEC 2015 suite of 15 functions of the dimension 10 and 30. The proposed model exploits only one-way migration from the islands to the mainland, no communication among islands during their runs occurs in the models in the test.

The model labeled *HiCo* with three islands was compared experimentally with all the individual algorithms taking part in the hierarchical model. The results of the comparison show that the proposed cooperation is beneficial for the performance, especially in the problems of dimension $D = 30$. Moreover, *HiCo* was also compared with hierarchical models labeled *HiCoxx*, where only two islands are used. The comparison reveals that the difference in performance is small for the $D = 10$ (*HiCo* was the second best) but significant difference was found in the problems of $D = 30$, where the proposed *HiCo* model was the best performing.

The results are promising for next research of more sophisticated models with controlled migration among the islands and the most recent adaptive DE variants [5, 6] included in cooperative model. Such models can bring an efficient optimization algorithm without deficiencies like stagnation and similar drawbacks.

Acknowledgments 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).

References

1. Brest, J., Greiner, S., Boškovič, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**, 646–657 (2006)
2. Bujok, P., Tvrdík, J., Poláková, R.: Differential evolution with rotation-invariant mutation and competing-strategies adaptation. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2253–2258 (2014)
3. Bujok, P., Tvrdík, J.: Parallel migration model employing various adaptive variants of differential evolution. In: ICAISC 2012—SIDE 2012, pp. 39–47. Springer-Verlag, Berlin, Heidelberg (2012)
4. Elsayed, S.M., Sarker, R.A., Essam, D.L., Hamza, N.M.: Testing united multi-operator evolutionary algorithms on the CEC2014 real-parameter numerical optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1650–1657 (2014)
5. Guo, S.M., Yang, C.C.: Enhancing differential evolution utilizing eigenvector-based crossover operator. IEEE Trans. Evol. Comput. **19**, 31–49 (2015)
6. Guo, S.M., Yang, C.C., Hsu, P.H., Tsai, J.S.H.: Improving differential evolution with successful-parent-selecting framework. IEEE Transactions on Evolutionary Computation (in press)
7. Hansen, N.: purecmaes.m (2009). https://www.lri.fr/hansen/purecmaes.m
8. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of the 1996 IEEE Internatinal Conference on Evolutionary Computation, pp. 312–317 (1996)
9. Lampinen, J., Zelinka, I.: On stagnation of differential evolution algorithm. In: MENDEL 2000, 6th International Conference on Soft Computing, pp. 76–83 (2000)
10. Liang, J.J., Qu, B.Y., Suganthan, P.N.: Problem definition and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization (2014). http://www.ntu.edu.sg/home/epnsugan/
11. Liang, J.J., Qu, B., Suganthan, P.N.: Ranking results of CEC14 special session and competition on real-parameter single objective optimization (2014). http://www3.ntu.edu.sg/home/epnsugan/
12. Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl. Soft Comput. **11**, 1679–1696 (2011)
13. Poláková, R., Tvrdík, J.: Cooperation of optimization algorithms: a simple hierarchical model. In: Proceedings of the IEEE Congress on Evolutionary Computation (accepted) (2015)
14. Poláková, R., Tvrdík, J., Bujok, P.: Controlled restart in differential evolution applied to CEC2014 benchmark functions. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2230–2236 (2014)
15. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**, 398–417 (2009)
16. Ruciński, M., Izzo, D., Biscani, F.: On the impact of the migration topology on the island model. Parallel Comput. **36**, 555–571 (2010)
17. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)
18. Storn, R., Price, K., Lampinen, J.: Differential Evolution—A Practical Approach to Global Optimization. Springer, Berlin, Germany (2005)

19. Tanabe, R., Fukunaga, A.: Evaluating the performance of SHADE on CEC 2013 benchmark problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1952–1959 (2013)
20. Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 71–78 (2013)
21. Tanabe, R., Fukunaga, A.: Improving the search performance of shade using linear population size reduction. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1658–1665 (2014)
22. Tvrdík, J.: Competitive differential evolution. In: Matoušek, R., Ošmera, P. (eds.) MENDEL 2006, 12th International Conference on Soft Computing, pp. 7–12 (2006)
23. Tvrdík, J.: Adaptation in differential evolution: a numerical comparison. Appl. Soft Comput. **9**, 1149–1155 (2009)
24. Tvrdík, J.: Self-adaptive variants of differential evolution with exponential crossover. Ser. Math.-Inform. **47**, 151–168 (2009) (Analele of West University Timisoara)
25. Tvrdík, J., Poláková, R., Veselský, J., Bujok, P.: Adaptive variants of differential evolution: towards control-parameter-free optimizers. In: Handbook of Optimization, pp. 423–449. Springer (2012)
26. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evol. Comput. **15**, 55–66 (2011)
27. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**, 945–958 (2009)
28. Zhang, Q., Yu, G., Song, H.: A hybrid bird mating optimizer algorithm with teaching-learning-based optimization for global numerical optimization. Stat. Optim. Inf. Comput. **3**, 54–65 (2015)

A Switched Parameter Differential Evolution for Large Scale Global Optimization – Simpler May Be Better

Swagatam Das, Arka Ghosh and Sankha Subhra Mullick

Abstract In this article we present two very simple modifications to Differential Evolution (DE), one of the most competitive evolutionary algorithms of recent interest, to enhance its performance for the high-dimensional numerical functions while still preserving the simplicity of its algorithmic framework. Instead of resorting to complicated parameter adaptation schemes or incorporating additional local search methods, we present a simple strategy where the values of the scale factor (mutation step size) and crossover rate are switched in a uniformly random way between two extreme corners of their feasible ranges for different population members. Also each population member is mutated either by using the DE/rand/1 scheme (where the base vector to be perturbed is a randomly chosen member from the population) or by using the DE/best/1 scheme (where the base vector is the best member of the population). The population member is subjected to that mutation strategy which was responsible for the last successful update at the same population index under consideration. Our experiments based on the benchmark functions proposed for the competitions on large-scale global optimization with bound constraints held under the IEEE CEC (Congress on Evolutionary Computation) 2008 and 2010 competitions indicate that the basic DE algorithm with these simple modifications can indeed achieve very competitive results against the currently best known algorithms.

Keywords Continuous optimization · Differential evolution · Large scale optimization · Success counter

S. Das (✉) · A. Ghosh · S.S. Mullick
Indian Statistical Institute, 203 B. T. Road, Kolkata 700 108, India
e-mail: swagatam.das@isical.ac.in

A. Ghosh
e-mail: arka_t@isical.ac.in

S.S. Mullick
e-mail: mullicksankhasubhra@gmail.com

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_9

1 Introduction

Over the past few decades, several families of evolutionary computing algorithms have been proposed for solving bound-constrained global optimization problems. Performances of these algorithms remain considerably good for problems with moderate number of decision variables or dimensions. However, most of them face difficulties in locating the global optimum with sufficient accuracy and without consuming too much Function Evaluations (FEs) as the number of dimensions of the search space increases beyond 100 or so. This is not surprising and is primarily caused by the exponential increase of the search volume with dimensions. Consider placing 100 points onto a real interval, say [0,1]. To obtain a similarly dense coverage, in terms of distance between adjacent points, the 10-dimensional space $[0,1]^{10}$ would require $100^{10} = 10^{20}$ points. The previously mentioned 100 points now appear as isolated points in a vast empty space. Usually the distance measures break down in higher dimensionalities and a search strategy that is valuable in small dimensions might be useless in large or even moderate dimensional search spaces.

Many real world problems demand optimization of a large number of variables. A few typical examples of such problems are shape optimization [1, 2], high-dimensional waveform inversion [3], and large scale economic load dispatch (involving 140 units or more) [4]. Recently researchers have been paying attention to the issue of designing scalable nature-inspired optimization techniques for optimizing very high dimensional functions. The ongoing interest of the scientific community is also evident from participations in the competitions on large scale single objective global optimization with bound constraints held under the IEEE CEC (Congress on Evolutionary Computation) 2008 and 2010 [5, 6].

The evolutionary methods for high-dimensional global optimization problems can be roughly categorized into three classes: the cooperative co-evolutionary methods, the micro Evolutionary Algorithms (EAs) and the Local Search (LS) based methods. Cooperative Co-Evolutionary Algorithms (CCEAs) [7–11] are well-known for solving high dimensional optimization problems and they result from an automatic divide and conquer approach. Recently some promising Cooperative Co-evolutionary (CC) algorithms were proposed like the CC versions of the Particle Swarm Optimization (CCPSO and CCPSO2) [10] and CC with Differential Grouping [11]. Micro–EAs (see for example [12–15]) are instances of typical EAs characterized by small population size and often simple fitness functions. Different forms of Memetic Algorithms (MAs) [16–19] developed by combining an LS method with a global evolutionary optimizer have been frequently applied to solve large scale function optimization problems.

Differential Evolution (DE) [20, 21] currently stands out as a very competitive evolutionary optimizer for continuous search spaces. Several attempts have been made to improve the performance of DE for moderate to high dimensional function optimization problems. Some noted DE-variants of current interest involve success-history based parameter adaptation strategies (like SaDE [22], JADE [23]), new mutation and crossover strategies (like Pro-DE [24], MDE-pBX [25]), and combining various offspring generation strategies (CoDE [26], EPSDE [27] etc.). For high-dimensional problems (more than 500 dimensions) DE has been adopted by methods encompassing all the three algorithmic philosophies outlined above. Owing to its inherent simplicity, DE was used as the base optimizer in Yang et al.'s first work [9] on random grouping based CCEAs. Zamuda et al. [28] extended DE by log-normal self-adaptation of its control parameters and by using cooperative co-evolution as a dimensional decomposition mechanism. Parsopoulos developed a cooperative micro-DE [29] for large scale global optimization. The self-adaptive DE was hybridized with MTS for large scale optimization by Zhao et al. [30]. Some other approaches of improving DE for high-dimensional function optimization can be found in [31–34].

We can see that in order to cope with the growing complexity of the problems to be solved, DE has been subjected to several modifications. However, despite the reported performance improvements, the improved DE algorithms are very often lacking one very important thing, that is the simplicity of the DE framework – the very reason why DE was and is loved by the practitioners of evolutionary computation. The present work is motivated by the question that can we improve DE for very high dimensional search spaces by simple parameter control strategies and by combining the basic ingredients of DE without any additional computation overheads (likely to be caused by external achieves, proximity and rank based parent selection schemes, additional local search schemes, keeping the long records of successful individuals etc.).

In this paper we present a simple DE scheme where the two crucial control-parameters of DE, namely the scale factor (equivalent to the mutation step size) F and the crossover rate Cr are switched between their respective limiting values in a uniformly random manner for each offspring generation process. Also each population member is mutated using either the DE/best/1 strategy or the DE/rand/1 strategy. The difference between these two strategies lies in the selection of the base vector to be perturbed. In case of DE/best/1, the base vector that has to be perturbed with the scaled difference of any two distinct population members is the best vector in the population yielding greatest fitness (i.e. smallest objective function value for a minimization problem). On the other hand, for the DE/rand/1 scheme, the base vector is a randomly chosen member from the current population. Each individual undergoes either of the two possible mutation strategies based on which strategy generated a successful offspring (which replaced the parent during selection) last time for the same population index. Thus, the choice of the mutation strategy depends on a unit length success memory of the record of just the last successful update. The proposed algorithm requires no tunable control parameter and is very easy to implement.

Switching of the scale factor between two extreme values (here 0.5 and 2) provides scopes for coarse search of large regions as well as refined search of smaller basins of attraction. Similarly by switching Cr values between 0 and 1, a balance between coordinate-wise search and generation of rotationally invariant search moves can be stricken. Our experiments indicate that the simple parameter switching coupled with the mixing of DE/best/1 and DE/rand/1 strategies can significantly improve the performance of DE on the high-dimensional function optimization problems. This conclusion is reached through a rigorous performance comparison of the proposed DE scheme with that of some of the most well-known large-scale optimizers including the winners of the two CEC competitions. While it is very hard (if not impossible) to analytically justify the suitability of the simple changes made to DE, we undertake some empirical studies based on the population spread and diversity to highlight the effectiveness of each of the modifications suggested.

2 The DE Algorithm

The initial generation of a standard DE algorithm consists of the four basic steps – initialization, mutation, recombination or crossover, and selection, of which, only last three steps are repeated into the subsequent DE generations. The generations continue till some termination criterion (such as exhaustion of maximum functional evaluations) is satisfied.

2.1 Initialization

DE begins search for the global optima in the *D* dimensional real parameter space by initiation of a random population of *Np* real-valued vectors whose components represent the *D* parameters of the optimization problem. A generalized notation used to identify the i^{th} solution (real parameter vector) of the present generation *G* can be shown as:

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots x_{D,i,G}].$$

Given the decision space bounds, $\vec{X}_{\max} = [x_{1,\max}, x_{2,\max}, \ldots x_{D,\max}]$ and $\vec{X}_{\min} = [x_{1,\min}, x_{2,\min}, \ldots x_{D,\min}]$, the j^{th} dimension of i^{th} individual can be initialized as:

$$x_{i,j} = x_{j,\min} + rand_{i,j} \times (x_{j,\max} - x_{j,\min}), \tag{1}$$

where $rand_{i,j}$ is a uniformly distributed random number lying in the range [0, 1] and it is instantiated anew for each ordered pair (*i*, *j*).

2.2 Mutation

In DE terminology, a population member (say *i*) of the current generation, known as the target vector, is chosen and is *differentially* mutated with the scaled difference vector(s) $(\vec{X}_{r_1,G} - \vec{X}_{r_2,G})$ to produce a mutant or *donor vector*. It is to be noted that the indices r_1 and r_2 are sampled from {1,2,…, Np} are different from the running index i and $(r_1, r_2 \in \{1, 2, \ldots Np\} \backslash \{i\})$. A scaling factor F, usually lying in the range [0.4, 2], scales the difference vector(s). Two commonly used DE mutation strategies are listed below:

$$DE\ /rand\ /1 : \vec{V}_{i,G} = \vec{X}_{r_1,G} + F\left(\vec{X}_{r_2,G} - \vec{X}_{r_3,G}\right), \tag{2a}$$

$$DE\ /best\ /1 : \vec{V}_{i,G} = \vec{X}_{best,G} + F.\left(\vec{X}_{r_1,G} - \vec{X}_{r_2,G}\right), \tag{2b}$$

where r_1, r_2, r_3 are mutually exclusive indices that are stochastically selected from $\{1, 2, \ldots, Np\}$.

2.3 Crossover

In DE, the crossover step aims to combine the individual components of the parent and the mutant vector into a single offspring commonly known as *trial vector* $\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, \ldots u_{D,i,G}]$ DE primarily employs either of the two crossover strategies: *exponential* (two-point modulo) and *binomial* (uniform). Binomial crossover is preferred since it does away with the inherent representational bias in n-point crossover by simulating D random trials. Moreover a recent work [35] attributing to the sensitivity of crossover to population size has reported the exponential variant to be more prone as compared to its binomial counterpart. Owing to the aforesaid observations, here we employ binomial crossover to form the trial vector.

In order to implement the binomial crossover, the control parameter *Crossover rate* (Cr) is set to a fixed value lying in the range [0,1] and then D independent random numbers, between 0 and 1, are sampled uniformly and compared with Cr to decide which component is to be included in the trial vector. The method is outlined as:

$$u_{j,i,G} = \begin{cases} \{v_{j,i,G}, & if\ rand_{i,j} < CR\ and\ j = j_r, \\ & x_{j,i,G}\ otherwise, \end{cases} \tag{3}$$

where j_r is a randomly chosen index from {1, 2, …, D} and it ensures that at least one component from the mutant vector is present in the offspring produced.

2.4 Selection

Finally, a selection process is performed through a one-to-one competition between the parent and the offspring to maintain a constant population size. The selection process can be described as:

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{\mathrm{i}} \; if \; f\left(\vec{U}_{\mathrm{i}}\right) \leq f\left(\vec{X}_i\right), \\ \vec{X}_i \; otherwise, \end{cases} \tag{4}$$

where $f(.)$ is the objective function to be minimized.

3 The Proposed Method

DE has 3 primary control parameters: the scale factor F, and the crossover rate Cr, and the population size Np. The performance of DE largely depends on F and Cr. Several efforts have been made in the past to control and adapt the value of these two parameters so that the algorithm may strike a balance between its explorative and exploitative behaviours on different fitness landscapes. Both of these parameters have their own allowable ranges. It is easy to see that Cr is similar to a probability value and hence it should lie in [0, 1]. Zaharie [36] derived a lower limit of F and her study revealed that if F be sufficiently small, the population can converge even in the absence of selection pressure. Ronkkonen et al. [37] stated that typically $0.4 < F < 0.95$ with $F = 0.9$ can serve as a good first choice. They also opine that Cr should lie in (0, 0.2) when the function is separable, while in (0.9, 1) when the function's variables are dependent. As is evident from [21] and the therein, numerous approaches have been proposed to improve the performance of DE by controlling or self-adapting these two control parameters and also by automating the choice of the appropriate offspring generation strategy. However, very often such methods may necessitate additional computational burdens, which, somewhat sacrifice the simplicity of DE. Thus, the question that naturally comes up is whether we can retain efficient search behaviour and adequate exploration-exploitation trade-off by doing something very simple? Can such easy modifications still result into a very competitive performance against the existing state-of-the-art? This article presents a humble contribution in this context.

The purpose of scaling factor F is to add weight to the difference vector and add it to base vector to produce mutant/donor vector. It is established in the study that F is strictly positive and greater than zero. A large value of F will support exploration, i.e. more of the feasible search volume can be covered. This property can be often desirable for solving high-dimensional optimization problems, since they possess a large search space. But, exploration is not helpful for converging and fine tuning of the solutions, necessary for detecting the optimum. A small value of F will serve the purpose of exploitation and support the convergence towards a solution.

In our proposal, for each population member, the F value is switched between 0.5 and 2 in a uniformly randomized way. Note that $F = 2$ is somewhat an unusual choice since this extreme value has not been reported for DE in commonly available papers. However, our experiments indicate that for the large scale problems this value can indeed enhance the performance than switching F between 0.5 and 1. When F is 2, the difference vector gets a higher importance. Consequently, the newly generated mutant point will be thrown relatively far from current base point, thereby enhancing the chances of venturing unexplored regions of the search space. When F takes a value of 0.5, the newly generated mutant point lies near by the base point as the difference vector gets very less weight. Thus it enhances the certainty of local neighborhood search. In Fig. 1 two possible distribution of the donor vectors have been shown around two mutant points generated by perturbing a base vector $\vec{X}_{base}$ with two values of the scale factor, $F = 0.5$ and $F = 2$ on a 2D search space.

Similarly for each individual, the Cr value is switched between 0 and 1. Note that this means in our proposed DE variant can either the mutant/donor vector is directly accepted as the final offspring (for $Cr = 1$) or the final offspring differs from the target (parent) vector at a single index determined by j_r (for $Cr = 0$). While the former situation corresponds to the generation of rotationally invariant points, the latter implies an axis parallel movement of the solution points.

Note that a plethora of DE variants has been published with different kinds of parametric (like sampling from a Cauchy or Gaussian distribution, see e.g. [22, 23]) and non-parametric (sampling from a uniform distribution like [31]) randomization in the tuning of F and Cr. However, such switching between only two extreme values has never been proposed earlier. In what follows we name the resulting DE variant as SWDE (Switching DE).

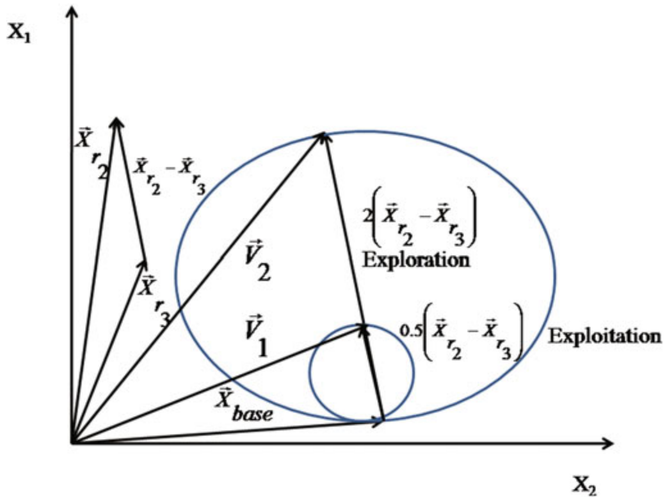

X_1
$\vec{X}_{r_2}$
$\vec{X}_{r_2} - \vec{X}_{r_3}$
$2(\vec{X}_{r_2} - \vec{X}_{r_3})$
Exploration
$\vec{V}_2$
$\vec{X}_{r_3}$
$\vec{V}_1$
$0.5(\vec{X}_{r_2} - \vec{X}_{r_3})$
Exploitation
$\vec{X}_{base}$
X_2

Fig. 1 Effect of $F = 2$ and $F = 0.5$ in DE mutation

Each individual in SWDE can be mutated by any of the commonly used mutation strategies in DE. It is now well-known that DE/best/1 induces somewhat greedy search behaviour and hence can be recommended for unimodal functions.

On the other hand, DE/rand/1 introduces more randomization (since different base vectors are perturbed to generate the mutant points for different individuals) and explorability and is, hence, suitable for multimodal functions. In our proposal we use a simple strategy for choosing any one of these two basic mutation schemes based on the success record of just the last successful update at the same population index. This means, at the first generation an individual is mutated either by DE/rand/1 or DE/best/1 chosen randomly. If the corresponding target individual is replaced by the trial (offspring), then the mutation scheme can be considered as a good choice, and will be used for that individual (of same index) in the next generation as well. If the trial is not selected for the individual, then the mutation strategy is unsuccessful, and in the next generation the other mutation scheme will be used for that individual.

We refer to this DE variant that combines the parameter switching scheme with a success-based selection of the mutation strategies as SWDE_Success (SWitching DE with Success-based selection of mutation scheme). Note that SWDE_Success does not require any control parameter to tune, except for the population size *Np*, which, however, is not really treated as a control parameter and is kept constant for most of the representative literature on DE. There is no need to fix any initial values for *F* and *Cr*, knowing only their feasible ranges would be fine. There is no parametric probability distribution whose parameters (like mean, variance, offset etc.) are required to be tuned.

Before moving on to the comparative study on standard benchmark suites, we would like to illustrate that DE with these simple modifications can indeed better preserve the population diversity, and thus can be helpful in retaining the useful information about promising search regions in a better way. Measurement of diversity level of the total population during the optimization work is another important aspect of empirical analysis, because maintaining diversity along the search process is another important aspect of large scale optimization. In proposed work "distance-to-average point" measurement for diversity of the population P_G at generation *G*, as presented in [38], is used as follows:

$$diversity(P_G) = \frac{1}{Np \times L} \sum_{i=1}^{Np} \sqrt{\sum_{j=1}^{D} \left(x_{ij} - \bar{x}_j\right)^2}, \tag{5}$$

where *Np* represents population size, *L* is the length of the longest diagonal in the search space of dimension *D* and $\bar{x}_j$ is the average value of *j*-th dimension of the vector. Variation of population diversity with respect to iteration as defined in (6) for the population is plotted in Fig. 2 for the Rastrigin's function in 50D. It is clear from the figure that the population of SWDE_Success never loses diversity prematurely. In this figure the red line depicts population diversity for SWDE_Success and blue one represents the same for standard DE.

In Fig. 3 we illustrate the distribution of population members of standard DE/best/1/bin (F = 0.8 and Cr = 0.8) and the proposed SWDE_Success on a 2D parameter space of the Rastrigin's function. The figures present screenshots of the

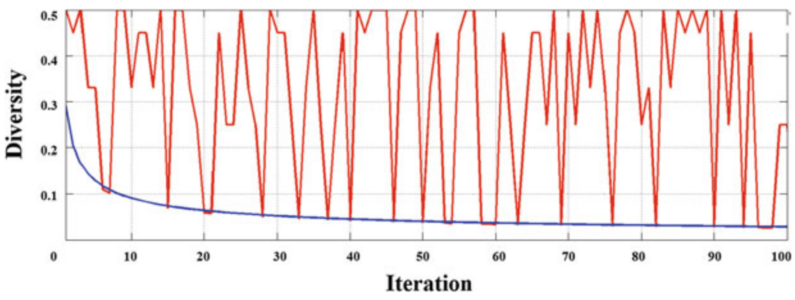
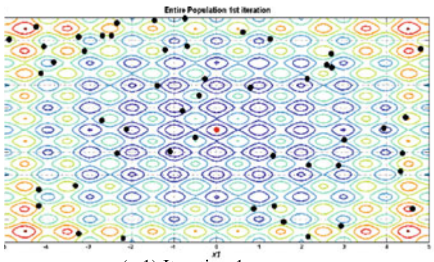
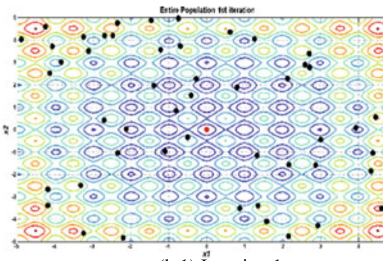


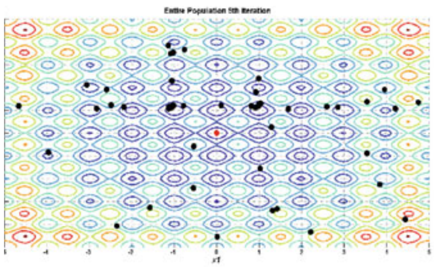
Fig. 2 Comparison of variation of population diversity with number of iterations between SWDE_Success and standard DE/best/1/bin



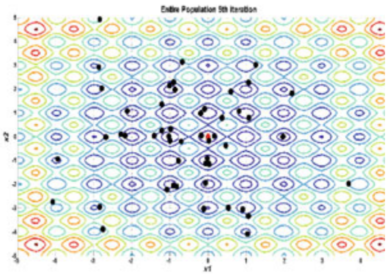
(a.1) Iteration 1



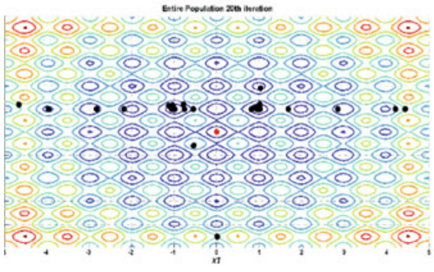
(b.1) Iteration 1



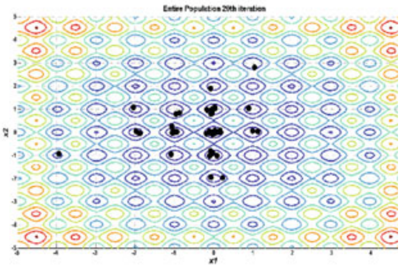
(a.2) Iteration 5



(b.2) Iteration 5



(a.3) Iteration 20



(b.3) Iteration 20

Global Optimum • Population Members

Fig. 3 Screen-shots of the evolving populations on the iso-contours of the 2D Rastrigin's function for (a) standard DE/best/1 scheme with $F = 0.8$ and $Cr = 0.8$ (b) SWDE_Success

population at 1st, 5-th and 20-th iterations. Note that for the two algorithms, the iterations were started from the same initial population, so that the different spread of the evolving populations may be attributed to their internal search mechanisms only. It can be seen that the population members following the SWDE_Success scheme can capture the global optimum more efficiently while still preserving the population diversity.

4 Experiments and Results

A popular choice for evaluating the performance of the DE algorithm is to use the benchmark suite proposed for the IEEE CEC (Congress on Evolutionary Computation) competitions. These suites contain collection of functions of diverse nature, which can successfully validate the performance of an optimization algorithm in a variety of scenarios.

The CEC 2008 [5] and CEC 2010 [6] test suites are specially designed with large scale minimization problems (i.e. of dimensions $D = 100$, 500, and 1000), and thus, useful for testing our proposed DE variant (SWDE_Success). Following standard procedure, the mean and standard deviation of the error value is used to measure the performance of an algorithm. The error is calculated as the difference between the actual value of the global optimum and the obtained value of the optimum. Only for function F7 of CEC 2008, the absolute value of the obtained optimum is recorded and compared, because for that function, the globally optimal function value is unknown. The population size has been taken as 100 for SWDE_Success in all the cases. To evaluate the scalability of the algorithm in the worse condition, the comparison is rendered on 1000 and 2000 dimensional problems.

For all the CEC 2008 benchmark problems and for all algorithms compared, the maximum number of Function Evaluations (FEs) corresponding to each run was taken to be $5000 \times D$, where D denotes the dimensionality of the functions following [5]. Similarly for the CEC 2010 benchmarks, the maximum number of FEs corresponding to each run was fixed to $3000 \times D$ [6, 18]. The parametric settings for all the peer algorithms were kept similar to their respective literatures. For SWDE_Success, the population size was fixed to $Np = 100$ for all 1000D problems and $Np = 150$ for the 2000D problems. We find that this choice (which is standard and straight forward) provides consistently good performance on the used benchmarks, and little improvement takes place with increased execution time if we increase the population size any further.

A non-parametric statistical test called Wilcoxon's rank sum test for independent samples [39] is conducted at the 5 % significance level in order to judge whether the results obtained with the best performing algorithm differ from the final results of rest of the competitors in a statistically significant way. In all result tables the statistical test results are summarized in the following way. If the final error yielded by an algorithm is statistically significantly different from that of the best performing algorithm on a particular function, then the mean error of the former is marked with a † symbol. If the difference of the error values found by one algorithm is not statistically significant, as compared to its best competitor, then the mean of this algorithm is marked with a ≈. The best performing algorithm in each case is marked with boldface.

In order to demonstrate how the proposed parameter switching scheme works harmoniously with the success-based mutation scheme, we begin with a comparative study among the proposed SWDE_Success, the standard DE/best/1/bin scheme with fixed F and Cr ($F = 0.8$, $Cr = 0.9$) and a SWDE that uses only DE/best/1 mutation scheme for all its population members. The obtained results are demonstrated in Table 1, which shows that SWDE provides much better results than DE/best/1 with fixed F and Cr values. Also it can be observed that SWDE_Success yields statistically better results as compared to both SWDE and DE/best/1 on all the 7 functions of the CEC 2008 test bed.

To compare the performance of SWDE_Success against the existing state-of-the-art, we consider the results of five other algorithms customised for large scale global optimization. Two of them are the variants of the Particle Swarm Optimisation (PSO) algorithm namely CCPSO2 [10] and EPUS-PSO (Efficient Population Utilization Strategy for Particle Swarm Optimization) [40], which use a variable grouping technique and an efficient population management scheme respectively. The third one is Sep-CMA-ES [41] a scalable variation of the popular CMA-ES optimization technique, which performs faster and better than the original algorithm, especially on separable functions. The fourth is MTS (Multiple Trajectory Search) [16], a hybrid local search method. The fifth one is MLCC (Multi-Level Cooperative Co-evolution) [42]. The results are listed Table 2, where SWDE_Success outperformed others, for all the functions except F6. For F6 CCPSO2 performed slightly better than SWDE_Success, however, result of the rank sum test indicates that this difference is not statistically significant. Thus, SWDE_Success is found to be more consistent on this benchmark suite. In terms of the average rank SWDE_Success is the clear winner, followed by CCPSO2 and MTS.

In Tables 3, 4 and 5 SWDE_Success results are compared with 11 other evolutionary optimizers including some recent DE-variants (DECC-ML, DECC-CG, DECC-DG, and DE/best/1/bin) on the CEC 2010 benchmarks. Out of the 20 high-dimensional functions SWDE_Success provided statistically significantly better results compared to all its peer algorithms on 14 functions, and ranked second in 3 functions. DECC-ML [43] performed best in three functions, namely F3, F11 and F16, jDELsgo [28] performed better on two functions F6 and F19, and MA-SW-Chains [18] performed better than others only in one case of F12. The rank sum test results indicate that on F3, the result of SWDE_Success is not statistically significantly different from the best result given by DECC_ML. Similarly for F6, the best result yielded by jDElsgo is statistically equivalent to that of SWDE_Success. For CEC 2010 functions also SWDE_Success holds the minimum average rank, the closest followers are jDELsgo, and MA-SW-Chains.

To further test the scalability of the proposed algorithm, three CEC 2008 functions (1 unimodal and 2 multimodals) of 2000 dimensions are used, as was also

Table 1 Performance improvement by the proposed algorithm (SWDE_Success) from DE/best/1/bin and SWDE (D = 1000)

Func.	F1	F2	F3	F4	F5	F6	F7
DE/best/1/bin	9.88e + 05†	1.291e + 04†	7.10e + 11†	6.53e + 03†	8.46e + 03†	1.76e + 01†	−2.13e + 01†
	1.25e + 05	5.62e + 01	2.35e + 09	2.55e + 02	3.52e + 02	7.89e + 00	1.15e + 01
SWDE	9.95e − 18†	9.046e + 01†	6.33e + 05†	5.23e − 09†	1.61e − 14†	2.50e − 09†	−7.00e + 03†
	1.235 − 19	5.46e + 01	5.73e + 01	4.61e − 05	2.25e − 12	1.16e − 07	1.22e + 01
SWDE_	**0.00e + 00**	**2.24e + 00**	**1.03e − 03**	**0.00e + 00**	**2.74e − 22**	**4.31e − 12**	**−9.85e + 04**
Success	**0.00e + 00**	**2.01e + 01**	**1.85e + 01**	**0.00e + 00**	**1.25e − 18**	**3.25e − 19**	**2.56e − 02**

Table 2 Performance of 6 large scale evolutionary optimizers including SWDE_Success on CEC 2010 benchmark suite ($D = 1000$)

Function	F1	F2	F3	F4	F5	F6	F7	Avg. Rank
CCPSO2 [10]	4.99e − 13† 9.51e − 14	7.55e + 01† 4.25e + 01	1.30e + 03† 2.15e + 02	1.17e − 03† 3.27e − 03	1.17e − 03† 3.25e − 03	**1.01e − 12** **1.68e − 13**	−1.44e + 04† 8.27e + 01	
RANK	3	4	4	3	4	1	3	3.14
Sep-CMA-ES [41]	7.79e − 15† 1.22e − 15	3.10e + 02† 9.22e + 00	9.09e + 02† 4.22e + 01	5.25e + 03† 2.48e + 02	3.91e − 04† 1.96e − 03	2.16e + 01† 3.19e − 01	−1.24e + 04† 9.36e + 01	
RANK	2	6	3	5	5	8	5	4.85
EPUS-PSO [40]	5.49e + 02† 2.82e + 01	4.55e + 01† 4.00e − 01	8.31e + 05† 1.56e + 05	7.56e + 03† 1.50e + 02	5.80e + 00† 3.92e − 01	1.84e + 01† 2.49e + 00	−6.68e + 03† 3.18e + 01	
RANK	6	3	6	7	6	5	6	5.57
MLCC [42]	8.25e − 13† 5.59e − 14	1.28e + 02† 4.75e + 00	1.77e + 03† 1.25e + 02	1.42e − 10† 3.31e − 10	4.17e − 13† 2.79e − 14	1.06e − 12 ≈ 7.68e − 14	−1.49e + 04† 1.52e + 01	
RANK	4	5	5	2	2	3	2	3.28
MTS [16]	1.21e − 03† 6.14e − 03	4.60e + 01† 1.5e + 00	1.77e − 02† 7.81e − 03	2.81e + 02† 4.74e + 02	9.45e − 08† 2.61e − 07	7.20e − 04† 2.67e − 03	−1.31e + 04† 3.45e + 01	
RANK	5	2	2	4	3	6	4	3.71
SWDE_ Success	**0.00e + 00** **0.00e + 00**	**2.24e + 00** **2.01e + 01**	**1.03e − 03** **1.85e − 02**	**0.00e + 00** **0.00e + 00**	**2.74e − 22** **1.25e − 18**	4.31e − 12 ≈ 3.25e − 19	**−9.85e + 04** **2.56e − 02**	
RANK	1	1	1	1	1	2	1	**1.14**

Table 3 Performance of 12 large scale evolutionary optimizers including SWDE_Success on CEC 2010 benchmark suite (F1 − F7, $D = 1000$)

Functions								
Algorithms		F1	F2	F3	F4	F5	F6	F7
jDElsgo [28]	Mean	8.85e − 20†	1.24e − 01†	3.79e − 12≈	8.01e + 10†	9.71e + 07†	**1.69e − 08**	4.32e − 02†
	SD	4.50e − 20	3.44e − 01	5.02e − 12	3.05e + 10	1.42e + 07	**4.02e − 08**	6.35e − 02
	(Rank)	(7)	(2)	(6)	(2)	(5)	**(1)**	(2)
DECC-ML [43]	Mean	1.36e − 25†	2.15e + 02†	**1.14e − 13**	3.53e + 12†	2.98e + 08†	7.94e + 05†	1.21e + 08†
	SD	1.81e − 25	2.91e + 01	**8.22e − 15**	1.51e + 12	9.32e + 07	3.87e + 06	7.65e + 07
	(Rank)	(3)	(7)	**(1)**	(7)	(10)	(8)	(10)
DASA [45]	Mean	1.51e − 21†	8.44e + 00†	7.21e − 11†	5.04e + 11†	6.21e + 08†	1.98e + 07†	7.75e + 00†
	SD	2.32e − 21	2.50e + 00	8.22e − 12	2.25e + 11	7.81e + 07	4.41e + 04	3.11e + 00
	(Rank)	(5)	(5)	(8)	(5)	(12)	(11)	(3)
DMS-PSO-SHS [46]	Mean	5.55e − 15†	8.52e + 01†	5.51e − 11†	2.41e + 11†	8.35e + 07†	8.25e − 02†	1.95e + 03†
	SD	4.02e − 14	2.02e + 01	3.21e − 10	3.31e + 10	6.15e + 06	9.95e − 01	1.55e + 02
	(Rank)	(6)	(6)	(7)	(3)	(4)	(4)	(6)
SDENS [34]	Mean	5.72e − 06†	2.22e + 03†	2.70e − 05†	5.12e + 12†	1.12e + 08†	2.23e − 04†	1.21e + 08†
	SD	4.42e − 06	8.92e + 01	1.52e − 05	2.12e + 12	2.23e + 07	4.56e − 05	6.52e + 07
	(Rank)	(10)	(10)	(9)	(9)	(6)	(3)	(8)
EOEA [44]	Mean	2.21e − 23†	3.61e − 01†	1.61e − 13≈	3.07e + 12†	2.26e + 07†	3.86e + 06†	1.21e + 02†
	SD	2.81e − 23	6.71e − 01	1.11e − 14	1.66e + 12	5.96e + 06	4.96e + 05	1.51e + 02
	(Rank)	(4)	(3)	(2)	(6)	(3)	(9)	(5)
DECC-G [9]	Mean	2.94e − 07†	1.34e + 03†	1.38e + 00†	1.75e + 13†	2.64e + 08†	4.91e + 06†	1.61e + 08†
	SD	8.64e − 08	3.24e + 01	9.75e − 02	5.34e + 12	8.44e + 07	8.01e + 05	1.31e + 08
	(Rank)	(9)	(9)	(10)	(11)	(9)	(10)	(11)
MLCC [42]	Mean	1.55e − 27†	5.52e − 01†	9.82e − 13≈	9.60e + 12†	3.80e + 08†	1.61e + 07†	6.81e + 05†
	SD	7.62e − 27	2.22e + 00	3.72e − 12	3.40e + 12	6.90e + 07	4.91e + 06	7.31e + 05
	(Rank)	(2)	(4)	(4)	(10)	(11)	(12)	(8)

(continued)

Table 3 (continued)

Functions								
Algorithms		F1	F2	F3	F4	F5	F6	F7
MA-SW-Chains [18]	Mean	2.09e − 14†	8.11e + 02†	7.21e − 13≈	3.52e + 11†	1.67e + 08†	8.13e + 04†	1.03e + 02†
	SD	1.98e − 14	5.81e + 01	3.41e − 13	3.12e + 10	1.07e + 08	2.83e + 05	8.71e + 01
	(Rank)	(7)	(8)	(3)	(4)	(8)	(7)	(4)
DECC-DG [11]	Mean	5.41e + 03†	4.33e + 03†	1.61e + 01†	4.78e + 12†	1.51e + 08†	1.63e + 01†	1.12e + 04†
	SD	2.02e + 04	1.93e + 02	3.31e − 01	1.48e + 12	2.12e + 07	2.72e − 01	7.41e + 03
	(Rank)	(11)	(11)	(11)	(8)	(7)	(5)	(7)
DE/best/1/bin	Mean	3.52e + 05†	9.30e + 03†	3.53e + 03†	5.25e + 19†	5.23e + 06†	2.55e + 01†	1.08e + 11†
	SD	2.69e + 05	1.3e + 04	2.23e + 02	9.23e + 17	9.63 + 05	1.21e + 01	9.08e + 10
	(Rank)	(12)	(12)	(12)	(12)	(2)	(6)	(12)
SWDE_Success	Mean	**1.59e − 30**	**1.21e − 03**	1.21e − 12≈	**1.00e + 06**	**3.39e + 04**	2.33e − 08≈	**1.05e − 03**
	SD	**1.35e − 25**	**2.35e − 06**	7.35e − 12	**2.54e + 02**	**3.37e + 02**	1.01e − 07	**1.01e − 02**
	(Rank)	**(1)**	**(1)**	(5)	**(1)**	**(1)**	(2)	**(1)**

Table 4 Performance of 12 large scale evolutionary optimizers including SWDE_Success on CEC 2010 benchmark suite (F8 − F14, $D = 1000$)

Functions Algorithms		F8	F9	F10	F11	F12	F13	F14
jDElsgo [28]	Mean	3.12e + 06†	3.13e + 07†	2.59e + 03†	2.21e + 01†	1.22e + 04†	7.12e + 02†	1.67e + 08†
	SD	3.23e + 06	5.02e + 06	3.18e + 02	1.53e + 01	2.05e + 03	1.39e + 02	2.09e + 07
	(Rank)	(2)	(5)	(5)	(4)	(6)	(2)	(6)
DECC-ML [43]	Mean	3.41e + 07†	5.93e + 07†	1.21e + 04†	**1.79e − 13**	3.56e + 06†	1.12e + 03†	1.70e + 08†
	SD	3.52e + 07	4.71e + 06	2.60e + 02	**9.55e − 15**	1.35e + 05	4.30e + 02	1.45e + 07
	(Rank)	(7)	(8)	(12)	**(1)**	(12)	(3)	(7)
DASA [45]	Mean	4.97e + 07†	3.61e + 07†	7.26e + 03†	1.97e + 02†	1.70e + 03†	1.20e + 03†	1.01e + 08†
	SD	8.94e + 07	4.78e + 06	2.61e + 02	1.52e − 01	2.24e + 02	7.34e + 02	7.85e + 06
	(Rank)	(9)	(6)	(10)	(9)	(4)	(4)	(4)
DMS-PSO-SHS [46]	Mean	1.25e + 07†	8.55e + 06†	5.59e + 03†	3.29e + 01†	6.15e + 02†	1.25e + 03†	1.76e + 07†
	SD	1.95e + 06	6.55e + 05	5.19e + 02	2.99e + 00	6.05e + 01	1.06e + 02	1.56e + 06
	(Rank)	(4)	(2)	(8)	(6)	(3)	(5)	(2)
SDENS [34]	Mean	5.15e + 07†	5.61e + 08†	6.81e + 03†	2.22e + 02†	4.12e + 05†	2.13e + 03†	1.83e + 09†
	SD	2.15e + 07	5.71e + 07	5.61e + 02	5.02e − 01	4.22e + 04	1.03e + 03	2.33e + 08
	(Rank)	(10)	(12)	(9)	(11)	(10)	(9)	(11)
EOEA [44]	Mean	1.01e + 07†	4.62e + 07†	1.02e + 03†	3.82e + 01†	1.57e + 04†	1.54e + 03†	1.64e + 08†
	SD	1.21e + 07	4.72e + 06	6.92e + 01	1.62e + 01	2.50e + 03	4.19e + 02	8.94e + 06
	(Rank)	(3)	(7)	(3)	(8)	(7)	(6)	(5)
DECC-G [9]	Mean	6.43e + 07†	3.20e + 08†	1.05e + 04†	2.30e + 01†	8.91e + 04†	5.11e + 03†	8.01e + 08†
	SD	2.81e + 07	3.36e + 07	2.94e + 02	1.75e + 00	6.81e + 03	3.91e + 03	6.02e + 07
	(Rank)	(11)	(11)	(11)	(5)	(9)	(10)	(10)
MLCC [42]	Mean	4.37e + 07†	1.22e + 08†	3.45e + 03†	1.91e + 02†	3.41e + 04†	2.01e + 03†	3.11e + 08†
	SD	3.44e + 07	1.29e + 07	8.75e + 02	6.91e − 01	4.21e + 03	7.21e + 02	2.71e + 07
	(Rank)	(8)	(10)	(6)	(10)	(8)	(8)	(8)

(continued)

Table 4 (continued)

Functions Algorithms		F8	F9	F10	F11	F12	F13	F14
MA-SW-Chains [18]	Mean	1.41e + 07†	1.42e + 07†	2.02e + 03†	3.81e + 01†	**3.61e − 06**	1.21e + 03†	3.11e + 07†
	SD	3.62e + 07	1.12e + 06	1.42e + 02	7.31e + 00	**5.91e − 07**	5.71e + 02	1.93e + 06
	(Rank)	(5)	(3)	(4)	(7)	**(1)**	(7)	(3)
DECC-DG [11]	Mean	3.01e + 07†	5.91e + 07†	4.53e + 03†	1.01e + 01†	2.52e + 03†	4.58e + 06†	3.46e + 08†
	SD	2.11e + 07	8.12e + 06	1.42e + 02	1.02e + 00	4.85e + 02	2.18e + 06	2.46e + 07
	(Rank)	(6)	(9)	(7)	(3)	(5)	(12)	(9)
DE/best/1/bin	Mean	2.46e + 11†	3.03e + 07†	4.65e + 01†	2.35e + 03†	2.36e + 06†	3.59e + 06†	3.25e + 17†
	SD	2.22e + 10	1.00e + 06	5.11e + 01	1.39e + 03	1.59e + 06	1.09e + 05	1.59e + 15
	(Rank)	(12)	(4)	(2)	(12)	(11)	(11)	(12)
SWDE_Success	Mean	**1.35e + 05**	**2.04e + 04**	**5.15e + 00**	1.55e − 09†	5.00e + 02†	**3.00e + 02**	**8.15e + 06**
	SD	**2.39e + 04**	**1.09e + 02**	**1.11e + 01**	1.35e − 10	3.55e + 01	**2.00e + 00**	**2.15e + 04**
	(Rank)	**(1)**	**(1)**	**(1)**	(2)	(2)	**(1)**	**(1)**

Table 5 Performance of 12 large scale evolutionary optimizers including SWDE_Success on CEC 2010 benchmark suite (F15 – F20, Average Rank, $D = 1000$)

Functions Algorithms		F15	F16	F17	F18	F19	F20	Avg. Rank on all functions F1 – F20
jDElsgo [28]	Mean SD (Rank)	5.85e + 03† 4.46e + 02 (5)	1.40e + 02† 3.42e + 01 (8)	1.00e + 05† 1.25e + 04 (7)	1.86e + 03† 3.11e + 02 (3)	**2.73e + 05** **2.12e + 04** **(1)**	1.51e + 03† 1.32e + 02 (7)	4.3
DECC-ML [43]	Mean SD (Rank)	1.54e + 04† 3.51e + 02 (11)	**5.07e − 02** **2.54e − 01** **(1)**	6.56e + 06† 4.61e + 05 (11)	2.42e + 03† 1.11e + 03 (5)	1.59e + 07† 1.71e + 06 (11)	9.92e + 02† 3.55e + 01 (4)	6.95
DASA [45]	Mean SD (Rank)	1.44e + 04† 3.66e + 02 (10)	3.94e + 02† 2.12e − 01 (9)	1.04e + 04† 8.9e + 02 (4)	4.42e + 03† 2.18e + 03 (7)	8.14e + 05† 5.56e + 04 (3)	1.03e + 03† 1.49e + 02 (6)	6.7
DMS-PSO-SHS [46]	Mean SD (Rank)	4.68e + 03† 2.15e + 02 (4)	6.95e + 01† 4.25e + 00 (4)	3.23e + 03† 4.05e + 02 (3)	2.26e + 03† 1.16e + 02 (4)	1.16e + 06† 1.06e + 05 (6)	3.52e + 02† 4.02e + 01 (2)	4.45
SDENS [34]	Mean SD (Rank)	7.36e + 03† 9.63e + 01 (8)	4.03e + 02† 2.53e + 00 (10)	1.02e + 06† 1.12e + 05 (10)	3.01e + 04† 1.21e + 04 (10)	8.82e + 05† 1.52e + 05 (4)	9.93e + 02† 1.63e + 01 (3)	8.6
EOEA [44]	Mean SD (Rank)	2.14e + 03† 1.24e + 02 (2)	8.26e + 01† 1.68e + 01 (6)	7.93e + 04† 8.80e + 03 (6)	2.94e + 03† 6.92e + 02 (6)	1.84e + 06† 9.97e + 04 (8)	1.97e + 03† 2.35e + 02 (8)	5.35
DECC-G [9]	Mean SD (Rank)	1.26e + 04† 8.91e + 02 (9)	7.65e + 01† 8.15e + 00 (5)	2.85e + 05† 1.98e + 04 (9)	2.45e + 04† 1.05e + 04 (9)	1.15e + 06† 5.15e + 04 (5)	4.05e + 03† 3.66e + 02 (10)	9.15
MLCC [42]	Mean SD (Rank)	7.11e + 03† 1.31e + 03 (7)	3.72e + 02† 4.72e + 01 (11)	1.52e + 05† 1.42e + 04 (8)	7.02e + 03† 4.72e + 03 (8)	1.32e + 06† 7.32e + 04 (9)	2.02e + 03† 1.82e + 02 (9)	8.05

(continued)

Table 5 (continued)

Functions Algorithms		F15	F16	F17	F18	F19	F20	Avg. Rank on all functions F1 – F20
MA-SW-Chains [18]	Mean SD (Rank)	2.86e + 03† 1.25e + 02 (3)	9.95e + 01† 1.45e + 01 (7)	1.25e + 00≈ 1.25e − 01 (2)	1.34e + 03† 4.34e + 02 (2)	2.84e + 05† 1.69e + 04 (2)	1.05e + 03† 7.25e + 01 (5)	4.6
DECC-DG [11]	Mean SD (Rank)	5.84e + 03† 1.04e + 02 (6)	7.32e + 03† 5.72e − 14 (2)	4.06e + 04† 2.86e + 03 (5)	1.18e + 10† 2.08e + 09 (12)	1.78e + 06† 9.58e + 04 (7)	4.89e + 07† 2.28e + 07 (12)	7.75
DE/best/1/bin	Mean SD (Rank)	6.66e + 05† 5.69e + 05 (12)	2.33e + 03† 1.11e + 03 (12)	3.95e + 07† 6.69e + 06 (12)	3.52e + 07† 1.09e + 07 (11)	2.55e + 09† 1.59e + 07 (12)	3.54e + 05† 9.23e + 04 (11)	10.01
SWDE_Success	Mean SD (Rank)	**1.10e + 03** **2.09e + 01** **(1)**	8.50e − 01† 1.00e − 01 (3)	**1.13e + 00** **3.13e − 01** **(1)**	**1.01e + 03** **2.12e + 01** **(1)**	5.50e + 06† 2.00e + 05 (10)	**3.09e + 00** **1.00e − 01** **(1)**	**1.85**

Table 6 Performance of SWDE_Success, CCPSO2 and Sep-CMA-ES on 3 functions from the CEC 2008 test-suite (D = 2000)

Algorithms	SWDE_Success	CCPSO2	Sep-CMA-ES
Functions	Mean (Std. Dev.)	Mean (Std. Dev.)	Mean (Std. Dev.)
F1	1.23e − 14≈ (1.01e − 06)	1.03e − 12† (2.56e − 13)	**7.12e − 15 (6.24e − 15)**
F3	**1.00e + 01 (2.10e + 00)**	2.91e + 03† (6.43e + 02)	1.73e + 03† (7.77e + 01)
F7	**−2.20e + 04 (1.01e + 02)**	−2.28e + 04† (1.90e + 02)	−2.46e + 04† (1.57e + 02)

done in [10]. The performance is compared with two popular evolutionary large scale optimizers of diverse origins, CCPSO2 and Sep-CMA-ES [41]. The obtained results are summarized in Table 6. Sep-CMA-ES only performed better than SWDE_Success in the case of the separable function F1. However, according to the rank sum test, the difference between the final mean errors of sep-CMA-ES and SWDE_Success is not statistically meaningful. For the multimodal non-separable problems F3 and F7, SWDE_Success performed statistically better than both CCPSO2 and sep-CMA-ES. CCPSO2 took the second place for F7 and Sep-CMA-ES did the same for F3.

5 Conclusion

To address the problem of optimizing very high-dimensional numerical functions, this paper presents a new variant of DE, referred here as the SWDE_Success, which uses a simple switching scheme for the two key parameters of DE, the scale factor and the crossover rate. SWDE_Success also employs a success-based selection of either of the two kinds of mutation strategies. The algorithm uses two very common mutation strategies (the greedy DE/best/1 and the explorative DE/rand/1 schemes) and applies a simple scheme selection process, which only depends on the success of a mutation scheme in the previous iteration in terms of generating a successful offspring (one which could replace its parent during the selection).

Exploring a huge search volume (induced by the large number of variables) with a limited population of candidate solutions is challenging and it requires a judicious balance between the explorative and exploitative tendencies of an evolutionary algorithm. This requirement is nicely fulfilled by the random selection of the control parameter values from their extremities. Our results indicate that a combination of the high and unconventional value of F (= 2) with the low value (= 0.5) can be indeed very useful for solving benchmark functions. In contrast to some of the most prominent approaches (like [22, 23, 26, 27]) that sample F values from the interval of (0.4, 1) and Cr values from (0, 1), our results indicate that most of the useful information about *F* and *Cr* values can remain attached to the boundaries of their feasible regions. This point requires further analytical and experimental investigations in future.

The future works may also include a detailed study on the dynamics and search procedure of SWDE_Success, alongside an explanation of its success. Also the parameter switching strategy may be further investigated in other optimization scenarios like for moderate dimensional problems, for multi-objective, constrained and dynamic optimization problems.

References

1. Foli, K., Okabe, T., Olhofer, M., Jin, Y., Sendhoff, B.: Optimization of micro heat exchanger: CFD, analytical results and multiobjective evolutionary algorithms. Int. J. Heat Mass Transf. **49**(5–6), 1090–1099 (2006)
2. Sonoda, T., Yamaguchi, Y., Arima, T., Olhofer, M., Sendhoff, B., Schreiber, H.A.: Advanced high turning compressor airfoils for low Reynolds number condition, part I: Design and optimization. J. Turbomach. **126**(3), 350–359 (2004)
3. Wang, C., Gao, J.: High-dimensional waveform inversion with cooperative coevolutionary differential evolution algorithm. IEEE Geosci. Remote Sens. Lett. **9**(2), 297–301 (2012)
4. Das, S., Suganthan, P.N.: Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Technical Report, Jadavpur University, India and Nanyang Technological University, Singapore (2010)
5. Tang, K., Yao, X., Suganthan, P., MacNish, C., Chen, Y., Chen, C., Yang, Z.: Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. In: Nature Inspired Computat. Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep. http://nical.ustc.edu.cn/cec08ss.php (2007)
6. Tang, K., Li, X., Suganthan, P., Yang, Z., Weise, T.: Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. In: Nature Inspired Computat. Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep. http://nical.ustc.edu.cn/cec10ss.php (2009)
7. Potter, M., De Jong, K.: Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol. Comput. **8**(1), 1–29 (2000)
8. Ray, T., Yao, X.: A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning. In: Proceedings of the IEEE CEC, pp. 983–999, May 2009
9. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Inf. Sci. **178**(15), 2986–2999 (2008)
10. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans. Evol. Comput. **16**(2), 210–224 (2011)
11. Omidvar, M.N., Li, X., Mei, Y., Yao, X.: Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans. Evol. Comput. **18**(3), 378–393 (2013)
12. Krishnakumar, K.: Micro-genetic algorithms for stationary and non-stationary function optimization, SPIE 1196. Intell. Control Adapt. Syst. (1989). doi:10.1117/12.969927
13. Huang, T., Mohan, A.S.: Micro–particle swarm optimizer for solving high dimensional optimization problems. Appl. Math. Comput. **181**(2), 1148–1154 (2006)
14. Dasgupta, S., Biswas, A., Das, S., Panigrahi, B.K., Abraham, A.: A micro-bacterial foraging algorithm for high-dimensional optimization. In: IEEE Congress on Evolutionary Computation (CEC 2009), pp. 785–792, Tondheim, Norway, May 2009

15. Rajasekhar, A., Das, S., Das, S.: μABC: a micro artificial bee colony algorithm for large scale global optimization. In: Soule, T. (ed.) Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '12), pp. 1399–1400, ACM, New York, NY, USA. doi:10.1145/2330784.2330951. http://doi.acm.org/10.1145/2330784.2330951
16. Tseng, L.Y., Chen, C.: Multiple trajectory search for large scale global optimization. In: IEEE Congress on Evolutionary Computation (CEC 2008), pp. 3052–3059, Hong Kong, June 2008
17. Zhao, S.Z., Suganthan, P.N., Das, S.: Self-adaptive differential evolution with multi-trajectory search for large scale optimization. Soft. Comput. **15**, 2175–2185 (2011)
18. Molina, D., Lozano, M., Herrera, F.: MA-SW-Chains: memetic algorithm based on local search chains for large scale continuous global optimization. In: IEEE Congress on Evolutionary Computation (CEC 2010), pp. 3153–3160, Barcelona, July, 2010
19. Molina, D., Lozano, M., Sánchez, A.M., Herrera, F.: Memetic algorithms based on local search chains for large scale continuous optimization problems: MA-SSW-Chains. Soft. Comput. **15**, 2201–2220 (2011)
20. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
21. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)
22. Qin, A.K., Huang, V., Suganthan, P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**(2), 398–417 (2009)
23. Zhang, J., Sanderson, A.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)
24. Epitropakis, M., Tasoulis, D., Pavlidis, N., Plagianakos, V., Vrahatis, M.: Enhancing differential evolution utilizing proximity based mutation operators. IEEE Trans. Evol. Comput. **15**(1), 99–119 (2011)
25. Islam, S.M., Das, S., Ghosh, S., Roy, S., Suganthan, P.N.: An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans. Syst. Man Cybern. B Cybern. **42**(2), 482–500 (2012)
26. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evol. Comput. **15**(1), 55–66 (2011)
27. Mallipeddi, R., Suganthan, P.N.: Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies. In: Proc. Swarm Evol. Memet. Comput., Chennai, India, pp. 71–78 (2010)
28. Zamuda, A., Brest, J., Boˇskovi´c, B., Zumer, V.: Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. In: IEEE Congress on Evolutionary Computation (CEC 2008), pp. 3718–3725, Hong Kong, June 2008
29. Parsopoulos, K.E.: Cooperative micro-differential evolution for high-dimensional problems. In: Genetic and Evolutionary Computation Conference 2009 (GECCO 2009), pp. 531–538, Montreal, Canada (2009)
30. Zhao, S.Z., Suganthan, P.N., Das, S.: Self-adaptive differential evolution with multi-trajectory search for large scale optimization. Soft. Comput. **15**, 2175–2185 (2011)
31. Brest, J., Maučec, M.S.: Self-adaptive differential evolution algorithm using population size reduction and three strategies. Soft. Comput. **15**(11), 2157–2174 (2011)
32. Wang, H., Wu, Z., Rahnamayan, S.: Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. Soft. Comput. **15**(11), 2127–2140 (2011)
33. Weber, M., Neri, F., Tirronen, V.: Shuffle or update parallel differential evolution for large-scale optimization. Soft. Comput. **15**(11), 2089–2107 (2011)
34. Wang, H., Wu, Z., Rahnamayan, S., Jiang, D.: Sequential DE enhanced by neighborhood search for large scale global optimization. In: IEEE Congress on Evolutionary Computation (CEC 2010), pp. 4056–4062, Barcelona, July, 2010
35. Zaharie, D.: Influence of crossover on the behavior of the differential evolution algorithm. Appl. Soft Comput. **9**(3), 1126–1138 (2009)

36. Zaharie, D.: Critical values for the control parameters of differential evolution algorithms. In: Proc. 8th Int. Mendel Conf. Soft. Comput., pp. 62–67 (2002)
37. Ronkkonen, J., Kukkonen, S., Price, K.V.: Real parameter optimization with differential evolution. In: The 2005 IEEE Congress on Evolutionary Computation (CEC2005), vol. 1, pp. 506–513. IEEE Press (2005)
38. Hu, J., Zeng, J., Tan, Y.: A diversity-guided particle swarm optimizer for dynamic environments. In: Proceedings of Bio-Inspired Computational Intelligence Applivations, vol. 9, no. 3, pp. 239–247. Lecture Notes in Computer Science (2007)
39. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. **1**(1), 3–18 (2011)
40. Hsieh, S.T., Sun, T.Y., Liu, C.C., Tsai, S.J.: Efficient population utilization strategy for particle swarm optimizer. IEEE Trans. Syst. Man Cybern. B Cybern. **39**(2), 444–456 (2009)
41. Ros, R., Hansen, N.: A simple modification in CMA-ES achieving linear time and space complexity. Lect. Notes Comput. Sci. **5199**, 296–305 (2008)
42. Yang, Z., Tang, K., Yao, X.: Multilevel cooperative coevolution for large scale optimization. In: Proc. IEEE Congr. Evol. Comput., pp. 1663–1670, June 2008
43. Omidvar, M.N., Li, X., Yao, X.: Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: Proc. IEEE Congr. Evol. Comput., pp. 1762–1769, July 2010
44. Wang, Y., Huang, J., Dong, W.S., Yan, J.C., Tian, C.H., Li, M., Mo, W.T.: Two-stage based ensemble optimization framework for large-scale global optimization. Eur. J. Oper. Res. **228**, 308–320 (2013)
45. Korošec, P., Šilc, J.: The differential ant-stigmergy algorithm for large scale real-parameter optimization. In: Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, vol. 5217, pp. 413–414, Springer, Berlin Heidelberg (2008)
46. Zhao, S., Liang, J., Suganthan, P.N., Tasgetiren, M.F.: Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 3845–3852 (2008)

The Initial Study on the Potential of Super-Sized Swarm in PSO

Michal Pluhacek, Roman Senkerik and Ivan Zelinka

Abstract In this initial study it is addressed the issue of population size for the PSO algorithm. For many years now it is understood that population size of several dozens is sufficient for the vast majority of optimization tasks. With strict limitation of cost function evaluations (CFEs) the setting is typically limited to adjusting the number of iterations of the algorithm. In this study it is investigated the possibility of using population of thousands of particles and its effect on the performance of the algorithm in limited CFEs. It is also proposed an alternative setting of acceleration constants in order to improve the performance of the PSO with super-sized population. The performance of the proposed method is tested on IEEE CEC 2013 benchmark set and compared with original PSO design and state of art methods.

Keywords Particle swarm optimization · PSO · Population size

1 Introduction

As one of the most prominent representatives of evolutionary computational techniques (ECTs) the Particle swarm optimization algorithm (PSO) [1–4] is still, years after its introduction, studied, modified and applied by many researchers. The well-known issue of premature convergence into locals [1, 2, 5] has been one of the

M. Pluhacek (✉) · R. Senkerik
Faculty of Applied Informatics, Tomas Bata University in Zlin,
Nam T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
e-mail: pluhacek@fai.utb.cz

R. Senkerik
e-mail: senkerik@fai.utb.cz

I. Zelinka
Faculty of Electrical Engineering and Computer Science,
Technical University of Ostrava, 17. Listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
e-mail: ivan.zelinka@vsb.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_10

biggest challenges for PSO researchers. Many different approaches were recently developed to improve the performance of PSO such as the multi-swarms [6], comprehensive learning [7] or orthogonal learning [8]. Despite the increasing complexity of the latest PSO modifications this study intends to prove that there is still potential for simple but effective alternations of the original design. The issue of population size and structure [9] has been studied previously but in these days the complexity of optimization tasks has significantly increased and the performance of the PSO with alternative population size setting may prove very promising when dealing with the most complex problems. The inspiration came from nature where insect swarm can easily have thousands of individuals.

The rest of the paper is structured as follows: In the following section the PSO algorithm and the motivation for the alternative population size setting is explained. The experiment is designed in Sect. 3 and results presented in Sect. 4. The results are discussed in Sect. 5 before the final conclusion.

2 Particle Swarm Optimization Algorithm and Population Size

Original PSO takes the inspiration from behavior of fish and birds. The knowledge of global best found solution (typically noted *gBest*) is shared among the particles in the swarm. Furthermore each particle has the knowledge of its own (personal) best found solution (noted *pBest*). Last important part of the algorithm is the velocity of each particle that is taken into account during the calculation of the particle movement. The new position of each particle is then given by (1), where x_i^{t+1} is the new particle position; x_i^t refers to current particle position and v_i^{t+1} is the new velocity of the particle.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{1}$$

To calculate the new velocity the distance from pBest and gBest is taken into account alongside with current velocity (2).

$$v_{ij}^{t+1} = w \cdot v_{ij}^t + c_1 \cdot Rand \cdot (pBest_{ij} - x_{ij}^t) + c_2 \cdot Rand \cdot (gBest_j - x_{ij}^t) \tag{2}$$

Where:

v_{ij}^{t+1} - New velocity of the ith particle in iteration $t + 1$. (component j of the dimension D).

w - Inertia weight value.

v_{ij}^t - Current velocity of the ith particle in iteration t. (component j of the dimension D).

c_1, $c_2 = 2$ - Acceleration constants.

$pBest_{ij}$ - Local (personal) best solution found by the ith particle. (component j of the dimension D).
$gBest_j$ - Best solution found in a population. (component j of the dimension D).
x_{ij}^t - Current position of the ith particle (component j of the dimension D) in iteration t.
Rand - Pseudo random number, interval (0, 1).

Finally the linear decreasing inertia weight [2, 3] is used. The dynamic inertia weight is meant to slow the particles over time thus to improve the local search capability in the later phase of the optimization. The inertia weight has two control parameters w_{start} and w_{end}. A new w for each iteration is given by (3), where t stands for current iteration number and n stands for the total number of iterations. The typical values used in this study were $w_{start} = 0.9$ and $w_{end} = 0.4$.

$$w = w_{start} - \frac{((w_{start} - w_{end}) \cdot t)}{n} \tag{3}$$

The population size is typically set to 30–60 for PSO [5–9]. In this work an experiment is carried out to investigate the impact of significant increase of the population size on the performance of the original PSO. As the limitation of maximum number of cost function evaluations (CFEs) is constant the increasing of population size leads to significant decrease of the number of iterations of the algorithm. The presumption in this work is that the super-sized swarms may be capable to outperform the PSO with typical population size settings on particularly complex optimization tasks. During initial experiments it was found out that the issue of premature convergence that is typical for PSO cannot be solved by simple addition of particles and decreasing the number of iterations. The performance of above mentioned approach is presented in this paper. However with the significantly increased population size it is possible to alter one of the fundamental principles of PSO and set both acceleration constants to the value of 1. By doing so, the performance of the algorithm can be significantly improved on complex test functions as is presented in the following sections.

3 Experiment Setup

In this initial study the performance of the newly proposed method was tested on the full IEEE CEC 2013 benchmark set [10] for $dim = 10$. According to the benchmark rules 51 separate runs were performed for each algorithm and the maximum number of cost function evaluations (CFEs) was set to 100000. The population size (NP) for the alternative setting was set to 2000 based on tuning experiment.

For different combinations of setting were tested and noted PSO 1, PSO 2, PSO 3 and PSO 4.

For PSO 1:
c_1,c_2 = 2; NP = 50; Iterations = 2000;

For PSO 2:
c_1,c_2 = 2; NP = 2000; Iterations = 50;

For PSO 3:
c_1,c_2 = 1; NP = 50; Iterations = 2000;

For PSO 4:
c_1,c_2 = 1; NP = 2000; Iterations = 50;

Other controlling parameters of the PSO and DE were set to typical values as follows:

$w_{start} = 0.9$;

$w_{end} = 0.4$;

$v_{max} = 0.2$;

4 Results

The mean results of all previously described PSO variants are presented in the following Table 1. The best results are given in bold numbers.

Furthermore as an example the mean history of the best found solution during the optimization in given in Figs. 1, 2 and 3.

In Table 2 the performance of PSO 4 is compared to the state of art methods based on PSO: the self-adaptive heterogeneous PSO for real-parameter optimization [11] noted fk-PSO and the Particle Swarm Optimization and Artificial Bee Colony Hybrid algorithm noted ABS-SPSO [12].

5 Results Discussion

The main focus of this study was the performance of proposed alternative setting of the PSO on the most complex benchmark functions (noted with ^{c} in the tables).

Initially the PSO with population of 2000 particles (PSO 2) did not outperform the original design and therefore proved that the simple act of increasing the size of the population at the cost of less iteration is not beneficial for the optimization process. However when the acceleration constants values were altered also the PSO with population size 2000 significantly improved its performance and outperformed all other settings on complex functions.

The performance of the PSO with super-sized population and alternative setting of acceleration constants (noted PSO 4) was significantly improved on the majority of the complex problems in the benchmark set (see Table 1). In Addition the performance on the unimodal yet non-primitive functions f_2 and f_3 is also very

Table 1 Mean results comparison, *dim* = 10, max. CFE = 100000

Function	f_{min}	PSO 1	PSO 2	PSO 3	PSO 4
f_1^u	−1400	−1.40E + 03	−1.40E + 03	−1.40E + 03	−1.40E + 03
f_2^u	−1300	2.45E + 05	1.35E + 06	7.24E + 05	**1.53E + 05**
f_3^u	−1200	1.86E + 06	2.99E + 07	1.72E + 07	**1.76E + 06**
f_4^u	−1100	**−5.20E + 02**	1.14E + 03	1.58E + 03	−1.94E + 02
f_5^u	−1000	−1.00E + 03	−9.99E + 02	−1.00E + 03	−1.00E + 03
f_6^m	−900	**−8.94E + 02**	−8.86E + 02	−8.79E + 02	−8.92E + 02
f_7^m	−800	**−7.96E + 02**	−7.87E + 02	−7.83E + 02	−7.94E + 02
f_8^m	−700	−6.80E + 02	–-6.80E + 02	−6.80E + 02	−6.80E + 02
f_9^m	−600	−5.97E + 02	−5.95E + 02	−5.96E + 02	−5.97E + 02
f_{10}^m	−500	−5.00E + 02	−4.98E + 02	−4.99E + 02	−5.00E + 02
f_{11}^m	−400	**−3.98E + 02**	−3.79E + 02	−3.96E + 02	−3.94E + 02
f_{12}^m	−300	−2.87E + 02	−2.72E + 02	−2.84E + 02	**−2.88E + 02**
f_{13}^m	−200	−1.80E + 02	−1.71E + 02	−1.73E + 02	**−1.83E + 02**
f_{14}^m	−100	**5.72E + 01**	7.45E + 02	1.43E + 02	2.23E + 02
f_{15}^m	100	8.45E + 02	1.34E + 03	7.92E + 02	**7.58E + 02**
f_{16}^m	200	2.01E + 02	2.01E + 02	2.01E + 02	2.01E + 02
f_{17}^m	300	**3.14E + 02**	3.43E + 02	3.16E + 02	3.20E + 02
f_{18}^m	400	4.32E + 02	4.44E + 02	**4.22E + 02**	4.23E + 02
f_{19}^m	500	5.01E + 02	5.03E + 02	5.01E + 02	5.01E + 02
f_{20}^m	600	6.03E + 02	6.03E + 02	6.03E + 02	**6.02E + 02**
f_{21}^c	700	1.08E + 03	**1.07E + 03**	1.09E + 03	1.10E + 03
f_{22}^c	800	**9.71E + 02**	1.72E + 03	1.12E + 03	1.19E + 03
f_{23}^c	900	1.81E + 03	2.46E + 03	2.00E + 03	**1.79E + 03**
f_{24}^c	1000	1.21E + 03	1.21E + 03	1.22E + 03	**1.20E + 03**
f_{25}^c	1100	1.30E + 03	1.30E + 03	1.31E + 03	**1.29E + 03**
f_{26}^c	1200	1.36E + 03	1.40E + 03	**1.35E + 03**	1.38E + 03
f_{27}^c	1300	1.67E + 03	1.73E + 03	1.73E + 03	**1.63E + 03**
f_{28}^c	1400	1.69E + 03	1.70E + 03	1.76E + 03	**1.66E + 03**

encouraging. The differences in convergence curves can be clearly observed from Figs. 1, 2 and 3. It is clear that in the selected cases (complex functions) the proposed alternative settings achieves not only better final results but also faster convergence speed.

Furthermore when compared with the more complex state of art algorithms this simple alternation of the original PSO presented very promising performance on the complex functions (Table 2).

These results clearly support the idea that super-sized swarms may prove effective for complex tasks and may bring new view into the discussion of optimal population size for swarm based ECTs.

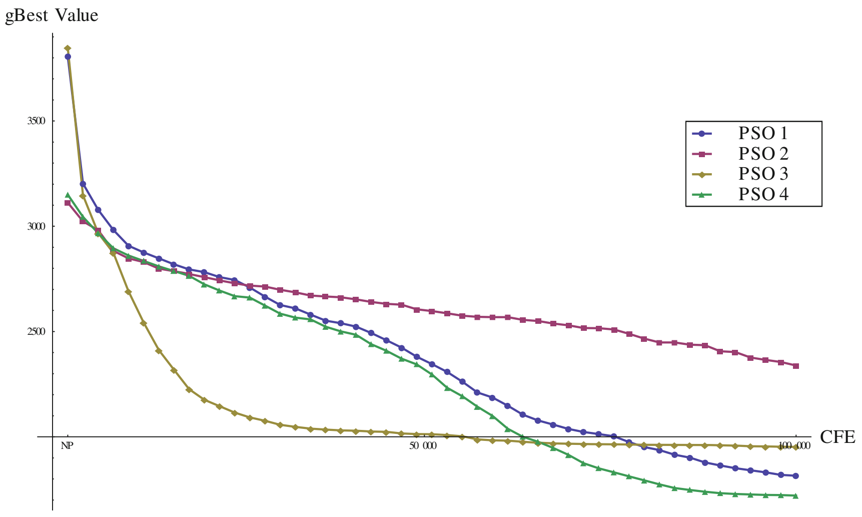


Fig. 1 Mean best value history comparison 51 runs – f_{23}

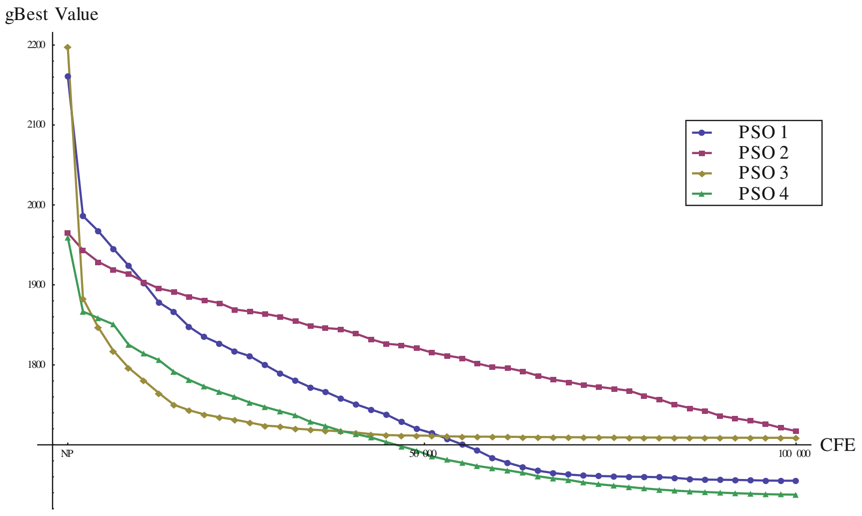


Fig. 2 Mean best value history comparison 51 runs – f_{27}

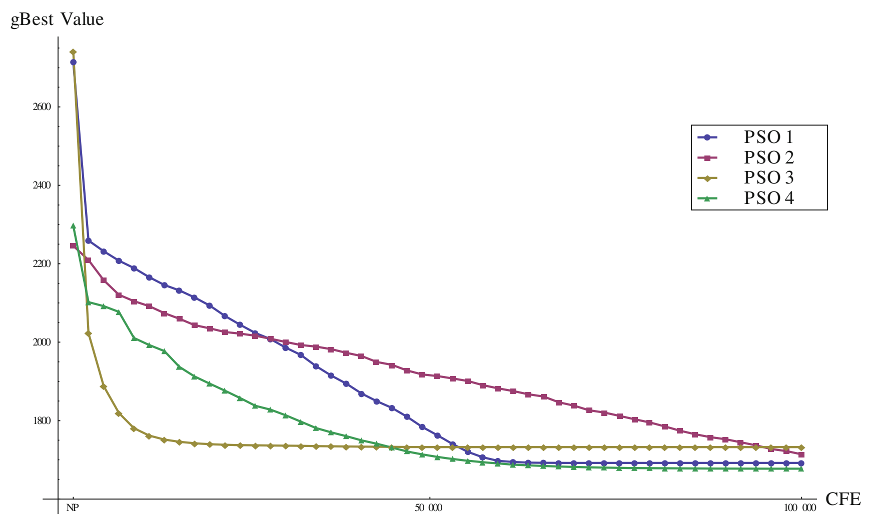


Fig. 3 Mean best value history comparison 51 runs – f_{28}

Table 2 Mean results comparison, *dim* = 10, max. CFE = 100000

Function	f_{min}	PSO 4	fk-PSO	ABS-SPSO
f_1^u	−1400	−1.40E + 03	−1.40E + 03	−1.40E + 03
f_2^u	−1300	1.53E + 05	**1.43E + 05**	1.48E + 05
f_3^u	−1200	1.76E + 06	6.74E + 05	**1.27E + 05**
f_4^u	−1100	−1.94E + 02	**−6.84E + 02**	1.30E + 03
f_5^u	−1000	−1.00E + 03	−1.00E + 03	−1.00E + 03
f_6^m	−900	−8.92E + 02	**−8.97E + 02**	−8.95E + 02
f_7^m	−800	−7.94E + 02	−7.98E + 02	**−8.00E + 02**
f_8^m	−700	−6.80E + 02	−6.80E + 02	−6.80E + 02
f_9^m	−600	−5.97E + 02	−5.97E + 02	−5.96E + 02
f_{10}^m	−500	−5.00E + 02	−4.99E + 02	−5.00E + 02
f_{11}^m	−400	−3.94E + 02	−4.00E + 02	−4.00E + 02
f_{12}^m	−300	−2.88E + 02	−2.93E + 02	**−2.94E + 02**
f_{13}^m	−200	−1.83E + 02	−1.89E + 02	**−1.94E + 02**
f_{14}^m	−100	2.23E + 02	−6.22E + 01	**−9.96E + 01**
f_{15}^m	100	7.58E + 02	**5.54E + 02**	5.96E + 02
f_{16}^m	200	2.01E + 02	2.00E + 02	2.00E + 02
f_{17}^m	300	3.20E + 02	3.11E + 02	**3.10E + 02**
f_{18}^m	400	4.23E + 02	**4.16E + 02**	4.17E + 02
f_{19}^m	500	5.01E + 02	5.01E + 02	**5.00E + 02**
f_{20}^m	600	6.02E + 02	6.03E + 02	6.02E + 02
f_{21}^c	700	1.10E + 03	**1.08E + 03**	1.10E + 03

(continued)

Table 2 (continued)

Function	f_{min}	PSO 4	fk-PSO	ABS-SPSO
f_{22}^{c}	800	1.19E + 03	9.22E + 02	**8.13E + 02**
f_{23}^{c}	900	1.79E + 03	**1.42E + 03**	1.50E + 03
f_{24}^{c}	1000	1.20E + 03	1.20E + 03	1.20E + 03
f_{25}^{c}	1100	**1.29E + 03**	1.31E + 03	1.30E + 03
f_{26}^{c}	1200	1.38E + 03	1.39E + 03	**1.33E + 03**
f_{27}^{c}	1300	**1.63E + 03**	1.67E + 03	1.65E + 03
f_{28}^{c}	1400	**1.66E + 03**	1.73E + 03	1.69E + 03

6 Conclusion

In this paper the idea of super-sized population for PSO was presented. The inspiration for this approach came from nature where insect swarm can easily have thousands of individuals. The population size was set to 2000 and the performance of the algorithm was tested on the CEC´13 benchmark set.

Furthermore the acceleration constants were changed to value of 1 and by doing so the performance of PSO algorithm with linear decreasing inertia weight and saturation of max. velocity was significantly improved on the majority of complex benchmark functions when compared with the original design. Furthermore the comparison of this very simple method with significantly more complex state-of-art methods brought very encouraging results.

It is necessary to support this initial and small scale study with further and more extensive studies. This will be the main objective of future research. The usefulness of this method for other evolutionary computational techniques will also be investigated.

Acknowledgments This work was supported by Grant Agency of the Czech Republic - GACR P103/15/06700S, further by financial support of research project NPU I No. MSMT-7778/2014 by the Ministry of Education of the Czech Republic. also by the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089, partially supported by Grant of SGS No. SP2015/142, VŠB - Technical University of Ostrava, Czech Republic and by Internal Grant Agency of Tomas Bata University under the project No. IGA/FAI/2015/057.

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
2. Yuhui, S., Eberhart, R.: A modified particle swarm optimizer. In: IEEE World Congress on Computational Intelligence, pp. 69–73, 4–9 May 1998
3. Nickabadi, A., Ebadzadeh, M.M., Safabakhsh, R.: A novel particle swarm optimization algorithm with adaptive inertia weight. Appl. Soft Comput. **11**(4), 3658–3670 (2011)

4. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers (2001)
5. van den Bergh, F., Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Inf. Sci. **176**(8), 937–971 (2006)
6. Liang, J., Suganthan, P.N.: Dynamic multi-swarm particle swarm optimizer. In: Swarm Intelligence Symposium, SIS 2005, pp. 124–129 (2005)
7. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. **10**(3), 281–295 (2006)
8. Zhi-Hui, Z., Jun, Z., Yun, L., Yu-hui, S.: Orthogonal learning particle swarm optimization. IEEE Trans. Evol. Comput. **15**(6), 832–847 (2011)
9. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC '02, 2002, pp. 1671–1676 (2002)
10. Liang, J.J., Qu, B.-Y., Suganthan, P.N., Hernández-Díaz, A.G.: Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization, Technical Report 201212. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore (2013)
11. Nepomuceno F., Engelbrecht A.: A self-adaptive heterogeneous pso for real-parameter optimization. In: Proceedings of the IEEE International Conference on Evolutionary Computation (2013)
12. El-Abd M.: Testing a Particle Swarm Optimization and Artificial Bee Colony Hybrid algorithm on the CEC13 benchmarks. In: IEEE Congress on Evolutionary Computation, pp. 2215–2220 (2013)

A Levy Interior Search Algorithm for Chaotic System Identification

Rushi Jariwala, Rohan Patidar and Nithin V. George

Abstract In this paper, an improved interior search algorithm (ISA) is designed by incorporating *Lêvy* flight for solving optimisation problems. *Lêvy* flight pattern seen in some birds, is a special type of movement along a straight line followed by sudden turns in random directions. The convergence rate of ISA is improved using the principles of *Lêvy* flight in the proposed levy interior search algorithm (LISA). LISA is validated against a set of benchmark optimisation problems to demonstrate its performance. Further, LISA is used for parameter identification of an integer order Rossler's chaotic system. Simulation results show that LISA outperforms other well-known existing optimisation algorithms like particle swarm optimisation (PSO), ISA and cuckoo search algorithm (CSA).

Keywords Interior search algorithm · Particle swarm optimisation · Cuckoo search algorithm · Chaotic systems

1 Introduction

Many popular deterministic optimisation techniques use a gradient descent approach and such methods can suffer from local optima problems when applied to solve multi-modal optimisation tasks. Meta-heuristic algorithms have been designed to solve this limitation of traditional deterministic optimisation schemes. Some of the most popular meta-heuristic algorithms include the genetic algorithm (GA) [6], particle swarm optimisation (PSO) [8], differential evolution (DE) [10], ant colony optimisation (ACO) [2] and artificial bee colony (ABC) optimisation algorithm [7].

R. Jariwala · R. Patidar · N.V. George (✉)
Department of Electrical Engineering, Indian Institute of Technology Gandhinagar, Ahmedabad 382424, Gujarat, India
e-mail: nithin@iitgn.ac.in

R. Patidar
e-mail: rohanpatidar@iitgn.ac.in

R. Jariwala
e-mail: rushi.jariwala@iitgn.ac.in

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_11

Even though most of the popular meta-heuristic algorithms are bio-inspired, there are a few other famous meta-heuristic algorithms which are inspired by other phenomena. For example, harmony search algorithm (HSA) is a meta-heuristic algorithm based on the way musicians create a piece of music [5]. Internal search algorithm (ISA) is a recent meta-heuristic algorithm, which is inspired by the manner in which a designer organizes objects in a room to enhance the overall aesthetics [4]. ISA has been successfully applied for engineering optimisation in [3].

Cuckoo search algorithm (CSA) is a recently developed evolutionary meta-heuristic algorithm, which has gained significant attention of the optimisation community [12]. CSA is inspired by the breeding behaviour of a species of cuckoo and has been successfully applied in solving several classes of optimisation problems in engineering and technology [1, 9, 11]. The effectiveness of CSA is solving global optimisation problems may be attributed to the concept of *Lêvy* flights, which are random motion that can been seen in some species of birds. With an objective to enhance the optimisation capability of ISA, we have designed a new meta-heuristic optimisation algorithm namely the *Lêvy* interior search algorithm (LISA) by incorporating some of the concepts of *Lêvy* flights into ISA. We have also designed a variable selection parameter to further enhance the convergence characteristics of LISA.

The rest of the paper is organized as follows. The *Lêvy* interior search algorithm is introduced in Sect. 2. The effectiveness of the new algorithm is tested in Sect. 3 by using the algorithm in solving some benchmark optimisation tasks. The proposed algorithm has also been employed for identifying unknown parameters in a chaotic system in Sect. 3 and the concluding remarks are made in Sect. 4.

2 Levy Interior Search Algorithm

ISA is an evolutionary computing algorithm, which has been proposed recently. ISA has been inspired from the design principles of a room, especially from the aesthetics point of view, in which the architect tries to replicate the best features of a room. An attempt has been made in this section to improve the optimisation capability of ISA by incorporating the principle of *Lêvy* flights. *Lêvy* flights are a flight pattern seen in birds and the proposed algorithm is called the *Lêvy* interior search algorithm (LISA). The basic steps involved in LISA are summarised as follows:

Step 1 Define the search space and randomly distribute n elements, x_i for $i = 1, 2, 3, \dots, n$ within the search space.

Step 2 Evaluate the fitness f_i for all the elements. Identify the element with the best fitness, which is referred to as x_{global}.

Step 3 Randomly cluster the elements into two groups, the composition and mirror groups. An element x_i will be added to the mirror group (MG) if

$$rand_i \leq \alpha \tag{1}$$

and will be added to composition group (CG) otherwise. In (1), α (with $0 \le \alpha \le 1$) is a selection parameter, which has been considered as a fixed value in a conventional ISA [4]. It may be noted that having more elements in CG improves the global search capability. In an endeavour to improve the convergence speed, we have used a dynamic α, which varies linearly upto 80 % of the iterations and is kept as a constant close to 1 after that. Keeping α near 1 towards the later stages of the iterations result in a larger mirror group (which can help in local search). Overall a variable α leads to faster convergence.

Step 4 An image of an MG element x_i is created at the position x_i^{new} by placing a mirror at x_{mirror}. The mirror is placed perpendicular to a line joining the element x_i of MG and x_{global}. The location of the mirror is given by

$$x_{mirror} = rand * x_i + (1 - rand) * x_{global} \tag{2}$$

and the updated position for the mirror element is given by

$$x_i^{new} = 2 * x_{mirror} - x_i. \tag{3}$$

This is analogous to placing a mirror near the best object in the room to emphasize its beauty and to make the view more attractive.

Step 5 The position of all elements in CG are updated to new positions

$$x_i^{new} = L_{CG} + (U_{CG} - L_{CG}) * rand \tag{4}$$

where L_{CG} and U_{CG} are the lower and upper bounds of the elements in CG during the previous generation. The position update in (4) will be discarded if the fitness at x_i^{new} is worse than that at the original position x_i. This concept is similar to a situation in which the designer move objects within a room to enhance the room's beauty and the objects will be placed at their new position only if the beauty of the room is enhanced by the change.

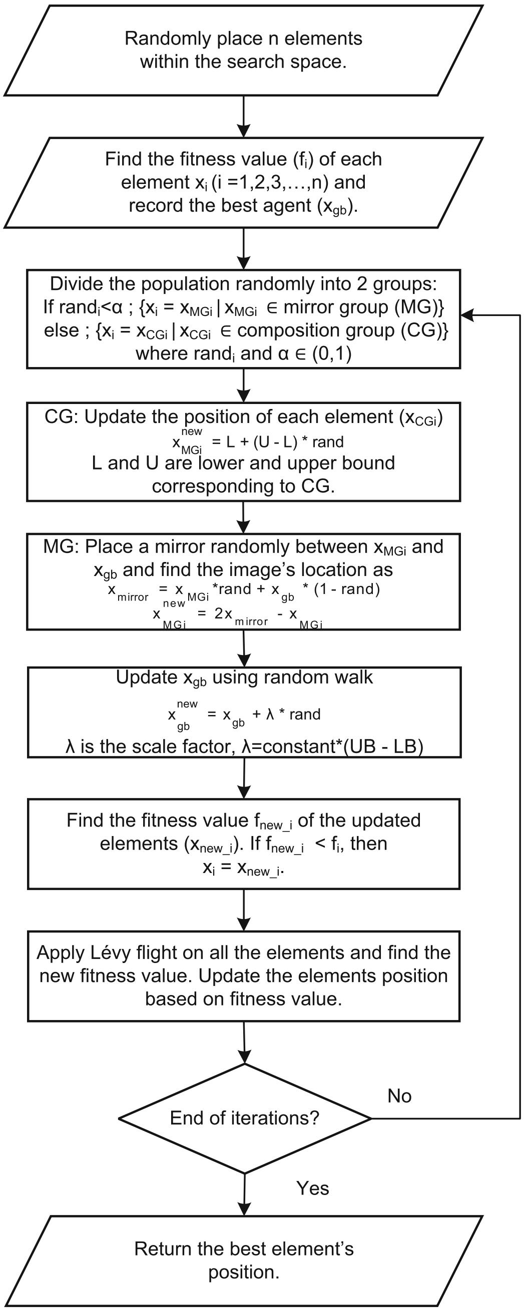
Step 6 The global best element is updated as

$$x_{global}^{new} = x_{global} + r_n * \lambda \tag{5}$$

where r_n is a normally distributed random vector and λ is a scale factor, which is dependent on the search area. The updated global best element is retained only if the fitness at the new position is better than that at the earlier position.

Step 7 An attempt is made to update the positions of all elements through a *Lévy* manoeuvre which is a random motion seen in birds. The position which is giving better fitness among the updated and original position is kept as the final position of the element in this step. *Lévy* flight is performed on all elements to obtain new positions

Fig. 1 Flowchart for LISA


Randomly place n elements within the search space.
Find the fitness value (f_i) of each element x_i ($i = 1,2,3,\ldots,n$) and record the best agent (x_{gb}).
Divide the population randomly into 2 groups: If $rand_i < \alpha$; $\{x_i = x_{MGi} \mid x_{MGi} \in$ mirror group (MG)$\}$ else ; $\{x_i = x_{CGi} \mid x_{CGi} \in$ composition group (CG)$\}$ where $rand_i$ and $\alpha \in (0,1)$
CG: Update the position of each element (x_{CGi}) $x_{MGi}^{new} = L + (U - L) * rand$ L and U are lower and upper bound corresponding to CG.
MG: Place a mirror randomly between x_{MGi} and x_{gb} and find the image's location as $x_{mirror} = x_{MGi} * rand + x_{gb} * (1 - rand)$ $x_{MGi}^{new} = 2x_{mirror} - x_{MGi}$
Update x_{gb} using random walk $x_{gb}^{new} = x_{gb} + \lambda * rand$ λ is the scale factor, λ=constant*(UB - LB)
Find the fitness value f_{new_i} of the updated elements (x_{new_i}). If $f_{new\_i} < f_i$, then $x_i = x_{new\_i}$.
Apply Lévy flight on all the elements and find the new fitness value. Update the elements position based on fitness value.
End of iterations?
No
Yes
Return the best element's position.

$$x_i^{new} = x_i + \mu \oplus Lévy(\beta), \tag{6}$$

where μ is the step size, $\oplus$ represents the entry wise multiplication operation and β is a *Lévy* flight parameter $(1 < \beta \leq 3)$.

Step 8 Combine the MG and CG into a single group of all elements.

Step 9 Repeat steps 2 to 8 until stopping criteria is met and the return the best element as the solution.

Figure 2 shows the schematic diagram of the different types of agent motion (for one iteration) which happens in the proposed LISA and the flowchart of the algorithm is given in Fig. 1.

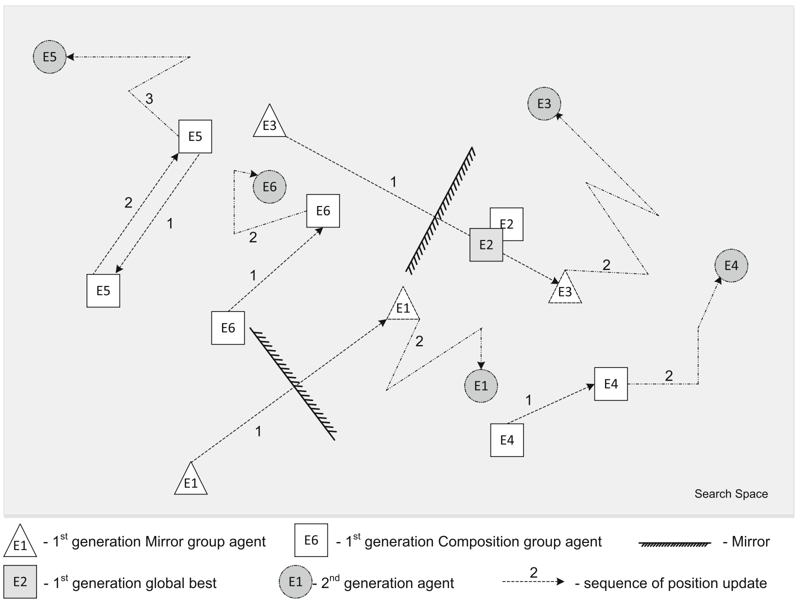

E5
3
E5
2
1
E5
E3
E6
E6
2
1
E6
1
E2
E2
E3
E3
2
E4
E1
2
E1
1
E4
2
1
E4
E1
Search Space
E1 - 1st generation Mirror group agent
E6 - 1st generation Composition group agent
- Mirror
E2 - 1st generation global best
E1 - 2nd generation agent
2 - sequence of position update

Fig. 2 Schematic diagram of agent movement in the search space for two generations of LISA

3 Simulation Study

The effectiveness of the proposed LISA in solving optimisation problems is evaluated in this section. This section is divided into two parts. In the first section, the feasibility of using proposed LISA is validated against a set of benchmark test functions. The results obtained are compared with that of a classical evolutionary algorithm like PSO. It is also compared with the results obtained using ISA and CSA.

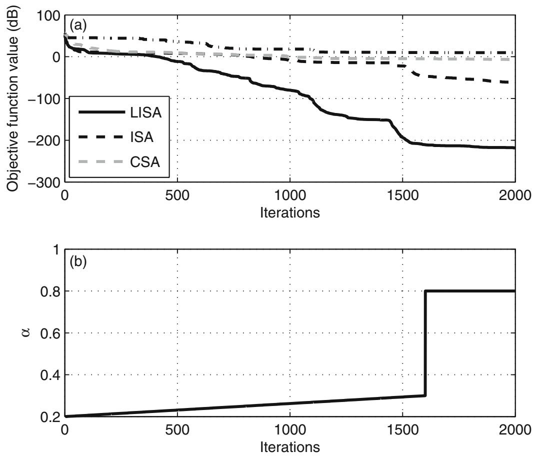


Fig. 3 (a) Comparison of convergence characteristics for computing the minimum value of a Rosenbrock function. (b) Variation of α with respect to iterations

In the second section, LISA is used for parameter identification of Rossler's system, which is a popular chaotic system.

3.1 Case A: Benchmark Test Functions

LISA has been employed for optimisation of a few benchmark problems. The task is to determine the minimum value of the test function (f_i) and we have considered four multimodal and four unimodal test functions for this study. The definition of the test functions, the range of the search space as well as the actual minimum value of each of test function is given in Table 1. For all the algorithms considered in this study, the population size has been taken as 100 and 50 for multimodal and unimodal functions respectively.

The various parameters used in this case are given below. The scale factor λ for ISA as well as LISA has been taken as $1 \times 10^{-9} \times (U - L)$, where U and L are the upper and lower bounds of the search space. For LISA, the value of the parameter α has been set to increase linearly from 0.2 to 0.3 for the first 80 % of the generations and thereafter α has been kept constant at 0.8. However in ISA, α is kept constant at 0.25. The *Lévy* flight step size of LISA has been considered as 0.1. For CSA, the step size is taken as 0.1 and $p_a = 0.25$. We have employed a standard PSO algorithm with the acceleration parameters c_1 and c_2 taken as 2 the inertial weight μ is linearly varied from 0.9 to 0.2 for improved convergence.

The mean and standard deviation of the error, for each of the algorithm studied, in reaching the global minimum value has been compared in Table 2. These values are an average of 100 independent experiments and have been measured at the end of 2000 iterations for multimodal functions and 50 iterations for unimodal functions. The improved capability of LISA in finding the optimal solutions, in com-

Table 1 Definition of the benchmark test functions

f_i	Function name	Dimension (d)	Function	Function minima	Range
f_1	Ackley	128	$f(x) = -20exp\big(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\big) - exp\big(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)\big) + a + exp(1)$	0	$[-32.768, 32.768]$
f_2	Zakharov	30	$f(x) = \sum_{i=1}^{d} x_i^2 + (\sum_{i=1}^{d} 0.5ix_i)^2 + (\sum_{i=1}^{d} 0.5ix_i)^4$	0	$[-5, 10]$
f_3	Rosenbrock	16	$f(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	0	$[-5, 10]$
f_4	De Jong	256	$f(x) = \sum_{i=1}^{d} x_i^2$	0	$[-5.12, 5.12]$
f_5	Shubert	2	$f(x) = \big(\sum_{i=1}^{5} icos((i+1)x_1 + i)\big)\big(\sum_{i=1}^{5} icos((i+1)x_2 + i)\big)$	-186.7309	$[-5.12, 5.12]$
f_6	Easom	2	$f(x) = -cos(x_1)cos(x_2)exp\big(-(x_1 - \pi)^2 - (x_2 - \pi)^2\big)$	-1	$[-100, 100]$
f_7	Michalewicz	2	$f(x) = -\sum_{i=1}^{d} sin(x_i)\big(sin^{20}(\frac{ix_i^2}{\pi})\big)$	-1.8013	$[0, 3.14]$
f_8	Beale	2	$f(x) = (1.5 - x_1 - x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	0	$[-4.5, 4.5]$

Table 2 Comparison of optimisation performance for benchmark problems solved using PSO, CSA, ISA and LISA

f_i		PSO	ISA	CSA	LISA
f_1	*Best*	2.1553	9.5345	$2.4135e-02$	**4.5564e – 13**
	Worst	5.3264	$7.1114e-1$	$1.9500e-01$	**7.1632e – 12**
	Mean	3.2861	7.9346	0.1408	**1.9817e – 12**
	Std.dev.	0.5108	0.7111	0.0241	**1.1037e – 12**
f_2	*Best*	0.4811	$4.4300e-02$	$1.4977e-11$	**0**
	Worst	27.3124	$1.3588e-10$	2.5473	**1.4142e – 18**
	Mean	6.6932	$2.7976e-12$	1.5343	**3.0143e – 20**
	Std.dev.	4.6694	$1.6499e-11$	0.3533	**1.4631e – 19**
f_3	*Best*	$2.7526e-04$	$6.5501e-11$	$8.1103e-03$	**2.2212e – 20**
	Worst	$6.6588e+01$	3.9867	$5.7964e-01$	**3.6953e – 11**
	Mean	6.5237	0.2392	0.1213	**8.8744e – 13**
	Std.dev.	7.2004	0.9515	0.1084	**4.0998e – 12**
f_4	*Best*	5.2929	1.4078	$2.1280e-01$	**2.1966e – 18**
	Worst	20.0427	7.6475	$4.0170e-01$	**6.3264e – 17**
	Mean	9.9611	3.7332	0.3117	**1.5660e – 17**
	Std.dev.	2.8848	1.2184	0.0416	**1.1083e – 17**
f_5	*Best*	$5.4898e-08$	$8.5265e-14$	$1.1591e-04$	**0**
	Worst	2.8961	$3.5650e-04$	$3.6777e-01$	**7.7051e – 11**
	Mean	0.0635	$3.9330e-06$	0.0427	**1.5054e – 12**
	Std.dev.	0.3175	$3.5689e-05$	0.0589	**8.5783e – 12**
f_6	*Best*	$7.2423e-08$	$3.9968e-15$	$1.3718e-03$	**0**
	Worst	0.8936	1.0000	1.0000	**1.0264e – 07**
	Mean	0.0315	0.0936	0.4837	**3.0303e – 09**
	Std.dev.	0.1262	0.2769	0.3796	**1.5672e – 08**
f_7	*Best*	$2.2639e-12$	$6.6613e-16$	$3.7148e-08$	**4.4409e – 16**
	Worst	$7.4111e-08$	$1.5832e-13$	$5.4580e-05$	**8.8818e – 16**
	Mean	$5.7420e-09$	$8.1512e-15$	$8.3602e-06$	**5.5067e – 16**
	Std.dev.	$1.0735e-08$	$2.3411e-14$	$1.0672e-05$	**1.1587e – 16**
f_8	*Best*	$5.6661e-10$	$5.9857e-19$	$1.9708e-07$	**3.5648e – 25**
	Worst	$7.2059e-05$	$3.8437e-12$	$6.8985e-04$	**1.3835e – 18**
	Mean	$3.1774e-06$	$1.8491e-13$	$8.8162e-05$	**3.8202e – 20**
	Std.dev.	$9.4007e-06$	$5.9948e-13$	$1.1552e-04$	**1.5044e – 19**

parison with PSO, CSA and ISA is clear from the Table. Figure 3 shows the convergence characteristics for the algorithms studied in finding the minimum value of the Rosenbrock test function. It may be observed that both LISA and ISA has improved convergence in comparison with PSO and CSA for the initial generations. The *Lévy* flight principle employed in LISA has helped it to achieve improved steady state error minimisation in comparison with all the algorithms compared.

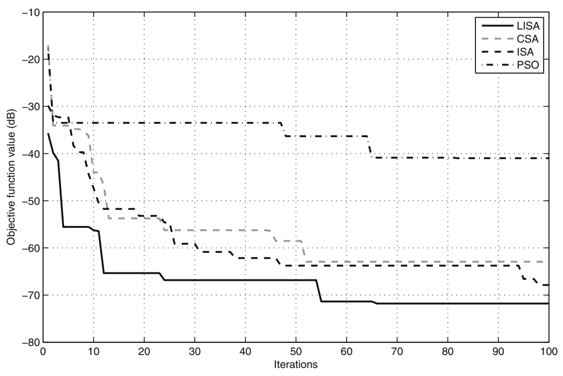


Fig. 4 Parameter estimation of Rossler's system

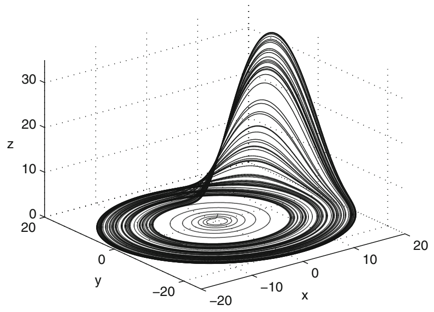


Fig. 5 Chaotic behaviour of Rossler's system

3.2 Case B: Parameter Identification in Chaotic Systems

The newly proposed algorithm is further tested by applying it to identify the parameters of a Rossler's system, which is an integer order chaotic system. The parameter identification task has been formulated as an optimisation task which is performed using ISA, PSO, CSA and LISA. The behaviour of a Rossler's system is governed by

$$\begin{aligned}\dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c)\end{aligned} \tag{7}$$

with, $a = 0.1, b = 0.1$ and $c = 14$. We have considered a search space of $[0, 30]$ for all the three unknown parameters and the initial states have been taken as $x(0) = 1$, $y(0) = 1$ and $z(0) = 1$. Figure 5 shows the chaotic behaviour of a Rossler's system. The main task in the parameter identification problem is to estimate the values of a, b and c from the measured values of input and output. The number of agents considered for each algorithm are 50 and number of iterations are 100. Figure 4 shows the convergence characteristics for this problem using the four algorithms studied. The

Table 3 Comparison of parameter estimation of Rossler's system among PSO, ISA, CSA and LISA

		Mean			Standard deviation			
Parameters	PSO	ISA	CSA	LISA	PSO	ISA	CSA	LISA
a (0.1)	0.2612	0.069981081	0.067983552	0.090070019	0.1642	0.0569	0.0674	0.0901
b (0.1)	4.3974	0.157075247	0.077400552	0.12495929	2.9885	0.1129	0.093	0.0468
c (14)	19.3503	14.09600148	13.96521907	14.02875086	3.8001	0.1284	0.1192	0.1136
MSE	2.20E−04	1.86E−07	2.21E−07	7.80E−08	1.19E−04	1.28E−07	9.37E-08	3.84E−08

mean and standard deviation of the error in estimation of the parameters (averaged over thirty independent experiments) is shown in Table 3. The improved capability of the proposed LISA in finding global solutions, in comparison with other popular algorithms, is evident from the simulation study.

4 Conclusions

A new optimisation algorithm, LISA inspired from ISA has been proposed in the paper. Levy flight is incorporated to the elements of ISA and along with it a new criteria for the selection is employed. The proposed LISA has been applied for solving both unimodal and multimodal benchmark optimisation problems. It is clearly evident from the simulation study that LISA is more efficient in optimisation than other classical evolutionary algorithms in terms of accuracy and convergence rate. LISA has also been tested on parameter identification problem of chaotic system which further demonstrates its better performance.

Acknowledgments This work was supported by the Department of Science and Technology, Government of India under the INSPIRE Faculty Award Scheme (IFA-13 ENG-45).

References

1. Bhargava, V., Fateen, S.E.K., Bonilla-Petriciolet, A.: Cuckoo search: a new nature-inspired optimization method for phase equilibrium calculations. Fluid Phase Equilib. **337**, 191–200 (2013)
2. Dorigo, M., Gambardella, L.M.: Ant colony system:a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
3. Gandomi, A., Roke, D.: Engineering optimization using interior search algorithm. In: 2014 IEEE Symposium on Swarm Intelligence (SIS), pp. 1–7 (Dec 2014)
4. Gandomi, A.H.: Interior search algorithm (ISA): a novel approach for global optimization. ISA Trans. **53**(4), 1168–1183 (2014)
5. Geem, Z.W., Kim, J.H., Loganathan, G.: A new heuristic optimization algorithm: Harmony search. Simulation **76**(2), 60–68 (2001)
6. Goldberg, D.E.: Genetic Algorithms. Pearson Education India (2006)
7. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. **39**(3), 459–471 (2007)
8. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (Nov 1995)
9. Patwardhan, A.P., Patidar, R., George, N.V.: On a cuckoo search optimization approach towards feedback system identification. Digit. Sig. Proc. **32**, 156–163 (2014)
10. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
11. Wong, P.K., Wong, K.I., Vong, C.M., Cheung, C.S.: Modeling and optimization of biodiesel engine performance using kernel-based extreme learning machine and cuckoo search. Renewable Energy **74**, 640–647 (2015)
12. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: Proceedings of IEEE World Congress on Nature and Biologically Inspired Computing, pp. 210–214 (2009)

Hybridization of Adaptivity and Chaotic Dynamics for Differential Evolution

Roman Senkerik, Michal Pluhacek, Donald Davendra, Ivan Zelinka, Zuzana Kominkova Oplatkova and Jakub Janostik

Abstract This research deals with the hybridization of the two modern approaches for evolutionary algorithms, which are the adaptivity and complex chaotic dynamics. This paper aims on the investigations on the chaos-driven adaptive Differential Evolution (DE) concept. This paper is aimed at the embedding of discrete dissipative chaotic systems in the form of chaotic pseudo random number generators for the state of the art adaptive representative jDE. Repeated simulations for two different driving chaotic systems were performed on the IEEE CEC 13 benchmark set. Finally, the obtained results are compared with the canonical not-chaotic jDE.

Keywords Differential evolution · Deterministic chaos · jDE

1 Introduction

This research deals with the hybridization of the two modern approaches for evolutionary algorithms, which are the adaptivity and embedding of complex chaotic dynamics. This paper is aimed at investigating the influence of chaotic dynamics to the performance of adaptive Differential Evolution (DE) algorithm [1].

R. Senkerik (✉) · M. Pluhacek · Z.K. Oplatkova · J. Janostik
Tomas Bata University in Zlin Faculty of Applied Informatics,
Nam T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
e-mail: senkerik@fai.utb.cz

M. Pluhacek
e-mail: pluhacek@fai.utb.cz

D. Davendra · I. Zelinka
Technical University of Ostrava, Faculty of Electrical Engineering and Computer, Science, 17. Listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
e-mail: donald.davendra@vsb.cz

I. Zelinka
e-mail: ivan.zelinka@vsb.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_12

The adaptive strategy of interest within this paper is the state of the art representative jDE. [2] Although a number of DE variants have been recently developed, including adaptive and self-adaptive strategies, the focus of this paper is the further development and experimental testing of ChaosDE concept, which is based on the embedding of chaotic systems in the form of Chaos Pseudo Random Number Generators (CPRNG) into the DE.

A chaotic approach generally uses the chaotic map in the place of a pseudo random number generator [3]. This causes the heuristic to map unique regions, since the chaotic map iterates to new regions. The task is then to select a very good chaotic map as the pseudo random number generator.

The focus of our research is the direct embedding of chaotic dynamics in the form of CPRNG for evolutionary algorithms. The initial concept of embedding chaotic dynamics into the evolutionary algorithms is given in [4]. Later, the initial study [5] was focused on the simple embedding of chaotic systems in the form of chaos pseudo random number generator (CPRNG) for DE and Self Organizing Migration Algorithm (SOMA) [6] in the task of optimal PID tuning. Also the PSO (Particle Swarm Optimization) algorithm with elements of chaos was introduced as CPSO [7]. The concept of ChaosDE proved itself to be a powerful heuristic also in combinatorial problems domain [8]. At the same time the chaos embedded PSO with inertia weigh strategy was closely investigated [9], followed by the introduction of a PSO strategy driven alternately by two chaotic systems [10].

Firstly, the motivation for this research is proposed. The next sections are focused on the description of evolutionary algorithm jDE, the concept of chaos driven jDE and the experiment description. Results and conclusion follow afterwards.

2 Related Work and Motivation

This research is an extension and continuation of the previous successful initial experiments with chaos driven PSO and DE algorithms [11, 12].

In this paper the concept for jDE strategy driven by chaotic maps (systems) is introduced. From the aforementioned previous research it follows, that very promising experimental results were obtained through the utilization of different chaotic dynamics. And at the same time it was clear that different chaotic systems have different effects on the performance of the algorithm. Unfortunately numerous experiments proved that the positive influence of chaotic dynamics is suppressed by either fixed (not-adaptive) or wrong settings of control parameters for meta-heuristic. The idea was then to connect into the one simple concept two different influences to the performance of DE, which are control parameters adaptability and chaotic dynamics as CPRNG. Such a concept is presented in this research. The adaptive jDE strategy is hybridized here with chaotic CPRNGs.

3 Differential Evolution

DE is a population-based optimization method that works on real-number-coded individuals [1, 13]. DE is quite robust, fast, and effective, with global optimization ability. A schematic of the canonical DE strategy is given in Fig. 1.

There are essentially five sections to the code depicted in Fig. 1. Section 1 describes the input to the heuristic. D is the size of the problem, G_{max} is the maximum number of generations, NP is the total number of solutions, F is the scaling factor of the solution and CR is the factor for crossover. F and CR together make the internal tuning parameters for the heuristic.

Section 2 in Fig. 1 outlines the initialization of the heuristic. Each solution $x_{i,j,G=0}$ is created randomly between the two bounds $x^{(lo)}$ and $x^{(hi)}$. The parameter j represents the index to the values within the solution and parameter i indexes the solutions within the population. So, to illustrate, $x_{4,2,0}$ represents the fourth value of the second solution at the initial generation.

After initialization, the population is subjected to repeated iterations in Sect. 3.

Section 4 describes the conversion routines of DE. Initially, three random numbers r_1, r_2, r_3 are selected, unique to each other and to the current indexed solution i in the population in *4.1*. Henceforth, a new index j_{rand} is selected in the solution. j_{rand} points to the value being modified in the solution as given in *4.2*. In *4.3*, two solutions, $x_{j},_{r1},_{G}$ and $x_{j},_{r2,G}$ are selected through the index r_1 and r_2 and their values subtracted. This value is then multiplied by F, the predefined scaling factor. This is added to the value indexed by r_3.

```
1. Input: $D, G_{\max}, NP \ge 4, F \in (0, 1+), CR \in [0,1]$, and initial bounds: $\vec{x}^{(lo)}, \vec{x}^{(hi)}$.
2. Initialize: $\begin{cases} \forall i \le NP \wedge \forall j \le D : x_{i,j,G=0} = x_j^{(lo)} + rand_j[0,1] \bullet \left(x_j^{(hi)} - x_j^{(lo)}\right) \\ i = \{1,2,\ldots,NP\},\ j = \{1,2,\ldots,D\},\ G = 0,\ rand_j[0,1] \in [0,1] \end{cases}$
3. While $G < G_{\max}$
    $\forall i \le NP$
        4. Mutate and recombine:
            4.1 $r_1, r_2, r_3 \in \{1,2,\ldots.,NP\}$, randomly selected, except: $r_1 \ne r_2 \ne r_3 \ne i$
            4.2 $j_{rand} \in \{1,2,\ldots,D\}$, randomly selected once each $i$
        4.3 $\forall j \le D, u_{j,i,G+1} = \begin{cases} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\ \quad \text{if } (rand_j[0,1] < CR \vee j = j_{rand}) \\ x_{j,i,G} \quad \text{otherwise} \end{cases}$
        5. Select
        $\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G+1} & \text{if } f(\vec{u}_{i,G+1}) \le f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases}$
    $G = G + 1$
```

Fig. 1 DE schematic

However, this solution is not arbitrarily accepted in the solution. A new random number is generated, and if this random number is less than the value of *CR*, then the new value replaces the old value in the current solution. The fitness of the resulting solution, referred to as a perturbed (or trial) vector $u_{j,i,G}$., is then compared with the fitness of $x_{j,i,G}$. If the fitness of $u_{j,i,G}$ is better than the fitness of $x_{j,i,G}$., then $x_{j,i,G}$. is replaced with $u_{j,i,G}$; otherwise, $x_{j,i,G}$. remains in the population as $x_{j,i,G+1}$. Hence the competition is only between the new *child* solution and its *parent* solution.

4 The Concept of Chaotic jDE

A simple and very efficient adaptive DE strategy, known as jDE, was introduced by Brest et al. [2]. This adaptive strategy utilizes basic DE/rand/1/bin scheme with a simple adaptive mechanism for mutation and crossover control parameters *F* and *CR*. The ensemble of these two control parameters is assigned to each individual of the population and survives if an individual is successful. The initialization of values of *F* and *CR* is fully random with uniform distribution for each solution in population. If the new generated solution is not successful, i.e. trial vector has worse fitness than compared original active individual; the new (possibly) mutated control parameters disappear together with not successful solution. The both DE control parameters can be randomly mutated with given probabilities τ_1 and τ_2. If the mutation condition happens, new random value of $CR \in [0, 1]$ is generated, together with new value of *F* which is mutated in $[F_l, F_u + F_l]$. These new control parameters are thereafter stored in the new population. Input parameters are typically set to $F_l = 0.1$, $F_u = 0.9$, $\tau_1 = 0.1$, and $\tau_2 = 0.1$ as originally given in [2, 14].

The general idea of basic ChaosDE (Chaos_jDE) and CPRNG is to replace the default pseudorandom number generator (PRNG) with the discrete chaotic map. As the discrete chaotic map is a set of equations with a static start position, we created a random start position of the map, in order to have different start position for different experiments (runs of EA's). This random position is initialized with the default PRNG, as a one-off randomizer. Once the start position of the chaotic map has been obtained, the map generates the next sequence using its current position.

In this research, direct output iterations of the chaotic maps were used for the generation of real numbers in the process of crossover based on the user defined *CR* value and for the generation of the integer values used for selection of individuals.

Previous successful initial experiments with chaos driven PSO and DE algorithms [11, 12] have manifested that very promising experimental results were obtained through the utilization of Delayed Logistic, Lozi, Burgers and Tinkerbelt maps. The last two mentioned chaotic maps have unique properties with connection to DE: strong progress towards global extreme, but weak overall statistical results, like average cost function (CF) value and std. dev., and tendency to premature stagnation. While through the utilization of the Lozi and Delayed Logistic map the continuously stable and very satisfactory performance of ChaosDE was achieved.

In this research, two representatives of these aforementioned two different influences were embedded into jDE, thus Chaos_jDE concept was introduces and tested.

5 Chaotic Maps

This section contains the description of discrete dissipative chaotic maps used as the chaotic pseudo random generators for jDE. Following chaotic maps were used: Burgers (1), and Lozi map (2).

The Burgers mapping is a discretization of a pair of coupled differential equations which were used by Burgers [15] to illustrate the relevance of the concept of bifurcation to the study of hydrodynamics flows. The map equations are given in (4) with control parameters $a = 0.75$ and $b = 1.75$ as suggested in [16].

$$\begin{aligned} X_{n+1} &= aX_n - Y_n^2 \\ Y_{n+1} &= bY_n + X_n Y_n \end{aligned} \tag{1}$$

The Lozi map is a discrete two-dimensional chaotic map. The map equations are given in (2). The parameters used in this work are: $a = 1.7$ and $b = 0.5$ as suggested in [16]. For these values, the system exhibits typical chaotic behavior and with this parameter setting it is used in the most research papers and other literature sources.

$$\begin{aligned} X_{n+1} &= 1 - a|X_n| + bY_n \\ Y_{n+1} &= X_n \end{aligned} \tag{2}$$

The illustrative histograms of the distribution of real numbers transferred into the range <0–1> generated by means of studied chaotic maps are in Fig. 2.

6 Results

IEEE CEC 2013 benchmark set [17] was utilized within this experimental research for the purpose of performance comparison of two versions of chaotic jDE and original (canonical) jDE,

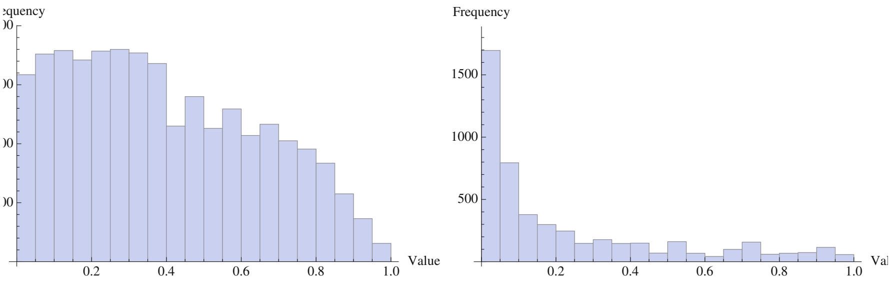


Fig. 2 Histogram of the distribution of real numbers transferred into the range <0–1> generated by means of the chaotic Lozi map (left) and Burgers map (right) – 5000 samples

Experiments were performed in the combined environments of Wolfram Mathematica and C language; canonical jDE for comparisons therefore used the built-in C language pseudo random number generator Mersenne Twister C representing traditional pseudorandom number generators in comparisons. All experiments used different initialization, i.e. different initial population was generated in each run.

Within this research, one type of experiment was performed. It utilizes the maximum number of generations fixed at 1500 generations, Population size of 75 and dimension *dim* = 30. This allowed the possibility to analyze the progress of all studied DE variants within a limited number of generations and cost function evaluations.

To track the influence of adaptive multi-chaotic approach, an experiment encompasses three groups:

- Chaos_jDE with Lozi map
- Chaos_jDE with Burgers map
- State of the art adaptive representative canonical jDE (with default PRNG).

Results of the experiments are shown in Tables 1 and 2. Table 1 contains the final average CF values, whereas Table 2 shows the minimum found CF values representing the best individual solution for all 50 repeated runs of two versions of

Chaos_jDE, and canonical jDE. Finally the Table 3 shows the overall performance comparison for all aforementioned DE versions. Within Tables 1 and 2, the bold values represent the best performance, italic equal. Some big difference (even for known extreme) for f2-f5 is given by the high complexity of composite function.

Table 1 Final average CF values: Performance comparison for two versions of Chaos_DE and canonical jDE on CEC 13 Benchmark Set, *dim* = 30, max. Generations = 1500

DE Version/Function	f min	jDE	Chaos_jDE Lozi	Chaos_jDE Burgers
f1	−1400	*−1400.00*	*−1400.00*	*−1400.00*
f2	−1300	9423247.23	2459441.56	**1714008.06**
f3	−1200	3478148.18	**1550246.13**	3370508.56
f4	−1100	24893.68	4461.53	**4174.92**
f5	−1000	*−1000.00*	*−1000.00*	*−1000.00*
f6	−900	−879.19	**−882.45**	−874.45
f7	−800	−787.25	−794.09	**−796.10**
f8	−700	−679.01	**−679.02**	−679.01
f9	−600	−565.52	−570.81	**−582.50**
f10	−500	−498.98	−499.84	**−499.96**
f11	−400	**−362.54**	−353.36	−333.69
f12	−300	−132.14	−150.63	**−189.51**
f13	−200	−20.26	−30.34	**−37.65**
f14	−100	**2164.55**	2608.70	4672.46
f15	100	**7303.26**	7353.36	7556.45
f16	200	202.73	**202.59**	202.71
f17	300	**387.12**	398.23	442.64
f18	400	619.07	616.37	**611.72**
f19	500	**507.63**	508.16	511.21
f20	600	612.54	612.42	**612.25**
f21	700	1007.26	**1002.47**	1011.02
f22	800	**3641.35**	3954.23	5702.65
f23	900	8447.26	**8354.20**	8667.09
f24	1000	1206.76	**1203.90**	1204.07
f25	1100	1396.26	1359.35	**1351.13**
f26	1200	1400.35	1400.11	**1400.07**
f27	1300	1880.70	1717.35	**1663.98**
f28	1400	*1700.00*	*1700.00*	*1700.00*

Table 2 Best solutions - minimal CF values: Performance comparison for two versions of Chaos_DE and canonical jDE on CEC 13 Benchmark Set, *dim* = 30, max. Generations = 1500

DE Version/Function	f min	jDE	Chaos_jDE Lozi	Chaos_jDE Burgers
f1	−1400	*−1400.00*	*−1400.00*	*−1400.00*
f2	−1300	2689600.43	748839.05	**535148.42**
f3	−1200	**17586.85**	17720.10	30670.27
f4	−1100	17613.32	1440.31	**919.47**
f5	−1000	*−1000.00*	*−1000.00*	*−1000.00*
f6	−900	−884.44	−885.63	**−886.09**
f7	−800	−795.20	−798.20	**−799.49**
f8	−700	−679.16	−679.14	**−679.24**
f9	−600	−569.61	−588.38	**−594.86**
f10	−500	−499.36	*−499.99*	*−499.99*
f11	−400	**−372.68**	−362.96	−361.32
f12	−300	−160.77	−275.49	**−281.09**
f13	−200	−41.88	−76.97	**−173.79**
f14	−100	**1525.76**	2038.37	3329.97
f15	100	**6554.82**	6839.32	6941.17
f16	200	**201.93**	201.94	202.16
f17	300	**377.58**	379.90	412.40
f18	400	583.29	**578.69**	585.44
f19	500	506.38	**505.08**	508.15
f20	600	612.10	611.88	**611.60**
f21	700	*900.00*	*900.00*	*900.00*
f22	800	**2991.66**	3127.59	4277.37
f23	900	7533.31	**7270.73**	7873.32
f24	1000	1202.35	1200.67	**1200.32**
f25	1100	1366.27	**1300.53**	1336.44
f26	1200	1400.17	1400.05	**1400.02**
f27	1300	1688.58	1622.76	**1604.68**
f28	1400	*1700.00*	*1700.00*	*1700.00*

Table 3 Overall statistical performance comparison

	jDE	Chaos_DE Lozi	Chaos_jDE Burgers
Average CF value	6 +, 3 =, 19-	7 +, 3 = , 18-	**12 +, 3 =, 13-**
Min. CF value	7 +, 4 =, 17-	4 +, 5 = , 19-	**12 +, 5 =, 11-**

7 Conclusion

The primary aim of this work is to use and test the hybridization of natural chaotic dynamics as the chaotic pseudo random number generators and self-adaptive mechanism for differential evolution algorithm. In this paper the novel concept of Chaos_jDE strategy driven by chaotic maps (systems) is introduced. These two different influences to the performance of DE were connected here into the one adaptive chaotic concept. Repeated simulations were performed on the IEEE CEC 13 benchmark set. The obtained results were compared with the original state of the art adaptive representative canonical jDE. The findings can be summarized as follows:

- The high sensitivity of the DE to the internal dynamics of the chaotic PRNG is fully manifested.
- When comparing the Chaos_jDE and original jDE, the chaos driven heuristic has outperformed the original jDE in both cases of final average CF values and min. CF values.
- Based on the previous point, we can assume that in case of differential evolution, the sensitivity to the internal chaotic dynamics driving the selection of individuals and crossover process may significantly increase the influence of adaptive tuning of control parameters. Nevertheless this will be more experimentally investigated in future work and research experiments.
- Furthermore the direct embedding of chaotic dynamics into the evolutionary/swarm based algorithms is advantageous, since it can be easily implemented into any existing algorithm. Also there are no major adjustments in the code required (instead of calling function Rand(), one iteration of chaotic system is taken).

Acknowledgements This work was supported by Grant Agency of the Czech Republic - GACR P103/15/06700S, further by financial support of research project NPU I No. MSMT-7778/2014 by the Ministry of Education of the Czech Republic and also by the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089, partially supported by Grant of SGS No. SP2015/142 and SP2015/141 of VSB - Technical University of Ostrava, Czech Republic and by Internal Grant Agency of Tomas Bata University under the projects No. IGA/FAI/2015/057 and IGA/FAI/2015/061.

References

1. Price, K.V.: An introduction to differential evolution. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization. McGraw-Hill Ltd., pp. 79–108 (1999)
2. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**(6), 646–657 (2006)

3. Aydin, I., Karakose, M., Akin, E.: Chaotic-based hybrid negative selection algorithm and its applications in fault and anomaly detection. Expert Syst. Appl. **37**(7), 5285–5294 (2010)
4. Caponetto, R., Fortuna, L., Fazzino, S., Xibilia, M.G.: Chaotic sequences to improve the performance of evolutionary algorithms. IEEE Trans. Evol. Comput. **7**(3), 289–304 (2003)
5. Davendra, D., Zelinka, I., Senkerik, R.: Chaos driven evolutionary algorithms for the task of PID control. Comput. Math Appl. **60**(4), 1088–1104 (2010)
6. Zelinka, I.: SOMA — self-organizing migrating algorithm. New Optimization Techniques in Engineering, vol. 141, pp. 167–217. Studies in Fuzziness and Soft Computing. Springer, Berlin (2004)
7. Coelho, L.D.S., Mariani, V.C.: A novel chaotic particle swarm optimization approach using Hénon map and implicit filtering local search for economic load dispatch. Chaos, Solitons Fractals **39**(2), 510–518 (2009)
8. Davendra, D., Bialic-Davendra, M., Senkerik, R.: Scheduling the Lot-Streaming Flowshop scheduling problem with setup time with the chaos-induced Enhanced Differential Evolution. In: 2013 IEEE Symposium on Differential Evolution (SDE), pp 119–126, 16–19 April 2013
9. Pluhacek, M., Senkerik, R., Davendra, D., Kominkova Oplatkova, Z., Zelinka, I.: On the behavior and performance of chaos driven PSO algorithm with inertia weight. Comput. Math Appl. **66**(2), 122–134 (2013)
10. Pluhacek, M., Senkerik, R., Zelinka, I., Davendra, D.: Chaos PSO algorithm driven alternately by two different chaotic maps - An initial study. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 2444–2449, 20–23 June 2013
11. Senkerik, R., Pluhacek, M., Davendra, D., Zelinka, I., Oplatkova, Z.: Performance testing of multi-chaotic differential evolution concept on shifted benchmark functions. In: Polycarpou, M., de Carvalho, A.P.L.F., Pan, J.-S., Woźniak, M., Quintian, H., Corchado, E. (eds.) Hybrid Artificial Intelligence Systems, Lecture Notes in Computer Science, vol. 8480, pp. 306–317. Springer International Publishing. doi:10.1007/978-3-319-07617-1_28
12. Pluhacek, M., Senkerik, R., Zelinka, I., Davendra, D.: New adaptive approach for chaos PSO algorithm driven alternately by two different chaotic maps – an initial study. In: Zelinka, I., Chen, G., Rössler, O.E., Snasel, V., Abraham, A. (eds.) Nostradamus 2013: Prediction, Modeling and Analysis of Complex Systems, vol 210, pp. 77–87. Advances in Intelligent Systems and Computing. Springer International Publishing. doi:10.1007/978-3-319-00542-3_9
13. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution - A Practical Approach to Global Optimization. Natural Computing Series, Springer, Berlin (2005)
14. Tvrdík, J., Poláková, R., Veselský, J., Bujok, P.: Adaptive variants of differential evolution: towards control-parameter-free optimizers. In: Zelinka, I., Snášel, V., Abraham, A. (eds.) Handbook of Optimization. Intelligent Systems Reference Library, vol. 38, pp. 423–449. Springer, Berlin Heidelberg (2013)
15. Elabbasy, E., Agiza, H., El-Metwally, H., Elsadany, A.: Bifurcation analysis, chaos and control in the burgers mapping. Int. J. Nonlinear Sci. **4** (3), 171–185
16. Sprott, J.C.: Chaos and Time-Series Analysis. Oxford University Press (2003)
17. Liang, J.J., Qu, B.-Y., Suganthan, P.N., Hernández-Díaz, A.G.: Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization, Technical Report 201212. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore (2013)

A Minimisation of Network Covering Services in a Threshold Distance

Miloš Šeda and Pavel Šeda

Abstract In this paper, we deal with a special version of the set covering problem, which consists in finding the minimum number of service centres providing specialized services for all customers (or larger units, such as urban areas) within a reasonable distance given by a threshold. If a suitable threshold is found that makes it possible to determine a feasible solution of the task, the task is transformed into a general set covering problem. However, this has a combinatorial nature and, because it belongs to the class of NP-hard problems, for a large instance of the problem, it cannot be used to find the optimal solution in a reasonable amount of time. In the paper, we present a solution by means of two stochastic heuristic methods - genetic algorithms and simulated annealing – and, using a test instance from OR-Library, we present the parameter settings of both methods and the results achieved.

Keywords Set covering • Unicost problem • Threshold • Reachability matrix • Genetic algorithm • Repair operator

1 Introduction

There are numerous discussions on how to optimise a network of public facilities (e.g. hospitals and schools) that provide essential services (health, education) for the population so that the cost of their operation is as low as possible and each

M. Šeda (✉)
Faculty of Mechanical Engineering, Brno University of Technology,
Technická 2, 616 69 Brno, Czech Republic
e-mail: seda@fme.vutbr.cz

P. Šeda
IBM Global Services Delivery Center, Technická 21, 616 00 Brno, Czech Republic
e-mail: pavelseda@email.cz

P. Šeda
Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1,
613 00 Brno, Czech Republic

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems
and Computing 378, DOI 10.1007/978-3-319-19824-8_13

inhabitant or an urban district has at least one of the service centres in an affordable distance. It is clear that the question of what is an affordable distance is debatable and could be determined by agreement of the ruling political parties. In this text, however, we ignore the political aspects and address a formal mathematical approach to solve such tasks.

In the literature, the general set covering problem is studied that does not address any threshold of availability, but it is directly given by the matrix of binary values and a covering of all columns by suitable choice of rows is looked for. This task is an NP-hard problem [3] and, for a larger problem instance, can be solved in a reasonable time only by heuristic methods.

The problem that we investigate can be converted to a set covering problem because, by using a threshold, the distance matrix is changed to binary reachability matrix. However, if the threshold is chosen inadequately, the original task may have a number of degenerative cases, described in the following section, and we will show how setting an appropriate threshold makes it possible to find a solution using genetic algorithms and simulated annealing.

2 Problem Formulation

Assume that the transport network contains m vertices, that can be used as operating service centres, and n vertices to be served, and for each pair of vertices v_i (considered as service centres) and v_j (serviced vertex) their distance d_{ij} is given and D_{max} is the maximum distance which will be accepted for operation between the service centres and serviced vertices.

The aim is to determine which vertices must be used as service centres for each vertex to be covered by at least one of the centres and for the total number of operating centres to be minimal.

Remark 1.

1. A condition necessary to solve the task is that all of the serviced vertices are reachable from at least one place where an operating service centre is considered.
2. Serviced vertex v_j is reachable from vertex v_i, which is regarded as an operating service centre if $d_{ij} \leq D_{\max}$. If this inequality is not satisfied, vertex v_j is unreachable from v_i.

Here, $a_{ij} = 1$ means that vertex v_j is reachable from v_i and $a_{ij} = 0$ means that it is not if v_i is operating service centre i.

Similarly, $x_i = 1$ means that service centre i is selected while $x_i = 0$ means that it is not selected.

Then, the set covering problem can be described by the following mathematical model:

Minimise

$$z = \sum_{i=1}^{m} x_i \tag{1}$$

subject to

$$\sum_{i=1}^{m} a_{ij} x_i \geq 1, \qquad j = 1, \ldots, n \tag{2}$$

$$x_i \in \{0, 1\}, \qquad i = 1, \ldots, m \tag{3}$$

The objective function (1) represents the number of operating centres, constraint (2) means that each serviced vertex is assigned at least to one operating service centre. The parameter D_{max} represents a threshold of service reachability.

Example:

Consider the following *distance matrix* which expresses service centres and serviced vertices (= customer locations) and $D_{\max} = 40$.

$$\begin{array}{c|cccccccc}
 & \multicolumn{8}{c}{\text{serviced vertices (customer locations)}} \\
\text{service centres} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 5 & 41 & 50 & 26 & 38 & 60 & 44 & 59 \\
2 & 49 & 82 & 13 & 67 & 68 & 20 & 32 & 31 \\
3 & 45 & 17 & 61 & 45 & 67 & 48 & 53 & 127 \\
4 & 37 & 170 & 195 & 32 & 77 & 88 & 90 & 30 \\
5 & 58 & 42 & 25 & 101 & 133 & 32 & 21 & 78
\end{array}$$

From $D_{\max} = 40$ we get the *reachability matrix* of serviced vertices from service centres.

$$\begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
5 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0
\end{array}$$

Since only service centre 3 is reachable from the second serviced vertex (serviced vertex 2 is covered by the 3rd service centre) and only service centre 1 is reachable from serviced vertex 5, these service centres must not be omitted. These two centres cover serviced vertices 1, 4, 5, and 2.

This means that the service centres must be found that cover the remaining uncovered vertices 3, 6, 7 and 8. This can be achieved either by selecting the service centres 2 and 5, or 4 and 5.

Thus, the example has two solutions, where four vertices are sufficient to cover serviced vertices (either 1, 3, 2, 5, or 1, 3, 4, 5)

In the general case, however, the selection of service centres for k uncovered vertices has 2^k possibilities and, thus, the task complexity increases exponentially with the number of uncovered vertices.

For large k, we must solve the task by a heuristic method, e.g., by a genetic algorithm [2, 4, 7, 8], ant colonies [6], differential evolution, SOMA (Self Organizing Migrating Algorithm) [12], HC12 meta-heuristic algorithm [13].

2.1 Special Cases

In this section, we will summarise cases for which the problem has no solution, or specified data need a modification.

We will show this directly by using the below reachability matrices.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

In the 3rd row of the previous matrix, we can see that service centre 3 can be omitted because it exceeds the threshold distance to all customers and nobody would visit it.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

In the 5th column of the previous matrix, we can see that the threshold distance is too low and the 5th customer has no chance to visit a centre in a reachable distance. The threshold must be increased to get at least one 1 in each column.

If a service centre must not be omitted, it represents only one centre for a customer, i.e., in the customer column, there is only one 1. Of course, we can have more necessary centres which cannot be omitted. However, if necessary centres cover all customers, then no centre needs to be added to the necessary ones and we immediately have a solution.

3 Computational Results

Since the mathematical model is simple, it seems that the problem could be solved by one of the optimisation toolboxes.

E.g. in GAMS (General Algebraic Modelling System) the main part of code is as follows:

```
VARIABLES
  X(I) decision variables
  XSum objective function;
BINARY VARIABLE X;
EQUATIONS
  EQ6(J) cover conditions
  EQ5    objective function (number of selected centres);
  EQ6(J) .. SUM(I,A(I,J)*X(I)) =G= 1;
  EQ5    .. XSum =E= SUM(I,C(I)*X(I));

MODEL COVER /ALL/;
SOLVE COVER USING MIP MINIMIZING XSum;
DISPLAY XSum.L, X.L;
```

However, this software tool is successful for only “small” instances. All computations leading to optimum were performed in a few seconds, but for larger instances, they ended with a run time error with GAMS indicating “insufficient space to update U-factor …”. It is caused by the fact that time complexity of the problem with m rows is $O(2^m)$ and, say, for an instance with 200 rows and 2000 columns tested in the following sections, its searching space has 2^{200} possible selections and $2^{200} = (1024)^{20} \approx 10^{60}$.

Therefore, for these cases, a heuristic approach must be used. Two of them, genetic algorithm and simulated annealing, have been implemented and recommendations of their parameter settings are presented, based on many tests with various sets of possible operators (selection, crossover, mutation, etc.).

3.1 Genetic Algorithm

Since the principles of GA's are well-known, we will only deal with GA parameter settings for the problems to be studied. Now we describe the general settings and the problem-oriented setting used in our application.

Individuals in the population (*chromosomes*) are represented as binary strings of length n, where a value of 0 or 1 at bit i (*gene*) implies that $x_i = 0$ or 1 in the solution respectively.

The *population size* is usually set in the range [50, 200], in our programme, implemented in Java, 200 individuals in the population were used, because 50 individuals led to a reduction chromosome diversity and premature convergence.

Initial population is obtained by generating random strings of 0 s and 1 s in the following way: First, all bits in all strings are set to 0, and then, for each of the strings, randomly selected bits are set to 1 until the solutions (represented by strings) are feasible.

The *fitness function* corresponds to the objective function to be maximised or minimised, here, it is minimised.

Three most commonly used methods of *selection* of two parents for *reproduction*, roulette selection, ranking selection, and tournament selection, were tested.

As to *crossover*, uniform crossover, one-point and two-point crossover operators were implemented.

Mutation was set to 5, 10 and 15 %, exchange mutation, shift mutation, and mutation inspired by well-known *Lin-2-Opt change operator* usually used for solving the travelling salesman problem [5] were implemented.

In *replacement* operation two randomly selected individuals with below-average fitness were replaced by generated children.

Termination of a GA was controlled by specifying a maximum number of generations *tmax*, e.g. $tmax \leq 10000$.

The chromosome is represented by an m-bit binary string S where m is the number of columns in the SCP. A value of 1 for bit i implies that service centre i is in the solution and 0 otherwise.

Since the SCP is a minimisation problem, the lower the fitness value, the more fit the solution is. The fitness of a chromosome for the unicost SCP is calculated by (4).

$$f(S) = \sum_{i=1}^{m} S_i \tag{4}$$

The binary representation causes problems with generating infeasible chromosomes, e.g., in initial population, in crossover, and/or mutation operations. To avoid infeasible solutions, a *repair operator* [10] is applied.

Let
$I = \{1, \ldots, m\}$ = the set of all rows;
$J = \{1, \ldots, n\}$ = the set of all columns;
S = the set of rows in a solution;
U = the set of uncovered columns;
w_j = the number of rows that cover column j, $j \in J$ in S.
$\alpha_j = \{i \in I \mid a_{ij} = 1\}$ = the set of rows that cover column j, $j \in J$;
$\beta_i = \{j \in J \mid a_{ij} = 1\}$ = the set of columns that are covered by row i, $i \in I$;

The repair operator for the unicost SCP has the following form:

```
initialise $w_j := |S \cap \alpha_j|, \forall j \in J$;
initialise $U := \{j \mid w_j = 0, \forall j \in J\}$;
**for each** column $j$ in $U$ (in increasing order of $j$) **do**
  **begin** find the first row $i$ (in increasing order of $i$) in $\alpha_j$
            that minimises $1/|U \cap \beta_i|$;
        $S := S + i$;
        $w_j := w_j + 1, \ \forall j \in \beta_i$;
        $U := U - \beta_i$
  **end**;
**for each** row $i$ in $S$ (in decreasing order of $i$) **do**
  **if** $w_j \geq 2, \forall j \in \beta_i$
    **then begin** $S := S - i$;
            $w_j := w_j - 1, \ \forall j \in \beta_i$
        **end** ;
{ $S$ is now a feasible solution to the SCP and contains no redundant rows }
```

Initialising steps identify the uncovered columns. **For** statements are "greedy" heuristics in the sense that in the 1st **for**, rows with low cost-ratios are being considered first and in the 2nd **for**, rows with high costs are dropped first whenever possible.

The genetic algorithm was tested using an instance from OR-Library [1] with 200 rows and 2000 columns.

Figure 1 shows the computation of a fitness function as depending on the sequence of generations of the algorithm. The upper characteristic shows the average value of the fitness function of all individuals in the population and the lower one shows the fitness function of the best individual in the population. It can be seen that the algorithm converges quite rapidly to a pseudo-optimal solution (no optimal solution is known for such a large instance).

The test results showed that one-point crossover gave worse results than two-point crossover, which, in turn, was even worse than a uniform crossover. Obviously, it would also be appropriate to monitor the width of the middle part of a chromosome in the two-point crossover because, when its width is very small, it brings very little change in comparison with parental chromosomes and reduces the diversity of genetic material in the population, which increases the risk of small or almost no changes in the fitness function after only a small number of generations.

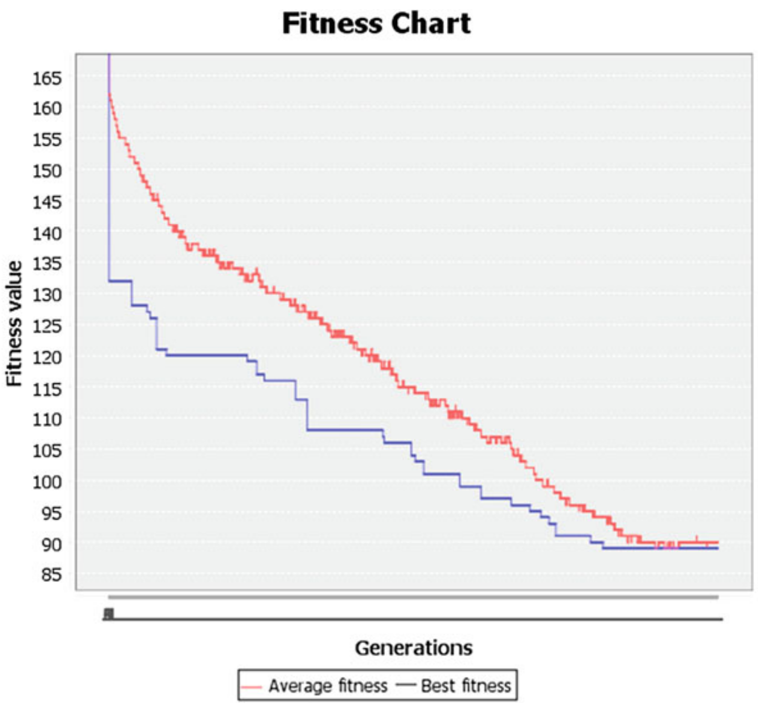


Fig. 1 Computation by genetic algorithm (instance with 200 rows and 2000 columns, roulette wheel selection, uniform crossover, mutation probability 5 %, 6500 generations, the best solution from 10 runs)

3.2 Simulated Annealing

The simulated annealing (SA) technique has been given much attention as a general-purpose way of solving difficult optimisation tasks. Its idea is based on simulating the cooling of a material in a heat bath - a process known as annealing. More details may be found in [9].

As in the previous section we only mention the parameter settings:

- *initial temperature* $T_0 = 10000$,
- *final temperature* $T_f = 1$,
- *temperature reduction function* $\alpha(t) = t * decrement$, where *decrement* $= 0.999$,
- *neighbourhood* for a given solution $\mathbf{x}_0$ is generated so that each neighbour is given by the inverse of one position in the binary string representing $\mathbf{x}_0$.

(This means that the number of neighbours of $\mathbf{x}_0$ is equal to the number of positions in this string.) (Fig. 2)

It is obvious that simulated annealing converges to the same, or a very close approximation of the optimal solution, which is consistent with the well-known "No free lunch theorem" [11]. Because the simulated annealing is a one-point method, only the dependence of the objective function for gradually updated points $\mathbf{x}_0$ (centres for generating neighbours) in the search space is plotted. The chart also shows that the likelihood of transfer to a worse neighbour is decreasing steadily as temperature decreases, which is also in line with the principle of the method.

As in the genetic algorithm, it is necessary to apply the repair operator for the

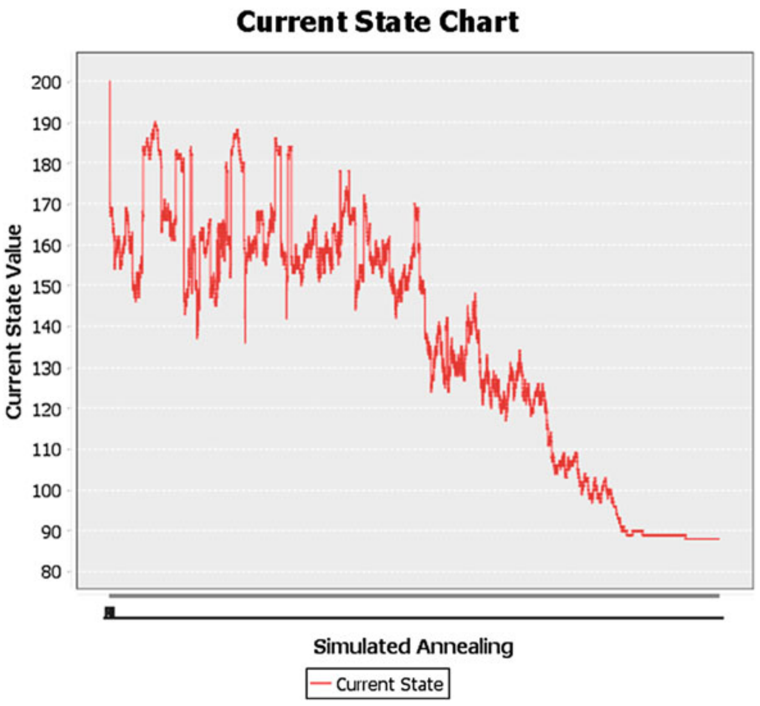


Fig. 2 Computation by simulated annealing (the same instance as in GA)

selected solution in the neighbourhood, as described in the previous section.

The computational time of GA for the tested instance with 200 rows was only 19 s on a computer with a processor frequency of 2.4 GHz and operating memory of 4 GB while SA takes more than 1 min. The reason is that, in each GA iteration, we generate only two children while SA creates the neighbourhood with 200 neighbours in each iteration when a new position is chosen.

An even worse situation would be for a hill-climbing algorithm, where each individual would have to be modified by the repair operator in order to determine the best neighbour. For simulated annealing, it is sufficient to apply the repair operator only to a randomly selected individual from the neighbourhood.

4 Practical Aspects

In the foregoing, we investigated the set covering problem in a version that did not take into account the importance of individual centres. If we apply the above procedure to minimising a network of hospitals and schools, we could get a solution where hospitals and schools respectively in cities would be omitted. In this case, it is appropriate to consider their importance given by their size or necessity. As the objective function is minimised, it is necessary to determine the weights so that the lower the weight, the higher the priority.

It could even be suitable to classify facilities with high importance as necessary as if they represent unique choice for some customers.

If weights of service centres are expressed by coefficients c_j, the corresponding mathematical model would change as follows:

Minimise

$$z = \sum_{i=1}^{m} c_i x_i \tag{5}$$

subject to

$$\sum_{i=1}^{m} a_{ij} x_i \geq 1, \quad j = 1, \ldots, n \tag{6}$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, m \tag{7}$$

From the point of view of the problem representation and parameter settings, there is no change with the exception of the objective function, for which Eq. (5) is used rather than Eq. (1).

5 Conclusions

In this paper, we studied the set covering problem in a special case, in which a threshold is defined. This task may be used for optimising networks providing public services with operation costs being minimal [12].

Due to the exponential time complexity, classical optimisation programs often based on a branch and bound method cannot be used to solve larger instances of (mixed-)integer programming problems. Therefore a heuristic approach was proposed.

The programme for solving this problem was implemented and parameter settings recommended based on testing many combinations of possible selections of their operators. It was shown that these methods yield very similar results when executed tens of times.

In the future, we will include weights of service centres in our considerations and try to implement other modern heuristic methods [13, 14].

References

1. Beasley, J.E.: OR-library: distributing test problems by electronic mail. J. Oper. Res. Soc. **11**, 1069–1072 (1990)
2. Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. Eur. J. Oper. Res. **94**, 392–404 (1996)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
4. Goldberg, D.E.: The Design of Innovation (Genetic Algorithms and Evolutionary Computation). Kluwer Academic Publishers, Dordrecht (2002)
5. Gutin, G., Punnen, A.P. (eds.): The Traveling Salesman Problem and Its Variations. Kluwer Academic Publishers, Dordrecht (2002)
6. Lessing, L., Dumitrescu, I., Stützle, T.: A comparison between ACO algorithms for the set covering problem. In: Dorigo, M. (ed.) ANTS 2004. Lecture Notes in Computer Science, vol. 3172, pp. 1–12. Springer-Verlag, Berlin (2004)
7. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Berlin (1996)
8. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics. Springer, Berlin (2002)
9. Reeves, C.R.: Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publications, Oxford (1993)
10. Šeda, M., Roupec, J., Šedová, J.: Transportation problem and related tasks with application in agriculture. Int. J. Appl. Math. Inf. **8**, 26–33 (2014)
11. Wolpert, D.H., McReady, W.G.: No Free Lunch Theorems for Optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)
12. Zelinka, I., Snášel, V., Abraham, A. (eds.): Handbook of Optimization: From Classical to Modern Approach. Berlin. Springer, Berlin (2013)
13. Matousek, R.: HC12: the principle of CUDA implementation. In: Proceedings of 16th International Conference on Soft Computing – MENDEL 2010. Mendel series vol. 2010, pp. 303–308, Brno (2010), ISSN: 1803-3814
14. Matousek, R., Zampachova, E.: Promissing GAHC and HC12 algorithms in global optimization tasks. J. Optim. Methods Softw. **26**(3), 405–419 (2011)

CUDA-based Analytic Programming by Means of SOMA Algorithm

Lumir Kojecky and Ivan Zelinka

Abstract Analytic programming is one of methods of symbolic regression that is composing simple elements into more complex units. This process can be used e.g. for approximation of measured data with suitable mathematical formula. To find the most suitable mathematical formula, it is necessary to use an evolutionary algorithm. The constructed formulas can consist of mathematical operators, functions, variables and constants. Since values of these constants are not known at the time of construction of the formula, it is necessary to estimate the values by means of another evolutionary algorithm. Unfortunately, due to this estimation, the whole process becomes too slow. Therefore, this algorithm is implemented in one of the most widespread programming architecture NVIDIA CUDA and the results in terms of execution time are compared.

Keywords Analytic programming · Symbolic regression · SOMA · Evolutionary algorithms · Parallelization · Cuda

1 Introduction

Symbolic regression [4, 20] is a process that is composing simple elements into more complex units. This process can be used for approximation of measured data with suitable mathematical formula, for neural network synthesis, logic circuit synthesis, etc. There are several approaches to symbolic regression, for example Genetic programming [7, 13], Grammatical Evolution [11, 16] or Analytic programming (AP) [18, 19].

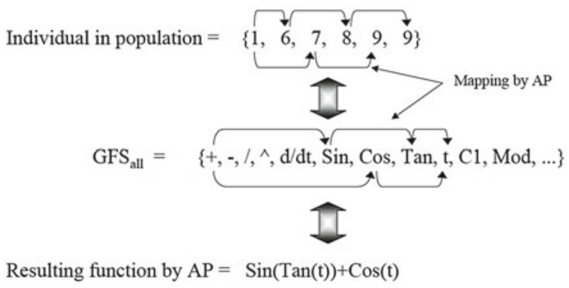
The main goal of analytic programming is to compose simple elements into more complex units – programs. These elements are placed in General Functional Set

L. Kojecky (✉) · I. Zelinka
Department of Computer Science, VSB-Technical University of Ostrava, FEI Tr. 17. Listopadu 15, Ostrava, Czech Republic
e-mail: lumir.kojecky@vsb.cz

I. Zelinka
e-mail: ivan.zelinka@vsb.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_14

(GFS) and they can consist of mathematical operators, functions, variables or constants. At the input, AP takes a set of indexes-pointers to the GFS. At the output, AP generates a program that is composed of elements indexed at the input. This process is for better imagination visualized in Fig. 1.



Individual in population = {1, 6, 7, 8, 9, 9}
Mapping by AP
GFS_all = {+, -, /, ^, d/dt, Sin, Cos, Tan, t, C1, Mod, ...}
Resulting function by AP = Sin(Tan(t))+Cos(t)

Fig. 1 Main principle of AP [19]

As the input set of indexes can be used individuals of the arbitrary evolutionary algorithm [19] like Differential evolution [17], Particle Swarm Optimization [5], Simulated Annealing [6], or Self-organizing Migrating Algorithm (SOMA) [2, 3, 12]. The input is then converted to a program, i.e. mathematical formula, and then is calculated area (difference) between the formula and measured data. The area is used as the individual's fitness – AP is used as fitness function in essence. As an example of an evolutionary algorithm, SOMA was chosen.

SOMA is a stochastic optimization algorithm based in essence on vector operations, imitating the social behavior of cooperating individuals. The population of individuals is randomly initialized at the beginning, evaluated and then is chosen individual with the highest fitness, which becomes a leader. In each migration loop, all other individuals are moving through the space towards the leader and seeking their best position with the highest fitness. Before each individual's step is created a binary perturbation (PRT) vector, that controls movement of the individual in allowed dimensions. Individuals thus do not move directly towards to the leader and the probability of finding the optimal position is increasing.

There are two options how to handle constants in AP. The first option [7] is to add many randomly generated constant values to GFS. However, the number of all possible generated formulas rapidly increases. The second option [19] is to add only one general constant into GFS, called K. This constant is then inserted into a formula at various places and it is indexed (1). Values of all indexed constants are estimated using a second evolutionary algorithm. Unfortunately, due to this estimation, the whole process becomes too slow. Therefore, this research explores the advantages of applying both evolutionary algorithm [1, 9] and symbolic regression algorithm [14, 15] on the GPU.

$$K_1 \cdot \sin(x + K_2) \tag{1}$$

2 Experiment Design

2.1 NVIDIA CUDA

The Compute Unified Device Architecture (CUDA) [10] is one of the most widespread parallel programming architecture invented by NVIDIA. It provides us a platform for accessing the NVIDIA GPU for processing in essence by extensions to standard programming languages like C/C++.

The general outline of CUDA is given in Fig. 2. In CUDA programs, functions launched from the CPU are referred to as kernels. Each kernel consists of a number of blocks that are executed independently from each other around an array of Streaming Multiprocessors. Each block is divided into a number of threads. These threads have access to the registers and shared memory in a block they belong to. The kernels can be made up of any combination of blocks and threads – the whole configuration depends on the application requirements. However, the number of threads in one block is limited by 1,024.

There are several types of memory having each its capabilities and purpose, differing in access latency, size, address space, scope, lifetime and whether it is cached or writable (summarized in the Table 1):

1. Global memory resides in device memory and it is used for communication between host and device, therefore is accessible from both CPU and GPU and it has lifetime of whole application. However, access latency to the global memory is very high (hundreds of cycles).
2. Constant memory shares the same memory banks as global memory, but there is only a limited amount of the memory that can be declared. On the other side this memory is cached and access to this memory is much faster than access to global memory.
3. Shared memory is a very fast on-chip memory that has a lifetime of a block and it is accessible only by threads within the same block. There is also a limited amount of the memory that can be declared (48 kB).
4. Registers are the fastest on-chip memory used for thread variables. Number of registers that are available per block is also limited.
5. Local memory is used for storing variables that will not fit in registers. This memory has the same access latency as global memory.

The CUDA syntax is really simple and does not differ from standard C/C++ code so much. Memory allocation of variables is given as: `cudaMalloc(&name, size * sizeof(type));`, where `name` is the variable name, `type` is the data type and `size` is the total size to be allocated. Then, data is copied to the GPU using `cudaMemcpy(GPU,CPU,size*sizeof(type),cudaMemcpyHostToDevice);`.

The kernel (marked by `__global__` keyword) is launched using execution configuration [10] syntax: `kernel <<< grid, block, shared >>> (parameters);`, where `grid` refers to the number of blocks in the grid, `block` refers to the number of threads in blocks, and `shared` refers to the size of allocated shared memory (this

Table 1 Summary of the scope and lifetime of CUDA memories

Memory	Scope EA	Lifetime
Global	Grid	Application
Constant	Grid	Application
Shared	Block	Kernel
Register	Thread	Kernel
Local	Thread	Kernel

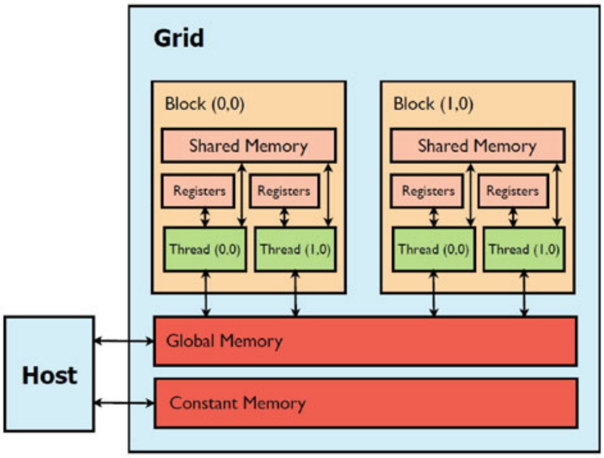


Fig. 2 CUDA outline [1]

parameter is optional). Within the kernel, blocks and thread indexes are given by `blockIdx` and `threadIdx` variables. After execution of kernels, the processed data is copied back to the host.

2.2 Code Design on the GPU

When designing the code on the GPU, it is highly desirable to write bigger kernels with more functionality instead of calling many small kernels. It is also desirable to avoid many memory transfers from CPU to GPU and vice versa. Finally, due to very high access latency of global memory, using shared memory is one of the most crucial steps in this application.

According to these recommendations, all memories in the program are generated only once. There is only one array for outer SOMA population, one array for all expressions, or one big array for all inner SOMA populations. Memory transfers between the CPU and GPU are reduced only to 3 cases:

1. Transferring GFS elements and data to approximate from the CPU to GPU. The data is read from the application configuration file.
2. Transferring number of constants in an expression from the GPU to CPU to decide whether to run the inner evolution to estimate the constants or not.
3. Transferring the final (best) expression and its constants from the GPU to CPU.

Not only fitness function evaluation [1], but all computation tasks are also moved from CPU to GPU (population generation, migrations, etc.). This approach reduces transferring data from CPU to GPU to a minimum. According to [10], any access to data in global memory compiles to a single global memory instruction if the size of the data type is 1, 2, 4, 8, or 16 bytes and the data is naturally aligned (address is a multiple of that size). This means that it is desirable to use structures of these sizes and align them well in memory.

As mentioned above, using shared memory is one of the most crucial steps in this application. First step is to identify whether it is worth to use it. To use the shared memory, it has to be allocated, data has to be loaded from global memory to shared memory, and all threads in a block need to be synchronized (to ensure that data are loaded by all threads). Then, the calculation is performed and the result data is stored back to global memory. This approach is unnecessary resource wasting when single thread is accessing the memory only for simple operation. In this program, shared memory is widely used – each individual or expression is processed in one block so this is an ideal candidate for using shared memory:

1. Mathematical fitness function evaluation is most often sequential process (if we do not take account of parallel reduction). Instead of step-by-step accessing slow global memory by one thread it is possible to load all values to shared memory by one instruction (at once by multiple threads) and then step-by-step access fast shared memory.
2. A similar approach is in the application used for determining the population leader individual. One hundred threads copy the individuals' values at once and sequential computation is processed over fast shared memory.
3. Shared memory is also very useful for composing expressions. GFS is loaded into shared memory at once as well as appropriate indexed expressions from the shared GFS. All these indexed expressions, one by one, have to be paired with other ones (pointers to the arguments) and checked whether the final expression is pathological or not.
4. The most important is to use the shared memory at the time of the expression evaluation, where each expression is evaluated concurrently in 50 points. The expression and the points to evaluate, as well as constant values, are loaded into shared memory that is also used for intermediate computations.

To efficient use of GPU resources it is necessary to appropriately design the grid. We decided that one individual in SOMA, as well as one expression in AP, is handled by one block of threads. Then, the grid consists of *PopSize* blocks and each block consists of *Dim* threads. This grid layout allows us to efficiently utilize shared memory to load the whole individual or the whole expression.

In population migration loops, each individual in the *main population* performs $\left\lfloor \frac{PathLength}{Step} \right\rfloor$ steps towards the leader individual. The steps are independent of each other and can be handled in parallel as one big *step population* of $PopSize \cdot \left\lfloor \frac{PathLength}{Step} \right\rfloor$ size. The entire migration loop process is then divided into the following steps (kernels):

1. Selection of the leader individual of the main population – one block of *PopSize* threads loads the values of all individuals into the shared memory and evaluates the best one.
2. Migration – calculation of the new possible individual's position (step) performed by $PopSize \cdot \left\lfloor \frac{PathLength}{Step} \right\rfloor$ blocks. Each thread in the block calculates a new position and saves it into the step population.
3. Composing the expressions from the step populations individuals – the process of the expression composition is, except parallel loading GFS to shared memory, fully sequential. Each element of the expression has to be paired with another ones (its arguments) and it has also to be checked whether it is not pathological (i.e. all arguments have to be closed).
4. Expression evaluation – the most time-consuming step. The process of evaluation is performed sequentially for one expression (element by element) but concurrently for all 50 points. In this step may occur two cases:
   (a) *The expression has no constants* – it is evaluated in the standard way.
   (b) *The expression has at least one constant* – the constants are estimated by a secondary evolutionary algorithm with its own main and step population. Individuals are here used as the constant values that are substituted into the expression which is then evaluated in the standard way.
5. Selection of the leader individuals of the step populations – *PopSize* blocks are identifying the best of all steps for each individual in the main population.
6. Moving the step leaders to the main population, if they have smaller cost than the original individual.

3 Simulations and Results

Experiments done in this research were focused on validation of the application using the GPU in terms of performance measurement and comparison with current single-thread C# application. For experiments here was at first measured performance of SOMA itself (AllToOne), optimizing Schwefel's function (2). Then was measured performance of the basic AP algorithm (no use of constants). Finally, the performance of the AP algorithm with estimation of constant values by means of secondary evolutionary algorithm was measured.

Each experiment was repeated 50 times. For testing, as measured data to approximation was generated 50 points of Quintic equation (3) in the interval $[-1, 1]$.

Table 2 Algorithm settings for SOMA. The dimension of the inner EA equals to the number of estimated constants

Parameter	Outer EA	Inner EA	Basic AP	SOMA only
Dimensions	50	–	[20, 30, 50]	[20, 30, 50]
PopSize	100	50	100	100
Migrations	12	11	100	200
PRT	0.1	0.1	0.1	0.1
PathLength	3	2	3	3
Step	0.11	0.21	0.11	0.11

Table 3 Dimension 20: SOMA only

Implementation	Max	Median	Average	Min
C#	3.07	2.88	2.90	2.85
CUDA	0.16	0.06	0.06	0.05
Speedup	19.19	48.00	48.33	57.00

Table 4 Dimension 20: Basic AP

Implementation	Max	Median	Average	Min
C#	6.32	6.19	6.18	6.02
CUDA	0.3	0.11	0.12	0.11
Speedup	21.07	56.27	51.50	54.73

Table 5 Dimension 30: SOMA only

Implementation	Max	Median	Average	Min
C#	4.24	4.20	4.20	4.17
CUDA	0.20	0.06	0.07	0.06
Speedup	21.20	70.00	60.00	69.50

The operating parameters for all use cases of SOMA are given in the Table 2. Simulations also included testing of optimization of various dimensions for SOMA and basic AP algorithm.

$$\sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|}) \tag{2}$$

$$x^5 - 2x^3 + x \tag{3}$$

All experiments were done on a desktop PC with Intel Core i7 at 3.4 GHz (3.9 GHz Turbo) and NVIDIA GeForce 750 Ti that has 1 GPU core (640 CUDA cores)

Table 6 Dimension 30: Basic AP

Implementation	Max	Median	Average	Min
C#	9.96	9.70	9.68	9.33
CUDA	0.26	0.15	0.15	0.15
Speedup	38.31	64.67	64.53	62.20

Table 7 Dimension 50: SOMA only

Implementation	Max	Median	Average	Min
C#	7.56	7.20	7.22	7.14
CUDA	0.21	0.11	0.11	0.10
Speedup	36.00	65.45	65.64	71.40

Table 8 Dimension 50: Basic AP

Implementation	Max	Median	Average	Min
C#	18.23	17.70	17.67	16.99
CUDA	0.41	0.29	0.30	0.29
Speedup	44.96	61.03	58.90	58.59

Table 9 Dimension 50: AP with constants

Implementation	Max	Median	Average	Min
C#	2210.68	1735.53	1726.06	1262.56
CUDA	116.60	111.63	111.68	106.94
Speedup	18.96	15.55	15.46	11.81

with 2,048 MB RAM memory, 1,020 MHz (1,085 MHz boost) core clock speed and compute capability version 5.0.

All results from the simulations are summarized in Tables 3, 4, 5, 6, 7, 8, 9. In each Table is recorded the worst (maximal), median, average and the best (minimal) simulation execution time of all 50 simulations of both C# and CUDA implementations.

4 Conclusion

In this paper was implemented CUDA-based Analytic programming by means of SOMA algorithm. The results in Tables 3, 4, 5, 6, 7, 9 show the performance of newly implemented algorithm under CUDA parallel platform compared to the same algorithm written in C# language. Three kinds of the algorithm were implemented:

pure SOMA, AP without secondary evolution to estimating constants, and full AP with constant estimation.

Simulations of pure SOMA and basic AP showed us that average speedup of CUDA implementation compared to C# implementation was surprisingly from 48× to 65× (and sometimes even more). This speedup was achieved especially by proper use of shared memory and minimizing memory transfers between the CPU and GPU. Simulations included testing of optimization of various dimensions. From the time results can be seen that lower dimension means lower computation time [8]. However, results in Tables 3 and 5 for CUDA implementation are almost same. The computation time for lower dimensions is so low and most of the time is consumed i.e. by memory allocation.

Simulations of full AP showed us that average speedup of CUDA implementation was 15×. In this case, the problem size is so big that one graphic card cannot process all data in one big transaction – the transaction has to be divided into more smaller transactions. The CPU results from Table 9 show us that there is also a big difference (75 %) between minimal and maximal execution time. This is caused by fact that number of constants in the expression is every time different – in the expression there can be only 1 constant or 20 constants. In CUDA, the difference is not so big for the reasons mentioned above.

All the results of the simulations are satisfactory, but AP is in essence a simple algorithm and there are many much more complicated algorithms. In the future research of the algorithm parallelization field, we want to utilize more than one CUDA GPU or to utilize the other parallel platforms to achieve even better results. Currently, we are preparing similar simulations on Anselm, x86-64 Intel based supercomputer hosted at VSB-Technical University of Ostrava.

Acknowledgments The following grants are acknowledged for the financial support provided for this research: Grant Agency of the Czech Republic – GACR P103/15/06700S and partially supported by Grant of SGS No. SP2015/142, VSB-Technical University of Ostrava.

References

1. Davendra, D., Gaura, J., Bialic-Davendra, M., Senkerik, R.: Cuda based enhanced differential evolution: a computational analysis. ECMS, pp. 399–404 (2012)
2. Davendra, D., Zelinka, I.: Flow shop scheduling using self organizing migration algorithm. In: Modelling and Simulation.[Proceedings of the European Conference.] Nicosia: European Council of Modelling and Simulation, pp. 195–200 (2008)
3. Davendra, D., Zelinka, I.: Optimization of quadratic assignment problem using self organising migrating algorithm. Comput. Inf. **28**(2), 169–180 (2012)
4. Davidson, J., Savic, D.A., Walters, G.A.: Symbolic and numerical regression: experiments and applications. Inf. Sci. **150**(1), 95–117 (2003)
5. Kennedy, J., Kennedy, J.F., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann (2001)
6. Kirkpatrick, S.: Optimization by simulated annealing: quantitative studies. J. Stat. Phys. **34**(5–6), 975–986 (1984)
7. Koza, J.R.: Genetic Programming III: Darwinian Invention and Problem Solving, vol. 3. Morgan Kaufmann (1999)

8. Matousek, R.: Hc12: the principle of cuda implementation. In: Proceedings of 16th International Conference on Soft Computing—Mendel 2010, vol. 2010, pp. 303–308 (2010)
9. Mussi, L., Daolio, F., Cagnoni, S.: Evaluation of parallel particle swarm optimization algorithms within the cuda architecture. Inf. Sci. **181**(20), 4642–4657 (2011)
10. Nvidia, C.: Nvidia Cuda C Programming Guide. NVIDIA Corporation 120 (2011)
11. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language, vol. 4. Springer Science & Business Media (2003)
12. Onwubolu, G.C., Babu, B.: New Optimization Techniques in Engineering, vol. 141. Springer Science & Business Media (2004)
13. O'Reilly, U.M.: Genetic programming ii: automatic discovery of reusable programs. Artif. Life **1**(4), 439–441 (1994)
14. Pospichal, P., Jaros, J., Schwarz, J.: Parallel genetic algorithm on the cuda architecture. Applications of Evolutionary Computation, pp. 442–451. Springer (2010)
15. Pospichal, P., Murphy, E., O'Neill, M., Schwarz, J., Jaros, J.: Acceleration of grammatical evolution using graphics processing units: computational intelligence on consumer games and graphics hardware. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 431–438. ACM (2011)
16. Ryan, C., Collins, J., Neill, M.O.: Grammatical evolution: evolving programs for an arbitrary language. Genetic Programming, pp. 83–96. Springer (1998)
17. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
18. Zelinka, I.: Analytic programming by means of soma algorithm. In: Proceedings of the 8th International Conference on Soft Computing, Mendel, vol. 2, pp. 93–101 (2002)
19. Zelinka, I., Oplatkova, Z., Nolle, L.: Analytic programming-symbolic regression by means of arbitrary evolutionary algorithms. Int. J. Simul. Syst. Sci. Technol. **6**(9), 44–56 (2005)
20. Zelinka, I., Volna, E.: Neural network synthesis by means of analytic programming-preliminary results. In: Proceedings of the 11th International Conference on Soft Computing, Mendel 2005, vol. 2005 (2005)

Computing Trading Strategies Based on Financial Sentiment Data Using Evolutionary Optimization

Ronald Hochreiter

Abstract In this paper we apply evolutionary optimization techniques to compute optimal rule-based trading strategies based on financial sentiment data. The sentiment data was extracted from the social media service StockTwits to accommodate the level of bullishness or bearishness of the online trading community towards certain stocks. Numerical results for all stocks from the Dow Jones Industrial Average (DJIA) index are presented and a comparison to classical risk-return portfolio selection is provided.

Keywords Evolutionary optimization · Sentiment analysis · Technical trading · Portfolio optimization

1 Introduction

In this paper we apply evolutionary optimization techniques to compute optimal rule-based trading strategies based on financial sentiment data. The number of application areas in the field of sentiment analysis is huge, see especially [1] for a comprehensive overview. The field of Finance attracted research on how to use specific financial sentiment data to find or optimize investment opportunities and strategies, see e.g. [2–4].

This paper is organized as follows. Section 2 describes the financial sentiment data used for the evolutionary approach to optimize trading strategies and portfolios. Section 3 presents an evolutionary optimization algorithm to create optimal trading strategies using financial sentiment data and how to build a portfolio using single-asset trading strategies. Section 4 contains numerical results obtained with the

R. Hochreiter (✉)
Department of Finance, Accounting and Statistics, WU Vienna University of Economics and Business, Wien, Austria
e-mail: ronald.hochreiter@wu.ac.at

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_15

presented algorithm and a comparison to classical risk-return portfolio optimization strategies as proposed by [5] using stock market data from all stocks in the Dow Jones Industrial Average (DJIA) index. Section 5 concludes the paper.

2 Financial Sentiments

We apply financial sentiment data created by PsychSignal.[1] The PsychSignal technology utilizes the wisdom of crowds in order to extract meaningful analysis, which is not achievable through the study of single individuals, see [6] for a general introduction to measurement of psychological states through verbal behavior. Let a group of individuals together be a crowd. Not all crowds are wise, however four elements have been identified, which are required to form a wise crowd: diversity of opinion, independence, decentralization and aggregation as proposed by [7]. These four elements are sometimes present in some forms of social media platforms, e.g. in the financial community StockTwits, from which the crowd wisdom used for the evolutionary approach described in this paper is derived.

Emotions are regarded as being unique to individual persons and occurring over brief moments in time. Let a mood be a set of emotions together. In order to quantify the collective mood of a crowd, distinct emotions of individual members within the crowd must be quantified. Subsequently, individual emotions can be aggregated to form a collective crowd mood. PsychSignals' Natural Language Processing Engine is tuned to the social media language of individual traders and investors based on the general findings of e.g. [8] and of [9] for the financial domain. The engine further targets and extracts emotions and attitudes in that distinct language and categorizes as well as quantifies these emotions from text. The methodology is based on the linguistic inquiry and word count (LIWC) project, which is available publicly.[2] See also [10] for a description of an algorithm on how to generate such a semantic lexicon for financial sentiment data directly.

The main idea is to assign a degree of bullishness or bearishness on stocks depending on the messages, which are sent through StockTwits,[3] which utilizes Twitter's application programming interface (API) to integrate StockTwits as a social media platform of market news, sentiment and stock-picking tools. StockTwits utilized so called *cashtags* with the stock ticker symbol, similar to the Twitter *hashtag*, as a way of indexing people's thoughts and ideas about companies and their respective stocks. The available sentiment data format is described in Table 1. The data was obtained through Quandl,[4] where PsychSignal's sentiment data for stocks can be accessed easily.

[1]http://www.psychsignal.com/.

[2]http://www.liwc.net/.

[3]http://www.stocktwits.com/.

[4]http://www.quandl.com/.

Table 1 PsychSignal.com StockTwits sentiment data format per asset

Variable	Content
Date	Day of the analyzed data
I_{bull}	Each message's language strength of bullishness on a 0–4 scale
I_{bear}	Each message's language strength of bearishness on a 0–4 scale
n_{bull}	Total count of bullish sentiment messages
n_{bear}	Total count of bearish sentiment messages
n_{total}	Total number of messages

Both intensities I_{bull} and I_{bear} are measured on a real-valued scale from 0 to 4, where 0 means no bullish/bearish sentiment and 4 the strongest bullish/bearish sentiment. We normalize these values to 1 by diving the respective value by 4 and obtain the variables i_{bull} and i_{bear}. Furthermore, we create two relative variables for the number of bullish and bearish messages, i.e. $r_{\text{bull}} = n_{\text{bull}}/n_{\text{total}}$ as well as $r_{\text{bear}} = n_{\text{bear}}/n_{\text{total}}$, such that we end up in the final data format we are going to use for subsequent analysis. See Table 2 for an example of the stock with the ticker symbol BA (The Boeing Company).

Table 2 Sentiment values for stock BA starting at the first trading days in 2011

	i_{bull}	i_{bear}	r_{bull}	r_{bear}	n_{total}
2011-01-03	0.59	0	0.50	0	4
2011-01-04	0	0	0	0	1
2011-01-05	0	0.11	0	1	1
2011-01-06	0.61	0	0.25	0	4
2011-01-07	0.52	0	0.17	0	6
2011-01-11	0.67	0	1	0	2

3 Evolutionary Investment Strategy Generation

We aim at creating an evolutionary optimization approach to generate optimal trading strategies for single stocks based on the sentiment analysis data described above. Evolutionary and Genetic Programming techniques have been applied to various financial problems successfully. See especially the series of books on Natural Computing in Finance for more examples, i.e. [11–13]. Generating automatic trading rules has been a core topic in this domain, see especially [14–17], and the references therein.

One main technique in the field of meta-heuristics and technical trading is to let the optimizer generate optimal investment rules given a set of technical indicators. However, instead of using a variety of technical indicators for generating an optimal trading rule, we use the above described financial sentiment data to create investment rules. Thereby we start by using a simplified rule-set approach, whereby the rules are generated by a special genotype encoding. Furthermore, as we are considering to create a portfolio allocation out of the single asset strategies and additionally focus on stocks only, we do not allow for shorting assets, i.e. the decision is whether to enter or exit a long position on a daily basis. The rule is based on the respective sentiment values, such that this basic rule-set can be defined as shown in Eq. (1).

$$\begin{aligned}&\left[\text{IF}(i_{\text{bull}} \geq v_1)\right]_{b_1} \left[\text{AND}\right]_{b_1 \& b_2} \left[\text{IF}(r_{\text{bull}} \geq v_2)\right]_{b_2} \text{THEN long.}\\ &\left[\text{IF}(i_{\text{bear}} \geq v_3)\right]_{b_3} \left[\text{AND}\right]_{b_3 \& b_4} \left[\text{IF}(r_{\text{bear}} \geq v_4)\right]_{b_4} \text{THEN exit.}\end{aligned} \tag{1}$$

Each chromosome within the evolutionary optimization process consists of the values $(b_1, b_2, b_3, b_4, v_1, v_2, v_3, v_4)$, where the b values are binary encoded (0, 1) and the v values are real values between 0 and 1. The b values indicate whether the respective part of the rule notated in square brackets is included (1) or not (0), while the v values represent the concrete values within the conditions. Consider the following example: the (randomly chosen) chromosome (0,1,1,1,0.4,0.3,0.5,0.2) results in the rule-set shown in Eq. (2).

$$\begin{aligned}&\text{IF } (r_{\text{bull}} \geq 0.3) \text{ THEN long position.}\\ &\text{IF } (i_{\text{bear}} \geq 0.5) \text{ AND IF } (r_{\text{bear}} \geq 0.2) \text{ THEN exit position.}\end{aligned} \tag{2}$$

In this special case, the sum of b_1 and b_2 as well as b_3 and b_4 must be greater or equal to 1, to have at least one condition for entering and leaving the long position. We end up with nine different possible assignments for b. A repair operator has to be applied after each evolutionary operation, which may distort this structure.

The evaluation of the chromosomes is such that the respective trading strategy is tested on the in-sample testing set of length T, i.e. we obtain a series of returns $r_1, \ldots, r_T$ for each chromosome, which can be evaluated with different financial metrics. The following strategy performance characteristics are considered:

– The cumulative return r, and the standard deviation σ.
– The maximum drawdown d, and the Value-at-Risk v_α ($\alpha = 0.05$), as well as
– the ratio s of expected return divided by the standard deviation, which is based on the Sharpe-ratio proposed by [18].

We use simple mutation operators for new populations because the chromosome encoding of the investment rule described above is short, i.e. contains only eight genes. The following mutation operators are applied:

– b binary flip: One randomly selected gene of the binary b part is $0-1$ flipped. The resulting chromosome needs to be repaired with the repair operator, which itself determines randomly, which of the two possibilities is set to 1 if necessary.

- v random mutation: One randomly selected gene of the binary v part is replaced by a uniform random variable between 0 and 1.
- v mutation divided in half: One randomly selected gene of the binary v part is divided in half. The rationale of this operation is that the intensities of bullishness and bearishness are often small, see e.g. Table 3 for the statistics of the sentiment values for a selected stock.

Table 3 Statistical summary of sentiment values for stock BA 2010–2014

	Minimum	First quantile	Median	Mean	Third quantile	Maximum
i_{bull}	0	0	0.3821	0.2987	0.5050	0.8250
i_{bear}	0	0	0	0.1763	0.3887	0.86

Besides these operators, elitist selection is applied as well as a number of random additions will be added to each new population. The structure of the algorithm is a general genetic algorithm, see e.g. [19] for a description of this class of meta-heuristics.

The analysis above is based on single assets. To compose a portfolio out of the single investment strategies, the resulting portfolio will be created as an equally weighted representation of all assets, which are currently selected to be in a long position by its respective trading strategy for each day.

4 Numerical Results

In this section we begin with a description of the data used to compute numerical results in Sect. 4.1. Section 4.2 summarizes the in-sample and out-of-sample results of the evolutionary sentiment trading strategy. A short overview of classical risk-return portfolio optimization is given in Sect. 4.3, and finally a performance comparison is presented in Sect. 4.4. Everything was implemented using the statistical computing language R [20].

4.1 Data

We use data from all stocks from the Dow Jones Industrial Average (DJIA) index using the composition of September 20, 2013, i.e. using the stocks with the ticker symbols AXP, BA, CAT, CSCO, CVX, DD, DIS, GE, GS, HD, IBM, INTC, JNJ, JPM, KO, MCD, MMM, MRK, MSFT, NKE, PFE, PG, T, TRV, UNH, UTX, V, VZ, WMT, XOM.

Training data is taken from the beginning of 2010 until the end of 2013. The out-of-sample tests are applied to data from the year 2014.

4.2 Results of the Evolutionary Optimization

For each stock, the optimal strategy was computed. The evolutionary parameters were set to be as follows:

- The initial population size has been set to 100, and each new population contains the
- 10 best chromosomes of the previous population (elitist selection), as well as
- 20 of each of the three mutation operators described above, and
- 10 random chromosomes, such that the population size is 80.

For evaluation purposes, the parameter s will be maximized. Of course, the system is flexible to use any other risk metric or a combination of metrics. See Table 4 for the in-sample performance results comparing a long-only buy-and-hold strategy of each asset compared to the trading strategy of the best respective strategy, e.g. the best strategy for AXP is $(1, 1, 1, 0, 0.44, 0.41, 0.41, 0.17)$ and for BA $(1, 0, 1, 0, 0.41, 0.37, 0.5, 0.41)$, while for CAT it is $(0, 1, 1, 1, 0.195, 0.34, 0.02, 0.24)$ to give an impression of single strategy results. The cumulative return performance r is raised (sometimes significantly) for almost all assets except for MCD, UTX, V. However, in those three cases the decrease in profit is low. The standard deviation σ is lower (i.e. better) in all cases, which was expected as the algorithm leaves the long-position for a certain time, such that the standard deviation clearly has to decrease. The Sharpe-ratio like metric s is better for all assets but DIS, JNJ, UTX, XOM. Again, the loss in all four cases is low compared to the gain of the other positions. In summary, the in-sample results show that the fitting of the algorithm works very well.

4.3 Classical Portfolio Optimization

To compare the performance of the portfolio created with single asset investment strategies based on financial sentiments with a standard approach to portfolio optimization, we construct a portfolio using classical risk-return portfolio selection techniques. [5] pioneered the idea of risk-return optimal portfolios using the standard deviation of the portfolios profit and loss function as risk measure. In this case, the optimal portfolio x is computed by solving the quadratic optimization problem shown in Eq. 3. The investor needs to estimate a vector of expected returns r of the

Table 4 Single stock in-sample results of the evolutionary optimization

	Long-only stock			Trading strategy		
	r	σ	s	r	σ	s
AXP	1.223	0.016	0.057	1.574	0.013	0.068
BA	1.450	0.016	0.063	1.771	0.013	0.070
CAT	0.575	0.018	0.034	0.978	0.014	0.043
CSCO	−0.070	0.019	0.006	1.246	0.011	0.061
CVX	0.597	0.013	0.042	0.723	0.012	0.044
DD	0.912	0.015	0.051	1.177	0.011	0.070
DIS	1.351	0.015	0.065	1.522	0.013	0.065
GE	0.842	0.015	0.048	1.054	0.013	0.050
GS	0.042	0.019	0.012	0.662	0.010	0.041
HD	1.825	0.014	0.082	2.209	0.012	0.087
IBM	0.430	0.012	0.036	0.711	0.005	0.083
INTC	0.249	0.015	0.022	0.600	0.009	0.043
JNJ	0.415	0.008	0.045	0.415	0.008	0.041
JPM	0.399	0.019	0.027	0.873	0.016	0.037
KO	−0.277	0.019	−0.004	0.363	0.006	0.042
MCD	0.549	0.009	0.052	0.515	0.006	0.060
MMM	0.688	0.013	0.048	0.795	0.012	0.051
MRK	0.359	0.012	0.032	0.516	0.010	0.041
MSFT	0.222	0.014	0.021	0.509	0.012	0.031
NKE	0.190	0.022	0.022	0.511	0.019	0.030
PFE	0.677	0.012	0.048	0.805	0.011	0.049
PG	0.332	0.009	0.036	0.406	0.007	0.046
T	0.238	0.010	0.026	0.397	0.007	0.040
TRV	0.805	0.012	0.053	0.946	0.011	0.064
UNH	1.400	0.016	0.063	2.443	0.013	0.091
UTX	0.621	0.013	0.042	0.563	0.011	0.042
V	1.530	0.017	0.062	1.494	0.010	0.072
VZ	0.471	0.011	0.041	0.572	0.009	0.045
WMT	0.464	0.009	0.045	0.654	0.007	0.058
XOM	0.473	0.012	0.039	0.506	0.010	0.036

assets under consideration as well as the covariance matrix $\mathbb{C}$. Finally the minimum return target μ has to be defined. Any standard quadratic programming solver can be used to solve this problem numerically.

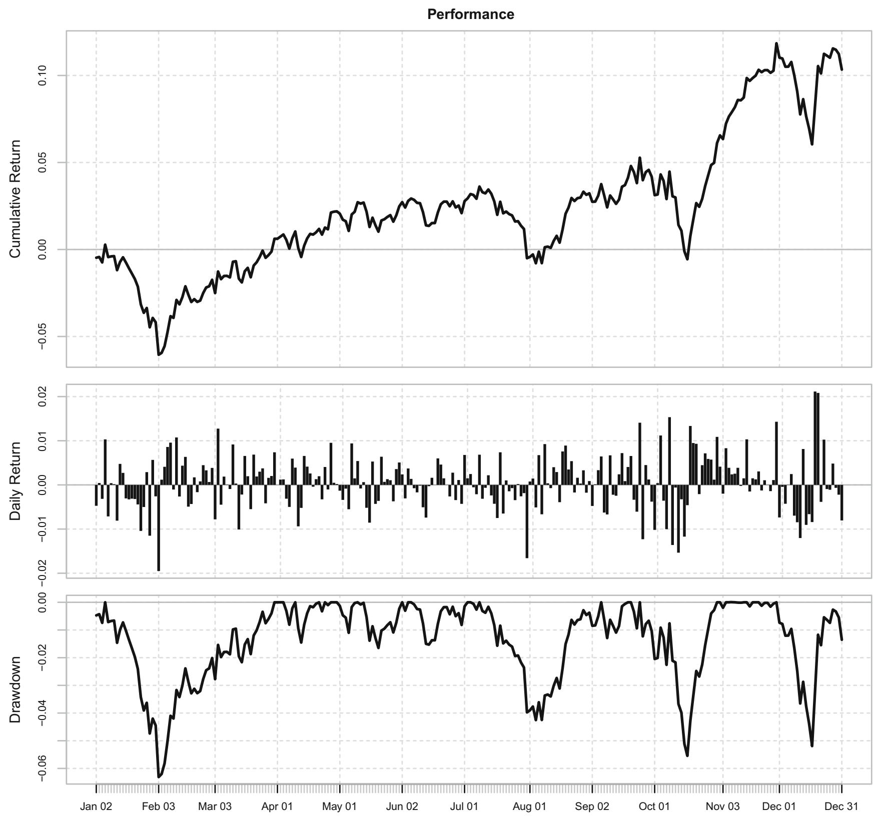


Fig. 1 Out-of-sample performance of a buy-and-hold Markowitz portfolio in 2014

Table 5 Optimal Markowitz portfolio using daily return data from 2010–2013

Ticker symbol	HD	JNJ	MCD	PG	UNH	V	VZ	WMT
Portfolio weight [%]	10.26	16.69	22.67	11.92	6.41	4.22	7.56	20.27

$$\begin{array}{ll} \text{minimize} & x^T \mathbb{C} x \\ \text{subject to} & r \times x \geq \mu \\ & \sum x = 1 \end{array} \tag{3}$$

In addition, we also compare the performance to the 1-over-N portfolio, which equally weights every asset under consideration. It has been shown that there are cases, where this simple strategy outperforms clever optimization strategies, see e.g. [21].

Table 6 Selected risk metrics for the different out-of-sample tests

	Markowitz	1-over-N	Evolutionary
Semi deviation	0.0042	0.0048	0.0038
Downside deviation (Rf = 0 %)	0.0040	0.0047	0.0037
Maximum drawdown	0.0631	0.0687	0.0549
Historical VaR (95 %)	−0.0092	−0.0105	−0.0081
Historical ES (95 %)	−0.0125	−0.0157	−0.0124

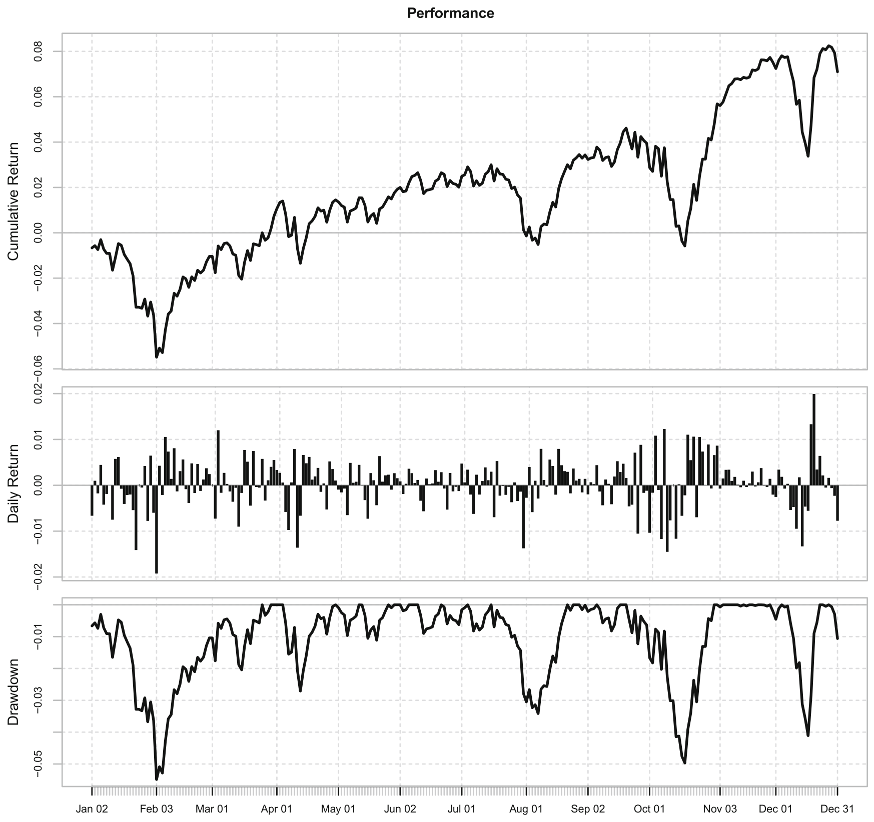


Fig. 2 Out-of-sample performance of an equally weighted portfolio out of the evolutionary sentiment trading strategies in 2014

4.4 Performance Comparison

The asset composition of the optimal Markowitz portfolio is shown in Table 5 - only eight out of the 30 assets are selected. The underlying covariance matrix was estimated from daily returns of the training data, i.e. using historical returns from the beginning of 2010 until the end of 2013. This portfolio is used as a buy-and-hold portfolio over the year 2014. This out-of-sample performance is shown in Fig. 1.[5] While the performance of the 1-over-N portfolio is not shown graphically, Fig. 2 depicts the performance of a portfolio, which is created by equally weighting all single asset trading strategies computed by the evolutionary optimization algorithm based on financial sentiment data into one portfolio. To get a better impression of the differences (see Table 6), where some important risk metrics are summarized for all three strategies. The evolutionary trading portfolio exhibits better risk properties than both other portfolios in all five metrics. Especially important is the reduction of the maximum drawdown, which is of importance to asset managers nowadays, because investors are increasingly looking to this metric if they are searching for secure portfolios.

5 Conclusion

In this paper an evolutionary optimization approach to compute optimal rule-based trading strategies based on financial sentiment data has been developed. It can be shown that a portfolio composed out of the single trading strategies outperforms classical risk-return portfolio optimization approaches in this setting. The next step is to include transaction costs to see how this active evolutionary strategy loses performance when transaction costs are considered. Future extensions include extensive numerical studies on other indices as well as using and comparing different evaluation risk metrics or a combination of metrics. One may also consider to create a more flexible rule-generating algorithm e.g. by using genetic programming. Finally, to achieve an even better out-of-sample performance the recalibrating of the trading strategy can be done using a rolling horizon approach every month.

References

1. Feldman, R.: Techniques and applications for sentiment analysis. Commun. ACM **56**(4), 82–89 (2013)
2. Bollen, J., Mao, H., Zeng, X.: Twitter mood predicts the stock market. J. Comput. Sci. **2**(1), 1–8 (2011)
3. Oliveira, N., Cortez, P., Areal, N.: On the predictability of stock market behavior using StockTwits sentiment and posting volume. Lect. Notes Comput. Sci. **8154**, 355–365 (2013)
4. Smailović, J., Grčar, M., Lavrač, N., Žnidaršič, M.: Stream-based active learning for sentiment analysis in the financial domain. Inf. Sci. **285**, 181–203 (2014)
5. Markowitz, H.: Portfolio selection. J. Financ. **7**(1), 77–91 (1952)
6. Gottschalk, L.A., Gleser, G.C.: Measurement of Psychological States Through the Content Analysis of Verbal Behaviour. University of California Press (1969)

[5]Performance graphs are generated using the `PerformanceAnalytics` R package [22].

7. Surowiecki, J.: The Wisdom of Crowds. Anchor Books (2005)
8. Das, S.R., Chen, M.Y.: Yahoo! for Amazon: sentiment extraction from small talk on the web. Manage. Sci. **53**(9), 1375–1388 (2007)
9. Tumarkin, R., Whitelaw, R.F.: News or noise? Internet postings and stock prices. Financ. Anal. J. **57**(3), 41–51 (2001)
10. Oliveira, N., Cortez, P., Areal, N.: Automatic creation of stock market lexicons for sentiment analysis using StockTwits data. In: Proceedings of the 18th International Database Engineering & Applications Symposium, ACM, pp. 115–123 (2014)
11. Brabazon, A., O'Neill, M. (eds.): Natural computing in computational finance. Volume 100 of Studies in Computational Intelligence. Springer (2008)
12. Brabazon, A., O'Neill, M. (eds.): Natural computing in computational finance, volume 2. Volume 185 of Studies in Computational Intelligence. Springer (2009)
13. Brabazon, A., O'Neill, M., Maringer, D. (eds.): Natural computing in computational finance, volume 3. Volume 293 of Studies in Computational Intelligence. Springer (2010)
14. Bradley, R.G., Brabazon, A., O'Neill, M.: Evolving trading rule-based policies. Lect. Notes Comput. Sci. **6025**, 251–260 (2010)
15. Brabazon, A., O'Neill, M.: Evolving technical trading rules for spot foreign-exchange markets using grammatical evolution. Comput. Manage. Sci. **1**(3–4), 311–327 (2004)
16. Brabazon, A., O'Neill, M.: Intra-day trading using grammatical evolution. In: Brabazon, A., O'Neill, M. (eds.) Biologically Inspired Algorithms for Financial Modelling, pp. 203–210. Springer (2006)
17. Lipinski, P., Korczak, J.J.: Performance measures in an evolutionary stock trading expert system. Lect. Notes Comput. Sci. **3039**, 835–842 (2004)
18. Sharpe, W.F.: The sharpe ratio. J. Portfolio Manage. **21**(1), 49–58 (1994)
19. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003)
20. R Core Team: R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2014)
21. DeMiguel, V., Garlappi, L., Uppal, R.: Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy? Rev. Financ. Stud. **22**(5), 1915–1953 (2009)
22. Peterson, B.G., Carl, P.: PerformanceAnalytics: econometric tools for performance and risk analysis. R package version 1.4.3541 (2014)

Part II
Neural Networks, Self-organization, Machine Learning

An Approach to ANFIS Performance

Stepan Dalecky and Frantisek V. Zboril

Abstract The paper deals with Adaptive neuro-fuzzy inference system (ANFIS) and its performance. Firstly, ANFIS is described as a hybrid system based on fuzzy logic/sets and artificial neural networks. Subsequently, modifications of ANFIS are proposed. The aim of these modifications is to improve performance, accuracy or reduce computational time. Finally, experiments are presented and findings are assessed.

Keywords ANFIS · Artificial neural network · Performance · Fuzzy sets · Fuzzy logic

1 Introduction

Many problems can be successfully solved using some combination of fuzzy logic [6, 7] and artificial neural networks [1, 6, 8]. Each of these two theories has its pros and cons. That is the reason why it is worth to develop hybrid system which takes advantages of both. One well known example of such system is the Adaptive neuro-fuzzy inference system (ANFIS) developed by Jang [3, 4, 6]. ANFIS has borrowed vagueness and fuzziness from fuzzy sets and learning capability from artificial neural networks. This system has been used for solving many problems, e.g. controlling [5], prediction [2] and classification problems. This paper describes ANFIS and proposed modification of ANFIS that leads to better performance, accuracy and/or less computational time.

S. Dalecky (✉) · F.V. Zboril
Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic
e-mail: idalecky@fit.vutbr.cz

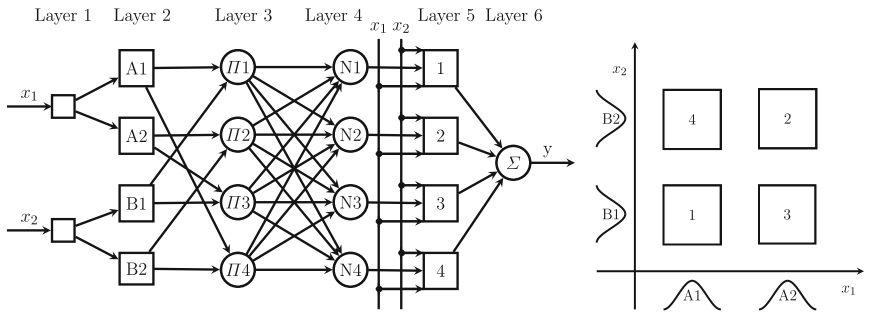
F.V. Zboril
e-mail: zboril@fit.vutbr.cz

2 ANFIS

From fuzzy sets point of view, ANFIS represents Sugeno model of the first order. On the other hand, from artificial neural networks point of view, ANFIS represents six layer feed-forward neural network. Architecture of ANFIS is shown in Fig. 1a.

It is obvious that ANFIS shown in this figure divides an input space as it is shown in Fig. 1b and next four rules can be derived from it:

Parameters k_{ij} are specific constants for each rule and their settings will be described later.



(a) Architecture (b) Division of the input space.

Fig. 1 ANFIS

Rule1 :
IF x_1 is A_1
AND x_2 is B_1
THEN $y = k_{10} + k_{11}x_1 + k_{12}x_2$

Rule2 :
IF x_1 is A_2
AND x_2 is B_2
THEN $y = k_{20} + k_{21}x_1 + k_{22}x_2$

Rule3 :
IF x_1 is A_2
AND x_2 is B_1
THEN $y = k_{30} + k_{31}x_1 + k_{32}x_2$

Rule4 :
IF x_1 is A_1
AND x_2 is B_2
THEN $y = k_{40} + k_{41}x_1 + k_{42}x_2$

2.1 Description of Layers

In this subsection, functions of all layers with respect to ANFIS [3, 6] architecture in Fig. 1a are explained into more details. Symbols $x_{ij}^{(l)}$ and $y_i^{(l)}$ denote j-th input and output of i-th neuron in l-th layer, respectively. If i-th neuron has one input only then this input is denoted simply as $x_i^{(l)}$.

Layer 1–Input Layer The first layer only distributes input values to the second layer using Eq. (1) as follows.

$$y_i^{(1)} = x_i^{(1)} \tag{1}$$

Layer 2 – **Fuzzification Layer** The second layer accomplish fuzzification of inputs using bell membership function Eq. (2) as follows.

$$y_i^{(2)} = bell(x_i^{(2)}; a_i, b_i, c_i) = \frac{1}{1 + \left(\frac{x_i^{(2)} - a_i}{c_i}\right)^{2b_i}} \tag{2}$$

It is obvious that parameters a_i, b_i, c_i determine shape of the bell function: centre, width, and slope.

Layer 3 – **Rule Layer** The third layer corresponds to the rules. Each neuron in this layer represents one rule. Inputs of this layer are membership degrees (outputs of the previous layer), outputs are strength of the rules computed using Eq. (3) as products of all inputs. $C(i)$ symbol denotes the set of neurons of the second layer which are connected to i-th neuron in the third layer.

$$y_i^{(3)} = \prod_{j \in C(i)} x_j^{(3)} = \mu_i \tag{3}$$

Layer 4 – **Normalization Layer** The fourth layer is used for normalization of rule strength. Inputs of all neurons are corresponding outputs of n neurons of the previous layer. Normalized strengths of corresponding rules are computed using Eq. (4).

$$y_i^{(4)} = \frac{x_i^{(4)}}{\sum_{j=1}^{n} x_j^{(4)}} = \frac{\mu_i}{\sum_{j=1}^{n} \mu_j} = \bar{\mu}_i \tag{4}$$

Layer 5 – **Defuzzification Layer** The fifth layer uses Eq. (5) to compute consequent (THEN part) strengths of the rule according to the first order Sugeno model [4, 6]. Inputs of this layer are outputs of the previous layer and input variables x_1 and x_2.

$$y_i^{(5)} = x_i^{(5)} \left[k_{i0} + k_{i1}x_1 + k_{i2}x_2\right] = \bar{\mu}_i \left[k_{i0} + k_{i1}x_1 + k_{i2}x_2\right] \tag{5}$$

Layer 6 – **Sum Layer** The sixth layer computes output of whole network as sum of layer inputs using Eq. (6) as follows.

$$y = y^{(6)} = \sum_{i=1}^{n} x_i^{(6)} = \sum_{i=1}^{n} \bar{\mu}_i \left[k_{i0} + k_{i1}x_1 + k_{i2}x_2\right] \tag{6}$$

2.2 Parameters and Learning

Good performance of proposed model considerably depends on values of parameters a_i, b_i, c_i in the second layer and k_{ij} in the fifth layer. It is impossible to determine optimal value of these parameters directly so we have to estimate them and then tune them to get better performance.

Process of tuning parameters is called learning of ANFIS. Parameters that have to be tuned are divided into two groups:

– parameters that are linear from the output point of view (parameters k_{ij}),
– parameters that are non-linear from the output point of view (parameters a_i, b_i, c_i).

Each group of parameters is tuned separately. One learning step consists of forward pass and backward pass.

Forward Pass Vector $\boldsymbol{k}$ of parameters k_{ij} is tuned during this pass. Suppose m inputs of ANFIS and n neurons in the third layer (number of rules) now, instead 2 inputs and 4 rules shown in Fig. 1a. Then vector $\boldsymbol{k}$ is $n(1 + m)$ vector of parameters of the fifth layer.

$$\boldsymbol{k} = \left[k_{10}\ k_{11}\ k_{12}\ \ldots\ k_{1m}\ k_{20}\ k_{21}\ k_{22}\ \ldots\ k_{2m}\ \ldots\ k_{n0}\ k_{n1}\ k_{n2}\ \ldots\ k_{nm}\right]^T$$

Let P be number of training vectors. Then outputs of network create vector $\boldsymbol{y}$ of P elements - each row of $\boldsymbol{y}$ is response of ANFIS to the one input vector. We can write

$$\boldsymbol{y} = \mathbf{A}\boldsymbol{k} \tag{7}$$

where $\mathbf{A}$ is $P \times n(1 + m)$ dimensional matrix which holds network state after the computation of the fourth layer output.

$$\mathbf{A} = \begin{bmatrix} \bar{\mu}_1(1) & \bar{\mu}_1(1)x_1(1) & \cdots & \bar{\mu}_1(1)x_m(1) & \cdots & \bar{\mu}_n(1) & \bar{\mu}_n(1)x_1(1) & \cdots & \bar{\mu}_n(1)x_m(1) \\ \bar{\mu}_1(2) & \bar{\mu}_1(2)x_1(2) & \cdots & \bar{\mu}_1(2)x_m(2) & \cdots & \bar{\mu}_n(2) & \bar{\mu}_n(2)x_1(2) & \cdots & \bar{\mu}_n(2)x_m(2) \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \bar{\mu}_1(p) & \bar{\mu}_1(p)x_1(p) & \cdots & \bar{\mu}_1(p)x_m(p) & \cdots & \bar{\mu}_n(p) & \bar{\mu}_n(p)x_1(p) & \cdots & \bar{\mu}_n(p)x_m(p) \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \bar{\mu}_1(P) & \bar{\mu}_1(P)x_1(P) & \cdots & \bar{\mu}_1(P)x_m(P) & \cdots & \bar{\mu}_n(P) & \bar{\mu}_n(P)x_1(P) & \cdots & \bar{\mu}_n(P)x_m(P) \end{bmatrix}$$

Equation (7) is a matrix notation of Eq. (6) and Eq. (5) for set of training vectors. Let $\boldsymbol{y}_d$ be $P \times 1$ vector of desired outputs.

$$\boldsymbol{y}_d = \left[y_d(1)\ y_d(2)\ \ldots\ y_d(p)\ \ldots\ y_d(P)\right]^T$$

Then error vector $\boldsymbol{e}$ of the network can be defined as follows

$$\boldsymbol{e} = \boldsymbol{y}_d - \boldsymbol{y} \tag{8}$$

Goal of the learning is to find such vector $\boldsymbol{k}$, respectively its estimate $\boldsymbol{k}^*$, for which mean square error defined by Eq. (9) is minimal (zero).

$$\|\boldsymbol{e}\|^2 = \|\boldsymbol{y}_d - \boldsymbol{y}\|^2 = \|\boldsymbol{y}_d - \mathbf{A}\boldsymbol{k}\|^2 \tag{9}$$

Vector $\boldsymbol{k}$ can be computed using Eq. (10) as follows (with respect to Eq. (7)).

$$\boldsymbol{k} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\boldsymbol{y}_d \tag{10}$$

However inverse matrix $(\mathbf{A}^T\mathbf{A})^{-1}$ can be calculated if and only if the matrix $(\mathbf{A}^T\mathbf{A})$ is regular. Unfortunately there is no guarantee of this and that is why we have to compute estimate of vector $\boldsymbol{k}$, denoted $\boldsymbol{k}^*$, using pseudoinverse matrix Eq. (11).

$$\boldsymbol{k}^* = \mathbf{A}^+\boldsymbol{y}_d \tag{11}$$

Symbol $\mathbf{A}^+$ denotes pseudoinverse of matrix $\mathbf{A}$. Finally, vector $\boldsymbol{k}^*$ is vector of estimated parameters of the fifth layer.

Backward Pass During this pass, parameters a_i and c_i are tuned (b_i remains constant). Method is similar to well-known backpropagation method - it uses backward propagation of errors. Error is computed using Eq. (12), where y_d and y denote desired and real output of the network, respectively.

$$E = \frac{1}{2}\left(y_d - y\right)^2 \tag{12}$$

Goal of learning is to minimize error E by changing values of parameters in the second layer. Assume that α_m denotes general parameter of the m-th neuron in the second layer. Then minimization is done according to Eq. (13) by tuning α_m. E should decreases if partial derivative of E with respect to α_m is negative. Symbol κ denotes a learning rate.

$$\Delta\alpha_m = -\kappa\frac{\partial E}{\partial \alpha_m} \tag{13}$$

Using the chain rule and Eq. (13) we can derive Eq. (14), where $n^{(5)}$ is number of neurons in the fifth layer (number of rules), $C^{-1}(m)$ is a set of neurons of the third layer which are connected with m-th neuron in the second layer and $\frac{\partial y_m^{(2)}}{\partial \alpha_m}$ is partial derivative of bell membership function with respect to α_m.

$$\Delta\alpha_m = \kappa\frac{y_d - y^{(6)}}{y_m^{(2)}}\left(\sum_{j\in C^{-1}(m)}\left(y_j^{(5)}\right) - \sum_{j=1}^{n^{(5)}}\left(\frac{y_j^{(5)}}{y_j^{(3)}}y_j^{(4)}\right)\sum_{l\in C^{-1}(m)}\left(y_l^{(3)}\right)\right)\frac{\partial y_m^{(2)}}{\partial \alpha_m} \tag{14}$$

Partial derivative of bell membership function with respect to a_i and c_i are computed using Eq. (15) and Eq. (16).

$$\frac{\partial y_i^{(2)}}{\partial a_i} = y_i^{(2)} \frac{2b_i}{c_i} \left(\frac{y_i^{(1)} - a_i}{c_i} \right)^{2b_i - 1} \tag{15}$$

$$\frac{\partial y_i^{(2)}}{\partial c_i} = \frac{2b_i \left(\frac{y_i^{(1)} - a_i}{c_i} \right)^{2b_i}}{c_i \left(\left(\frac{y_i^{(1)} - a_i}{c_i} \right)^{2b_i} + 1 \right)^2} \tag{16}$$

Finally, values of Δa_i and Δc_i are computed by substitution Eq. (15) and Eq. (16) into Eq. (14).

3 Proposed Modifications of ANFIS

This section describes proposed modifications of ANFIS. After each modification is presented their impact on performance or accuracy is discussed.

3.1 Different Number of Fuzzy Sets for Each Input

Original ANFIS uses the same number of fuzzy sets for each input variable regardless of variable properties. For example if one variable does not change much (e.g. affects output only linearly or similarly) there is no need to has as much fuzzy sets as must have a variable which changes a lot (affects outputs non-linear, high frequency etc.). We can distribute the same amount of fuzzy sets more precisely and put it where they improve accuracy and performance with the same or even less computational time.

This modification brings performance and/or accuracy in comparison with the original ANFIS, depending on the use. The worst case that should happened is that all inputs have the same number of fuzzy sets and it can be easily reached even with this modification.

3.2 Data Normalization

Great method for improve ANFIS performance is data pre-processing. Z-score normalization Eq. (17) is used for this purpose.

$$z = \frac{x - \mu}{\sigma} \tag{17}$$

Symbol x denotes input/output variable, μ denotes mean and σ is standard deviation. Such normalization can be used for every input/output variable but we have to save mean and standard deviation values for each variable to be able to normalize training vectors, testing vectors etc. in the same way and also for reconstructing original value of variable.

Benefit of this modification heavily depends on data properties and it will be discussed in the experiment section.

3.3 Fuzzy Sets Initialization

Parameters a_i, b_i and c_i have to be properly initialized. Good approach is to analyse training data and estimate these parameters. After the z-score normalization, minimum and maximum of each input variable is computed, desired fuzzy count for variable is divided uniformly from minimum to maximum. Parameter b_i is a constant (e.g. $b_i = 2$ for reasonable membership function shape), parameters a_i and c_i are computed using Eq. (18).

$$step = \frac{max - min}{n} \quad a_i = min + \frac{step}{2} + i \cdot step \quad c_i = \frac{step}{2} \tag{18}$$

Symbols *min* and *max* denote minimum and maximum of input values respectively, n is desired number of fuzzy sets.

This modification has main impact on performance. Proper fuzzy sets initialization can significantly decrease learning time on the other hand bad fuzzy sets initialization increases learning time because learning process start from point far from the solution. Accuracy is affected negligibly because it should converges to the almost same solution as well but it takes more time.

3.4 Changing Learning Rate κ

We borrowed this modification from article [3]. Idea is to modify learning rate in order to improve performance. Symbol κ in Eq. (14) is computed using Eq. (19).

$$\kappa = \frac{\eta}{\sqrt{\left(\frac{\partial E}{\partial a_i}\right)^2 + \left(\frac{\partial E}{\partial c_i}\right)^2}} \tag{19}$$

Symbol κ denotes new learning rate, η is old learning rate. η is usually initialized from interval 0.001 to 0.1. Main advantage is that new learning rate doesn't depend on input data as much as original κ. This modification also brings possibility to tune η depending on success of error minimization. If network error decreases then learning rate increases, on the other hand if error oscillates learning rate decreases, finally if error increases there is no change of learning rate. η is increased by 5 % after 4 consecutive iterations which decrease error. If error oscillates (up, down, up, down) during 4 consecutive iterations η is decreased by 10 %.

Described modification may slightly decrease accuracy in favour to significant performance improvement.

4 Experiments

Two problems has been chosen to demonstrate how proposed system works - function approximation and controlling of a system.

At the first, some metrics to measure performance and accuracy are described.

Performance can be divided into two aspects:

– number of learning iterations (forward and backward pass),
– time that ANFIS needs to converge (to stop learning).

Accuracy is measured as difference between ANFIS output $y(p)$ and desired output $y_d(p)$ for training vector $p \in P$ using E or EA from Eq. (20).

$$E(P) = \frac{\sum\limits_{p \in P} \left(y(p) - y_d(p) \right)^2}{|P|} \quad EA(P) = \frac{\sum\limits_{p \in P} \left| y(p) - y_d(p) \right|}{|P|} \tag{20}$$

4.1 Function Approximation

Two non-linear functions and their modifications have been chosen to demonstrate performance and accuracy of proposed ANFIS. Both functions are uniformly sampled in each axis in interval $< -5, 5 >$ with 0.1 steps.

The first function is two variables *sinc* function Eq. (21). This function is symmetric in terms of input variable so it is appropriate to have same number of fuzzy sets for each input. In this experiment 3 fuzzy sets are used. To demonstrate efficiency of different number of fuzzy sets for each input, function *sincm* Eq. (21) is chosen and 4 fuzzy sets are used for input variable x, only 3 fuzzy sets are used for y.

$$sinc(x,y) = \frac{\sin x}{x} + \frac{\sin y}{y} \quad sincm(x,y) = \frac{\sin 2x}{x} + \frac{\sin y}{y} \tag{21}$$

To verify and confirm results another function *f* and its modified version *fm* have been chosen Eq. (22). In case of *f*, 3 fuzzy sets are used for *x* and *y* while in case of *fm*, 4 fuzzy sets are used for *x* and 3 for *y* only.

$$f\,(x,y) = \sin^2 x \cdot \cos y \quad fm\,(x,y) = \sin^2 x \cdot \cos \frac{y}{2} \tag{22}$$

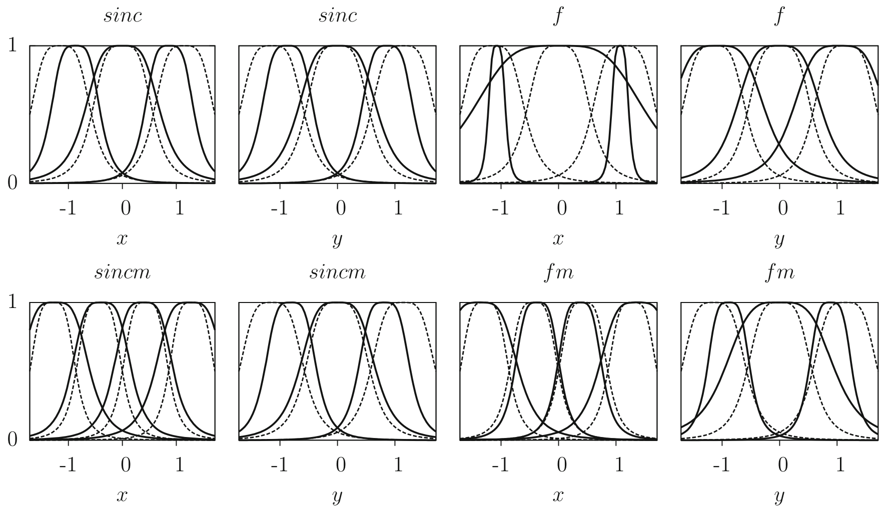


Fig. 2 Membership function after initialization and after learning.

Fuzzy sets after initialization are shown[1] in Fig. 2 by dashed line and solid line is used to represent fuzzy sets after learning. Corresponding sets have same colours (shade of black). It can be seen that there are noticeable differences between fuzzy sets of original (*sinc*, *f*) and modified (*sincm*, *fm*) functions.

Performance and accuracy of this experiments are shown in Table 1. To avoid impact of initial learning rate several η are chosen $\{\eta_1 = 5 \cdot 10^{-4}, \eta_2 = 1 \cdot 10^{-3}, \eta_3 = 5 \cdot 10^{-3}, \eta_4 = 1 \cdot 10^{-2}, \eta_5 = 3 \cdot 10^{-2}, \eta_6 = 5 \cdot 10^{-2}\}$. Table is organized as follows: Each experiment is on two rows. The first one contains used ANFIS configuration[2] and number of iterations for each η while the second row contains accuracy[3] and time in seconds.[4]

[1]Values on both axes are normalized by z-score. So they don't directly correspond to the values of *sinc*.

[2]OrigAB is used for original ANFIS with A fuzzy sets for *x* and B for *y* while ModAB means our modified ANFIS with A fuzzy sets for *x* and B for *y*.

[3]Accuracy means maximal error computed using E from Eq. (20).

[4]Time is measured on our testing machine with Intel Core i5-2540M.

Table 1 Number of learning iterations and consumed time

	η_1	η_2	η_3	η_4	η_5	η_6	Σ
Orig33	210	159	65	39	16	11	**500**
10^{-3}	4.72	3.43	1.47	0.90	0.42	0.41	**11.34**
Mod33	134	92	29	17	7	5	**284**
10^{-3}	3.03	2.06	0.65	0.38	0.17	0.12	**6.42**
Orig33	266	212	103	66	37	41	**725**
10^{-4}	6.48	4.59	2.28	1.46	0.86	1.04	**16.71**
Mod33	184	135	51	38	50	28	**486**
10^{-4}	4.11	2.96	1.14	0.86	1.12	0.62	**10.79**

(a) sinc

	η_1	η_2	η_3	η_4	η_5	η_6	Σ
Orig33	292	237	122	82	38	26	**797**
$4 \cdot 10^{-3}$	6.98	5.51	2.85	1.88	0.94	0.84	**19.00**
Mod33	208	158	64	39	17	11	**497**
$4 \cdot 10^{-3}$	4.94	3.65	1.50	0.92	0.41	0.26	**11.68**

(b) f

	η_1	η_2	η_3	η_4	η_5	η_6	Σ
Orig33	177	129	47	28	11	7	**399**
$5 \cdot 10^{-2}$	4.09	2.81	1.15	0.67	0.32	0.25	**9.30**
Orig44	254	201	103	81	63	65	**767**
$2 \cdot 10^{-4}$	12.64	9.36	4.74	3.69	2.87	2.99	**36.29**
Mod43	192	143	83	87	99	115	**719**
$2 \cdot 10^{-4}$	6.51	4.79	2.76	2.87	3.32	3.89	**24.15**

(c) sincm

	η_1	η_2	η_3	η_4	η_5	η_6	Σ
Orig33	251	198	91	58	25	16	**639**
$5 \cdot 10^{-2}$	5.79	4.24	2.02	1.38	0.61	0.53	**14.58**
Mod34	168	121	43	25	10	7	**374**
$2 \cdot 10^{-4}$	5.49	3.95	1.42	0.86	0.36	0.26	**12.33**
Orig44	203	153	61	37	15	10	**479**
$2 \cdot 10^{-4}$	9.83	6.88	2.75	1.67	0.71	0.48	**22.31**
Mod43	221	169	72	45	21	14	**542**
$2 \cdot 10^{-4}$	7.20	6.25	2.37	1.46	0.69	0.48	**18.45**

(d) fm

It can be seen in Table 1 that modified ANFIS works well. Function *sinc* Table 2a is learned in about 76.1 % less iterations and 76.8 % less time with error below 10^{-3} in average, about 50 % of iterations and time are saved with error below 10^{-4}. With *sincm* Table 2c modified ANFIS uses only about 6.7 % less iterations but time is reduced about 50.3 % with the same accuracy $2 \cdot 10^{-4}$. Function *f* Table 2b, ANFIS used in about 60.4 % less iterations and 62.6 % less time with same accuracy. Finally, using *fm* iterations are about 11.6 % more but time is reduced about 21.0 %. So proposed modifications save time and iterations from 7 % to nearly 80 %.

4.2 Controlling Discrete System

The second experiment is controlling discrete system Eq. (23) which has been presented in literature [4]. This system has one state variable y and one input u.

$$y(k+1) = \frac{y(k) \cdot u(k)}{1 + y(k)^2} - \tan(u(k)) \tag{23}$$

Goal of controlling is to produce such input u that force state variable y (in this case the same variable as output variable) to track desired trajectory given by Eq. (24).

$$y_d(k) = 0.6 \sin\left(\frac{2\pi k}{250}\right) + 0.2 \sin\left(\frac{2\pi k}{50}\right) \tag{24}$$

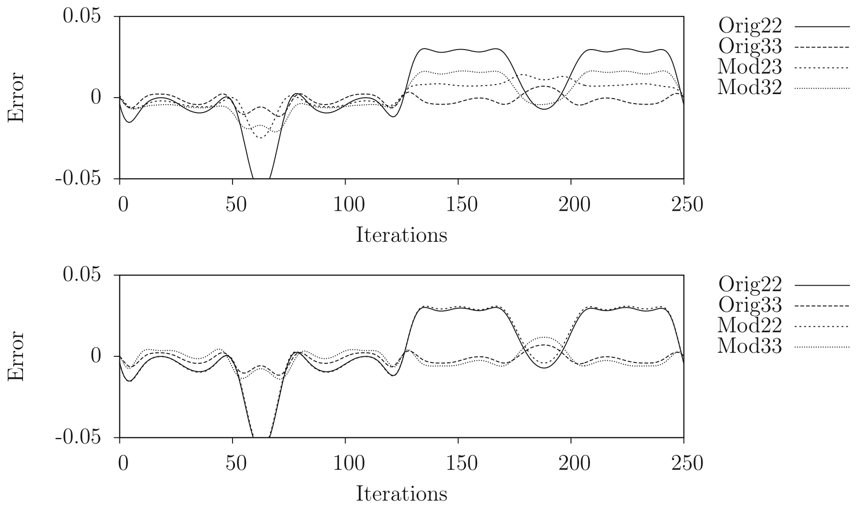


Fig. 3 Comparison of original and proposed ANFIS.

Table 2 Average error

	Orig22	Orig33	Mod22	Mod23	Mod32	Mod33
Average error	0.0161	0.0023	0.0162	0.0058	0.0097	0.0021

Inverse learning [4] is used to generate training data and control the system. Size of training vector is 100 samples, initial learning rate $\eta = 5 \cdot 10^{-3}$. Training environment consists of follows: $y(0) = 0$ and action $u(k)$ is generated randomly each step from range $< -1, 1 >$ using uniform distribution. Training vectors are $\left[y(k), y(k+1); u(k)\right]$.

Accuracy of controlling is shown in Fig. 3. It can be seen that error of proposed ANFIS is slightly lower than error of original ANFIS. Main improvement comes from possibility to have different number of fuzzy sets and make ANFIS to fit your needs - accuracy vs. performance. Exact results can bee seen in Table 2 computed using EA from Eq. (20).

5 Conclusion

This paper presented modification of ANFIS. Main goal was to improve performance and to get more accurate results in reasonable time and it has been reached. Proposed modification relies on these changes: number of fuzzy sets for each input may differ, data normalization is added and intelligent fuzzy sets initialization is used, also changing learning rate from original Jang article [3] is borrowed.

To verify benefits of modification two experiments were done. The first one was function approximation and the second one was controlling of a discrete dynamic system. In both cases modified ANFIS had better performance and/or accuracy. Accuracy improvement is not significant in all experiments but the lower number of iterations and less computational time also have to be counted in. Improvement of accuracy and/or performance heavily depends on solved problem and training data. These experiments verified that proposed modification can be successfully used to improve ANFIS performance and/or accuracy.

Further research could be done in way of tuning parameters. For example extended Kalman filter [9] can be used to speed up learning.

Acknowledgments 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 Reliability and Security in IT project (FIT-S-14-2486).

References

1. Aliev, R.A., Aliev, R.R.: Soft Computing and its Applications. World Scientific Publishing Co., Pte. Ltd. (2001)
2. Boyacioglu, M.A., Avci, D.: An adaptive network-based fuzzy inference system (anfis) for the prediction of stock market return: the case of the istanbul stock exchange. Expert Syst. Appl. **37**(12), 7908–7912 (2010)
3. Jang, J.-S.R.: Adaptive-network-based fuzzy inference system. (1993)
4. Jang, J.-S.R., Sun, C.-T., Mizutani, E.: Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Prentice Hall (1997)
5. Liu, Y., Chen, Z., Xue, D., Xu, X.: Real-time controlling of inverted pendulum by fuzzy logic. In: IEEE International Conference on Automation and Logistics, 2009. ICAL '09, pp. 1180–1183 (2009)
6. Negnevitsky, M.: Artificial Intelligence: A Guide to Intelligent Systems. Addison-Wesley (2002)
7. Pedrycz, W., Gomide, F.: An Introduction to Fuzzy Sets: Analysis and Design. The MIT Press (1998)
8. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson (2010)
9. Simon, D.: Training fuzzy systems with the extended kalman filter. Fuzzy Sets and Systems (2002). http://academic.csuohio.edu/simond/fuzzyopt/fss.pdf

Values and Bayesian Probabilities of Mental States from BSDT PL Analysis of Memory ROCs

Petro Gopych and Ivan Gopych

Abstract BSDT PL is a binary (using the binary signal detection theory, BSDT) version of primary language, PL, an extension of mathematics by the hypothesis of concurrent infinity: all things of the world are describable by one-way infinite binary strings with one-way infinite common beginning. Together with its phenomenology formalization, BSDT PL enables descriptions of any life and mind phenomena with the rigor of mathematics. In this paper, BSDT PL knowledge and knowledge-related notions are defined and applied to designing BSDT PL semi-representational memory for words. By BSDT PL fitting of memory ROC curves measured in healthy humans and patients with injured brains, the values and Bayesian probabilities of mental states serving verbal memory's items are found. BSDT PL measure of subjectivity of word recognition mechanisms is proposed and numerically estimated. It is demonstrated that statistical learning and Bayesian learning are the constituents of cognitive learning. Applications are briefly discussed.

Keywords Concurrent infinity • Subjectivity • Semi-representational memory • Neural network • Object • Symbol • Knowledge • Cognitive learning

1 Introduction

Statistical learning, e.g., [1] and Bayesian leaning, e.g., [2] are unable to cope with problems of learning from a single example and learning to process meaningful messages. But these problems are solvable by recent cognitive learning [3].

P. Gopych (✉)
Universal Power Systems USA-Ukraine LLC, 3 Kotsarskaya St., Kharkiv 61012, Ukraine
e-mail: pmgopych@gmail.com

I. Gopych
Kharkiv Regional Clinical Oncology Centre, 4 Lisoparkivs'ka St., Kharkiv 61070, Ukraine
e-mail: gipgip80@gmail.com

The discreteness of active agents' descriptions ensures their ability to learn from a single example [3, 4]. To provide their ability to process meaningful messages, a strict theory of meaning and meaningful communication is needed. Since traditional mathematics ignores meanings by its very essence, it was done by an extension of its axiomatic by the hypothesis of *concurrent infinity* and its *phenomenology formalization* [5]. Resulting *primary language*, PL, and its binary (using the binary signal detection theory, BSDT [4]) version, BSDT PL, provide a possibility to describe meanings and other subjective phenomena, to design algorithms/devices for processing meaningful messages [5], and to account for data measured in cognitive (where meanings are essential) experiments [3, 6].

In this paper, BSDT PL *theory of knowledge* or *epistemology* has been developed and used to design a semi-representational memory for meaningful messages, e.g., words and a theory of communication by meaningful messages, e.g., words. Complete quantitative BSDT PL description of memory ROCs (receiver operating characteristics) measured in healthy humans and patients with specific brain injuries [7] has been achieved (see [3] for fitting methodology and complementary results) and *everything formal* that can ever be known from memory ROC measurements *about subjectivity* of mechanisms of recall/recognition of meaningful words has been extracted. A person's capacity to discern features of things, his/her cognitive capacity, the values and Bayesian probabilities of his/her memory states have quantitatively been found. It has been demonstrated that statistical learning [1] and Bayesian learning [2] are *independent but complementary* mechanisms constituting together the cognitive learning [3].

2 BSDT PL and Phenomenology Formalization

Concurrent infinity adds to the axiomatic of traditional mathematics the concept of *coevolution* [5] and the *principle of sufficient reason* ("a thing cannot occur without a cause which produces it" [8, p. 3]). It postulates the possibility of describing the things by their unique *PL strings*, $c_{xi}x_j^i$ – *infinite on semi-axes* binary strings with an *infinite on a semi-axis common* beginning. PL string's affix $x_j^i \in S_{xi}$ is its i-bit *PL name* (one of 2^i binary vectors with spin-like components $\pm$ 1 constituting an i-dimensional binary vector space S_{xi}); its one-way infinite beginning of the length of $\aleph_0$ bits, c_{xi}, is *PL name's context* ($\aleph_0$ is Cantor's aleph naught, the totality of natural numbers). All PL strings of the same infinite length (of the same *meaning or evolutionary complexity*) constitute together an ultimate or proper class S_{cx0} – the set that cannot be a subset of any other set; the amount of $c_{xi}x_j^i \in S_{cx0}$ is also infinite but countable, $\aleph_0$. PL names x_j^i of PL strings $c_{xi}x_j^i$ name/enumerate all things of the world, known as well unknown but conceivable. The totality of PL strings is complete and consistent; Gödel's incompleteness holds for the totality of x_j^i taken without their contexts or for its infinite fractions only. That is, primary language, PL, is mathematics of PL strings; PL computations are performed by super-Turing machines. If the arrangement of PL strings with their one-way infinite common beginning is destroyed then the PL transforms back into traditional mathematics [5].

Concurrent infinity leads simultaneously to *phenomenology formalization*. It means it is postulated the *literary equivalence* of such usually incommensurable things as a *one-way infinite binary PL string* $c_{xi}x_j^i$, *meaning* of the x_j^i, the *real-world physical device* devoted to recognize the meaning of x_j^i and *feeling* ("quale" or primary thought) it causes. Of this follows that PL strings $c_{xi}x_j^i$ define meanings of their PL names x_j^i or subjective/first-person experiences or psychological (momentary internal) states of the agent who is the owner of devices recognizing the x_j^i [5].

BSDT [4] is the best technique for computations with binary strings/PL names x_j^i. For this reason, the version of the PL employing BSDT coding/decoding algorithms is most convenient for given the infinite common context PL processing of PL names; we refer to it as BSDT PL. BSDT abstract selectional machines, ASMs [9], implement its best algorithms; they have some different but computationally equivalent (Hamming distance, convolutional, neural network) forms and represent BSDT's *universal neural network computational units* [4, 6, 9].

3 Elements of BSDT PL Epistemology

BSDT PL allows us to develop epistemology as a *quantitative* discipline.

3.1 BSDT PL Things, Living Organisms, Objects, and Their Features

A thing, say $\mathcal{Y}_k^i = c_{yi}y_k^i$, is anything of the world that is different from anything else [5]. Each of them can be considered as an infinitely deep hierarchy of other things evolutionary embedded into the first one [10]. Since the thing's properties or *features* are given by its embedded things, their amount is *infinite but countable* [5, 10].

A living organism, say an animal or a human, is a thing that is able to change itself to self-sustain and self-reproduce itself in its permanently changing environment [10]. To be able to adapt to it, an organism must be able to detect, discern, classify, memorize and utilize those environmental features that are essential for its aim. We refer to the organism's part classifying and memorizing these features as its *memory system* or its *neural space* [6]. The organism's part detecting environmental signals and discerning specific patterns of them is called the organism's *sensory system*; it consists of sensors and sensory pathways and generates the neural space's inputs. The organism's part that could transform the neural space's outputs into the organism's actions is called its *executory system*; it consists of behavioral pathways and behavioral devices/actuators that directly influence the environment. The neural space (memory as a whole) is divided into (consist of) numerous neural *sub*spaces (memory items), each of which, say $\mathcal{X}^i_k$, represents a *finite* fraction of the thing's features.

An object is a representation of a thing by the organism's neural subspace $\mathcal{X}^i_k = c_{xi}x^i_k$ or the meaning of the subspace's PL name x^i_k (a binary code the $\mathcal{X}^i_k$ is devoted to process [5]). The object specifies, see (1), a *finite* set of *i* of the thing's *distinctive* features but remains unspecified *infinite* number of its other features. The claim "a subspace represents a thing" means it is able to produce self-generated outputs coinciding with outputs it produces when it is stimulated by inputs from the sensory system exposed to signals from the thing. For this reason, if *i* of the thing's features are only considered then the subspace $\mathcal{X}^i_k$ and the object the $\mathcal{X}^i_k$ represents are *equivalent*; otherwise, $\mathcal{X}^i_k$ is simply a thing with infinite number of its features.

Objects reduce infinite number of features of a thing to be perceived to a finite set of them and, in this way, ensure the finiteness of the thing's particular representation. The amount of objects an organism constructs for a thing is as large as the amount of neural subspaces it uses to process its sensory inputs. Separate objects produce together the thing's whole/composite object, with the list of features of its constituents. If a human (researcher) uses an additional man-made apparatus to enhance his/her normal (healthy) sensory system then, with its help, he/she teaches a new his/her neural subspace (his/her new neural subspace learns) to recognize new his/her sensory inputs; in this way, the researcher designs for himself/herself a new the thing's object representing the set of the thing's features that are unavailable to his/her unaided sensory system. The amount of such new objects could be as large as the amount of different experimental set-ups, measurement and data-processing protocols one could design to enhance his/her sensory system, i.e., it is *infinite in principle but restricted at any time* by practical reasons.

3.2 BSDT PL Semi-representational Memory and First-Person Knowledge

The BSDT PL defines an organism's memory or its neural space as a manifold or a "society" of numerous mutually interacting memory items or neural subspaces $\mathcal{X}^i_k = c_{xi}x^i_k$ each of which stores *the only* meaningful memory record, $c_{xi}x^i_k$; that is the reason why BSDT PL learning paradigm *one-memory-trace-per-one-network* [6] should be a property of all organisms, not only humans. To recall/recognize the memory record the $\mathcal{X}^i_k$ stores, it is insufficient to identify binary code x^i_k pointing to or *representing* the $\mathcal{X}^i_k$; it is also needed to *activate the physical body* of $\mathcal{X}^i_k$; that is the reason why BSDT PL neural space is a *semi-representational memory* system [6, 9].

For the *ik*th subspace $\mathcal{X}^i_k = c_{xi}x^i_k$, c_{xi} is a brain structure $\mathcal{C}_{xi}$ that, after its learning, can generate 2^i neural subspaces $\mathcal{X}^i_k$ devoted to process 2^i meaningful messages $c_{xi}x^i_k$ related to the same category of them, $\mathcal{C}_{xi} = c_{xi}x^i$, x^i (x^i is a set of i empty slots to be filled by components of different x^i_k in the course of learning the category's items, see Fig. 8 in [5]). Each x^i_k is represented as a pattern of i spikes or a *symbol* (a certain the subspace's physical response, Sect. 3.3) specified by components of x^i_k, x^i_{kj} ($k = 1, 2, \ldots 2^i$; $j = 1, 2, \ldots\ i$). In other words, c_{xi} is the physical body $\mathcal{C}_{xi}$ of all neural subspaces $\mathcal{X}^i_k$ *before* they were learned to process different x^i_k and $\mathcal{X}^i_k$ is a fraction of $\mathcal{C}_{xi}$ *after* it was learned to process particular x^i_k. Throughout this paper, cf. [10], we use same upper-case and lower-case letters to designate real-world things (e.g., $\mathcal{C}_{xi}$ or $\mathcal{X}^i_k$) and their symbolic descriptions/names (e.g., c_{xi} or x^i_k).

The dimensionality i of PL name x^i_k of PL string $c_{xi}x^i_k$ defines *the total amount*, it is also i, of the object's features (respective thing's distinctive features) the subspace $\mathcal{X}^i_k$ is able to recognize; these features distinguish the object the $\mathcal{X}^i_k$ represents from other objects of the same category the organism *knows* (its memory stores/represents). The *j*th component of x^i_k, x^i_{kj} (it is physically implemented as a spike [5]), points to the *j*th object's feature and to its real-brain implementation (the order of numbering these features does not matter [5]); the sign of x^i_{kj} defines whether the presence $\left(x^i_{kj} = +1\right)$ or the absence $\left(x^i_{kj} = -1\right)$ of the *j*th feature specifies the object the $\mathcal{X}^i_k$ represents (spikes targeted to excitatory and inhibitory synapses give positive and negative values of x^i_{kj}, respectively). We refer to the *ik*th *meaningful* binary vector x^i_k, PL name of PL string $c_{xi}x^i_k$ interpreted as described, as given the context *PL knowledge* or *first-person knowledge* about the object the $\mathcal{X}^i_k$ represents or about its meaning $c_{xi}x^i_k$. "Given the context" means not only given the body of neural subspace $\mathcal{X}^i_k$ but also given specific sensors (enhanced by additional apparatus or not) and sensory pathways; it is because the organism's sensory system prepares inputs to all the organism's neural subspaces. Hence, PL knowledge is a *real-brain* or *first-person* meaningful symbolic (using specific the subspace's output, cf. Sect. 3.3) representation of the thing's distinctive features by the organism's particular neural subspace. First-person knowledge is essentially *subjective* and directly available to the organism itself only.

3.3 Symbols, Knowledge Understanding, and Third-Person Knowledge

Let the $\mathcal{Y}^i_k = c_{yi}y^i_k$ be an arbitrary in general but always the same environmental thing, which is called a *symbol* (e.g., a behavioral action/gesture), produced by the organism's executory system in response to the output x^i_k of its neural subspace $\mathcal{X}^i_k$. If it is, then y^i_k is a *third-person* (directed to others) meaningful symbolic representation of first-person knowledge x_k^i. Let another organism be *a copy* of the first one (has the same anatomy and developed in the same environment [10]). If $\mathcal{Y}_k^i$ is able to activate another organism's sensors feeding its subspace $\mathcal{X}_k^i$ then everything formal or symbolic that these sensors generate from their current inputs is $\check{z}_k^i$, another organism's initial response to $\mathcal{Y}_k^i$. Since $\check{z}_k^i$ is "noised" by signals from other things, another organism's sensory pathways rectify it and transform $\check{z}_k^i$ into $\check{z}_k^i$, an input to neural subspace $\mathcal{X}_k^i$ (the same as for the first organism). If z_k^i is identified by another organism's $\mathcal{X}_k^i$ as x_k^i (the same as for the first organism) then another organism identifies meanings of links of the chain $\check{z}_k^i \rightarrow z_k^i \rightarrow x_k^i$ as the meaning of PL name y_k^i of the symbol $\mathcal{Y}_k^i$. It means another organism stimulated by the symbol $\mathcal{Y}_k^i$ regenerates in its subspace $\mathcal{X}_k^i$ the meaningful record $c_{xi}x_k^i$ the subspace $\mathcal{X}_k^i$ of the first organism stores. In such a case, we say that communicating organisms *understand* the meaning $c_{yi}y_k^i$ of the symbol $\mathcal{Y}_k^i$ in the same way or, briefly, *understand* each other.

Mutual understanding of symbols is only possible if communicators are copies or "mirror" replicas of each other [5], i.e., if they have equivalent sensory/executory systems and neural subspaces. It means, in particular, an understanding takes place if communicating organisms have such a neural subspace that is active when they *produce or only perceive* the symbol. This BSDT PL prediction [5] is directly supported by discovering "mirror neurons" (they are active when an animal/human performs an action or only observes it, e.g., [11] and [12]) and by discovering "secrete-and-sense cells" (they secrete and sense the same signaling molecule by same secretion reporters and sensory receptors [13]). Another direct validation of this prediction is given by empirical finding of spatially and temporally coherent brain activity in humans mutually understanding meaningful messages of each other [14].

Third-person knowledge is such a *symbolic* representation of an organism's first-person knowledge that other organisms (and the organism itself) *are able to perceive and understand*. Any third-person knowledge could be presented by many different arrangements of symbols (organism's knowledge-specific actions) but, since it is understandable by others, it is always reducible to its same first-person form.

3.4 Knowledge Conditioning, Legacies of Evolution and Development

First-person knowledge is a mix of an organism's *internal* symbolism (binary strings implemented as patterns of spikes) and its *internal* reality (real-brain devices devoted to process these binary strings); its meaning is always *certain* (unconditional), complete, unique, unambiguous and directly perceivable by (available to) the organism itself only (these features originate in PL phenomenology formalization and the uniqueness of each of PL strings [5]). Third-person knowledge is a kind of *environmental* things and its meaning is by definition *conditioned* because it has only a sense with respect to particular organism, its first-person particular knowledge, and particular mechanisms of its first-to-third-person and third-to-first-person knowledge transformations (Sect. 3.3).

PL equation (cf. [10]) describing the *ik*th neural subspace (item of memory) $\mathcal{X}_k^i$ is

$$\mathcal{X}_k^i = c_{xi}x_k^i = \mathcal{E}_{mn}\mathcal{F}_n^m\left(x_k^i\right) = \mathcal{E}_{mn}c_{fm}\left(x_k^i\right)f_n^m \tag{1}$$

where $\mathcal{F}_n^m\left(x_k^i\right)$ are brain representations of the subspace's features distinguishing the object the $\mathcal{X}_k^i$ represents from other objects of the same category of them; PL strings $c_{fm}\left(x_k^i\right)f_n^m$ give meanings of $\mathcal{F}_n^m\left(x_k^i\right)$; the sign $\mathcal{E}_{mn}$ designates PL-specific operation of *evolutionary embedding* of parts of a whole into the whole itself [10]. The features $\mathcal{F}_n^m\left(x_k^i\right)$ are also *distinctive* features of the thing the $\mathcal{X}_k^i$ represents. Pairs of indices i and k ($k = 1, 2, \ldots, 2^i$), m and n ($n = 1, 2, \ldots, 2^m$; $2^m \geq i$) enumerate categories of objects/subspaces and features of these objects/subspaces [5, 10].

PL Eq. 1 reflects the equivalence of the subspace/object as a whole $\mathcal{X}_k^i = c_{xi}x_k^i$ (i of features of $\mathcal{X}_k^i$ are indicated but unspecified) and the subspace/object as an embedded set $\mathcal{E}_{mn}\mathcal{F}_n^m\left(x_k^i\right) = \mathcal{E}_{mn}c_{fm}\left(x_k^i\right)f_n^m$ of i of its features $\mathcal{F}_n^m\left(x_k^i\right)$ completely specified by PL strings $c_{fm}\left(x_k^i\right)f_n^m$. Given their contexts, x_k^i and f_n^m provide first-person knowledge of the $\mathcal{X}_k^i$ (or of the object the $\mathcal{X}_k^i$ represents) and of each of its features $\mathcal{F}_n^m\left(x_k^i\right)$, respectively. If contexts c_{xi} and $c_{fm}\left(x_k^i\right)$ are further unspecified then $\mathcal{X}_k^i$ is simultaneously the neural space and the object it represents.

For an organism, PL strings $c_{fm}\left(x_k^i\right)f_n^m$ represent complete histories of evolutionary design and developmental elaboration of features $\mathcal{F}_n^m\left(x_k^i\right)$ [10] of the subspace/object $\mathcal{X}_k^i$. Their contexts and PL names, $c_{fm}\left(x_k^i\right)$ and f_n^m, give the legacy of all the organism's ancestors (of its *evolution*) and the legacy of all the organism's individual life experiences (of its *development*), respectively; the former is fixed as the anatomy of the organism's body and in general remains the same over its life, the latter is fixed as an anatomy-specific easy-to-change symbolic features of the organism's body and in general is changeable over its life. Here, it does not matter in which way it was actually done; it is only important that, since life experience is accumulated over the organism's life, PL strings $c_{fm}\left(x_k^i\right)f_n^m$, in addition to their m-bit affixes f_n^m, could have in general different in length explicitly specified binary *fractions* of their contexts, providing BSDT PL descriptions of this experience or items of the organism's first-person knowledge about the features $\mathcal{F}_n^m\left(x_k^i\right)$. Normally, in an act of meaningful communication, organisms generate and perceive symbols that bear no information about specific form of first-person knowledge items initiated their appearance. To observe them directly, special efforts are required (Sect. 4).

3.5 The Significance of First-Person Knowledge and the Probability of Its Use

For certainty, let us consider a memory for meaningful words studied by rating measurements of memory ROCs in humans, e.g., [7]. For brevity, let us restrict ourselves by considering the only word $\mathcal{Y}^i_k$ that, given the context c_{yi}, bears third-person knowledge item y^i_k; we will refer to this word, its PL name and its context as $\mathcal{Y}$, y and c_y, respectively. To better comply with notations of the work [3], let us refer to the dimension of y^i_k and other participating vectors as N. It means, in what follows, $\mathcal{Y}$, y, c_y, $\mathcal{X}$, x, c_x, N, etc. are respectively understood as particular samples of $\mathcal{Y}^i_k$, y^i_k, c_{yi}, $\mathcal{X}^i_k$, x^i_k, c_{xi}, i, etc. Thanks to such simplified notations, (1) takes the form

$$\mathcal{X} = c_x x = \mathcal{E}_{mn}\mathcal{F}^m_n(x) = \mathcal{E}_{mn}c_{fm}(x)f^m_n = \mathcal{E}_r\mathcal{F}_r = \mathcal{E}_r c_{vr}v_r \tag{2}$$

where $\mathcal{F}^m_n(x) = \mathcal{F}_r$ are N real-brain representations of features distinguishing the neural subspace/object $\mathcal{X} = c_x x$ and the word $\mathcal{Y}$ it represents; PL strings $c_{fm}(x)f^m_n = c_{vr}v_r$ are meanings of $\mathcal{F}^m_n(x) = \mathcal{F}_r$; indices $r(r = 1, 2, \ldots, N;\ N \le 2^m)$ are PL names f^m_n presented as successive natural numbers [5]. Each PL string $c_{vr}v_r$ consists of a one-way infinite substring $c_{vr} = c^u_{mn}(x)$ and a finite substring $c_{vr} = c^s_{mn}(x)f^m_n$; the former is an initial fraction of $c_{fm}(x)$ given as a part of physical body of $\mathcal{X}$, the latter is a concatenation of PL name f^m_n and everything symbolic in $c_{fm}(x)$, $c^{\,s}_{mn}(x)$; in the history of designing the $\mathcal{F}^m_n(x)$, $c_{fm}(x) = c^u_{mn}(x)c^{\,s}_{mn}(x)$, $c^{\,u}_{cm}(x)$ and $c^{\,s}_{mn}$ represent contributions of human evolution and human individual development, respectively.

We understand the length of v_r in bits, V_r, as *a measure of significance* of the feature $\mathcal{F}_r$ of the word $\mathcal{Y}$ represented by the subspace $\mathcal{X}$ – the amount of information about $\mathcal{F}_r$ extracted from an individual's life experience and stored in memory. It reflects the importance of $\mathcal{F}_r$ for discovering the $\mathcal{Y}$ among other similarly presented words: the more the V_r the more the significance of $\mathcal{F}_r$ is and, consequently, the more the chance that particular individual will use $\mathcal{F}_r$ to identify $\mathcal{Y}$. Since V_r is related to a chance of employing the $\mathcal{F}_r$ for an identification of $\mathcal{Y}$, *after a normalization*, it can also be treated as *a probability* of the use of $\mathcal{F}_r$ in different acts of decoding the $\mathcal{Y}$. Consequently, the significance of $\mathcal{F}_r$ and the probability of its use are given by the same but *differently normalized* (Sect. 4.2) parameter V_r. If $V_r = V_t$ at all $r \ne t$ ($r = 1, 2, \ldots, N$ and $t = 1, 2, \ldots, N$) then, in different appearances of $\mathcal{Y}$, all $\mathcal{F}_r$ are used for the identification of $\mathcal{Y}$ with the same probability. If at some $r \ne t$ some $V_r \ne V_t$ then, in different identifications of $\mathcal{Y}$, different $\mathcal{F}_r$ are used with different probabilities.

4 Numerical Validation of BSDT PL Epistemology

4.1 First-Person Knowledge Manifestation in BSDT PL ROCs and BDPs

First-person knowledge $c_{vr}v_r$ about the *r*th feature $\mathcal{F}_r$ of the word $\mathcal{Y} = c_y y$ represented by the subspace $\mathcal{X} = c_x x$ is directly quantified by the probability of its use, V_r; first-person knowledge about the $\mathcal{Y}$ as a whole is quantified by the set of N probabilities V_r we understand as components of a string/vector V. In a separate communication act, neither the whole V nor its separate V_r are observable (Sect. 3.4); they can only be detectable in *multiple* acts of recall/recognition of $\mathcal{Y}$. The subspace $\mathcal{X}$, the best BSDT PL decoding device [4, 6, 9], compares the amount of $\mathcal{F}_r$ available in a current identification event, $H(x,z)$, with a threshold defining the confidence of decisions, θ: if $H(x,z) \geq \theta$, then z is interpreted as a knowledge of $\mathcal{Y}$ [3, 4, 6]. Here, $H(x,z) = j$ with $0 \leq H_0 \leq j \leq N$ is Hamming distance between the vector x (it is fixed, its N components x_r point to $\mathcal{F}_r$ the $\mathcal{X}$ stores) and a vector z (it is changeable, its N components z_r point to $\mathcal{F}_r$ found by sensory system for the current identification event); H_0 is the amount of $\mathcal{F}_r$ that are certainly known ($0 \leq H_0 \leq N$; given the state of $\mathcal{X}$ and the health of sensory system, H_0 depends on the state of the environment only); θ is separately chosen for each decoding event [3, 4, 6, 9].

If all probabilities V_r are the same then all $H(x,z) = j$ are generated with equal probabilities (remember, the order of enumeration of $\mathcal{F}_r$ is not specified, Sect. 3.2) and, consequently, all sets of j of $\mathcal{F}_r$ are equally probable. If, at different r, V_r are not the same then, in different events of decoding the $\mathcal{Y}$, some of $H(x,z)$ become more/less probable/significant. If the significance/probability of an event $H(x,z) = j$ is p_j then, we say, it is the *j*th component of a string/vector p. The p includes also the probability/significance p_0 of an extreme case when everything about $\mathcal{Y}$ is certainly unknown (all features of $\mathcal{Y}$ are unknown/ignored) and the probability/significance p_{N+1} of an extreme case when everything about $\mathcal{Y}$ is certainly known (all features of $\mathcal{Y}$ are taken into account). As a result, N parameters V_r specifying *separate* $\mathcal{F}_r (r = 1, 2, \ldots, N)$ are transformed into $N + 2$ parameters $p_j (j = 0, 1, \ldots, N + 1)$ specifying *separate arrangements of* $\mathcal{F}_r$. It means $H(x, z)$ and $P(N, H, \theta)$, the probability of recall/recognition of $\mathcal{Y}$, should additionally depend on the *fuzzy* parameters V and p, respectively; for this reason, it is more accurately to write $H(x, z, V)$ instead of $H(x, z,)$ and $P(N, H, \theta), p$ instead of $P(N, H, \theta)$; if all V_r and, consequently, all p_j are equal to each other then $H(x, z, V) = H(x, z,)$ and $P(N, H, \theta), p = P(N, H, \theta)$.

Given the environment (the amount of certainly known $\mathcal{F}_r$, H_0), by changing the θ, an individual changes the actual amount $H(x, z,)$ of $\mathcal{F}_r$ used to identify the $\mathcal{Y}$ and, consequently, *his/her probability* $P_H(N, H, \theta), p$ of an identification of $\mathcal{Y}$. Given the confidence of decoding decisions (a threshold θ), by changing the environment (the amount of certainly known $\mathcal{F}_r$, H_0), he/she changes the $H(x, z)$ and *his/her probability* $P_H(N, H, \theta), p$ of an identification of $\mathcal{Y}$. Discrete-valued functions $P_H(N, H, \theta), p$ and $P_H(N, H, \theta), p$ define BSDT PL ROCs and BDPs (basic decoding performance functions) [3, 4, 6]. $P_H(N, H, \theta)$, p can be compared, e.g., [3] with empirical memory ROCs measured in animals/humans, e.g., [7]; $P_H(N, H, \theta)$, p can be compared, e.g., [6] with empirical psychometric functions measured in animals/humans, e.g., [15].

4.2 Values and Bayesian Probabilities of Mental States Found from ROCs

For the healthy sensory system, the perfect subspace $\mathcal{X} = c_x x$, and the same at all r relative significance V_r of features $\mathcal{F}_r$ of the word $\mathcal{Y}$ (Sect. 4.1), BSDT PL correct decoding probability $P(N, H, \theta)$, p can analytically be found [4, 6] as a discrete-valued function $P(N, q, i_\theta)$ of arguments N (the dimensionality of x or the amount of $\mathcal{F}_r$ to be considered), q (the environmental cue [3, 4, 6, 9] or a fraction of certainly known $\mathcal{F}_r$, H_0/N), and j_θ (specific choice of θ for the criterion $H(x, z,) \geq \theta$ or the rate of decision confidence [3, 4, 6, 9]). In general case [3], the measured hit rate or empirically estimated correct decoding probability of $\mathcal{Y}$, P^j_{exp}, is given by

$$P^j_{\mathrm{exp}} = p^j_{\mathrm{exp}} P(N, q, j) \tag{3}$$

where $j = j_\theta$, a measurement parameter. Given the j_θ, estimations p^j_{exp} of all $N + 2$ parameters p_j can be found as described in [3] by fitting empirical memory ROCs, e.g., [7] and the use of (3); examples are shown in Fig. 1. Some other memory parameters found by BSDT PL fitting of same ROCs [3] are given in Table 1 (numerical ROC analysis, e.g., [7] using the traditional signal detection theory gives no information of values and Bayesian probabilities of memory states).

Presenting the P^j_{exp} as *a product* (3) of two factors, p^j_{exp} and $P(N, q, j)$, implies coexistence of two *independent but complementary* components of a compositional word recognition process: (1) discovering a pattern of j of an input's features $\mathcal{F}_r$ and its *preliminary* identification as the object $\mathcal{X}$ representing the $\mathcal{Y}$; (2) weighting the results of the process 1 by N parameters V_r designed for each of $\mathcal{F}_r$ on the basis of human individual life experience and their transformation into the jth experience-specific (motivationally biased) value of p_j defining (e.g., dark bar in Fig. 1H) the *probability* that $\mathcal{X}$ will *in the end* recognize $\mathcal{Y}$ with the rate of confidence j. The process 1 is implemented by sensory and memory systems taken together; the process 2 is implemented by a widely distributed whole-brain *value system* (it is described in, e.g., [16]) defining the significance/importance or cognitive value or, simply, the value of each of the person's mental (e.g., memory) states. We suggest that parameters p_j defined in Sect. 4.1 and extracted, with the help of (3), from memory ROCs (Fig. 1) do give a *quantitative* BSDT PL representation of mental states' values that so far were only *qualitatively* defined, e.g., [16] on the basis of neuroscience arguments.

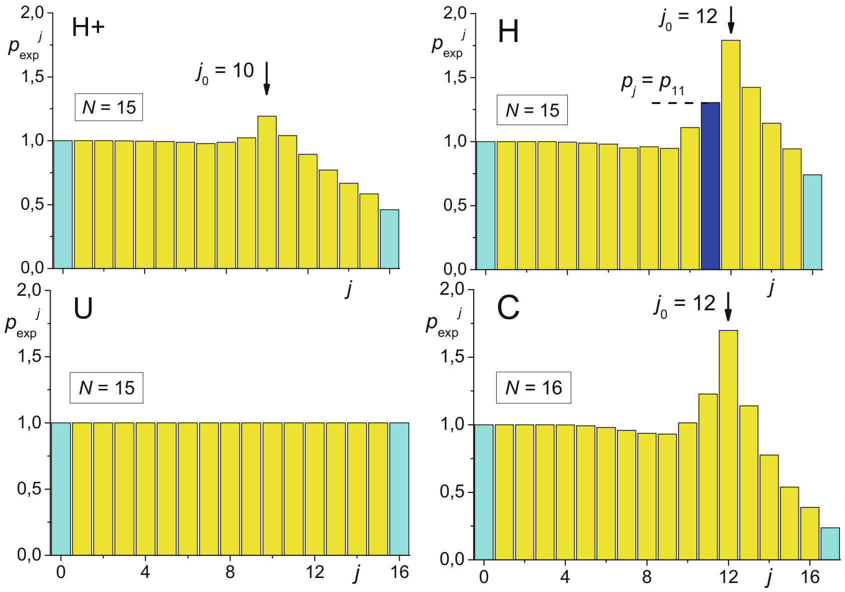


Fig. 1 Estimations p^j_{exp} of (cognitive) values p_j of jth states of the subspace $\mathcal{X} = c_x x$ dealing with j features $\mathcal{F}_r$ of the word $\mathcal{Y}$ in humans learned to recognize the $\mathcal{Y}$ (C, control/healthy subjects, cf. Fig. 2C in [3]; H and H+, patients with injured brains; U, cognitively unlearned/naïve subjects). Estimations of p_0 (the value of overdoubt decisions) and p_{N+1} (the value of overconfident decisions) are shown as left-most and right-most shaded bars, respectively. The height of the dark bar, $p_j/(N+2) = p_{11}/17$, gives, for patients of the H-group, the probability that $\mathcal{Y}$ is recognized with the rate of confidence $j = 11$. Histograms were found by BSDT PL fitting of memory ROCs measured in [7]; magnitudes of N and j_0 are from Table 1

Values p_j of mental (e.g., human memory) states are normalized by (4), a formalization of the fact of certainty of the word to be recognized (Fig. 1U); its mathematically equivalent form is given by (5):

$$\sum p_j = N + 2, \tag{4}$$

$$\sum p_j/(N+2) = 1. \tag{5}$$

Of (4) and (5) follows that values p_j of respective states of $\mathcal{X}$ presented as ratios $p_j/(N+2)$ are simultaneously the *probabilities* of identification of $\mathcal{Y}$ by j of its features $\mathcal{F}_r$. These probabilities are manifestations of *first-person* knowledge items v_r (Sect. 3.5) and, consequently, are essentially *subjective*, as it is for *Bayesian probabilities* [2, 8]. *Prior* Bayesian probabilities are presented in Fig. 1U; their distribution is uniform and reflects, for a person, the absence of his/her previous life experience. Examples of *frequency estimations* $p^j_{\text{exp}}/(N+2)$ of *posterior* Bayesian probabilities are given in Fig. 1C, H, and H+; they were obtained by BSDT PL fitting of memory ROCs measured using *multiple* the word's presentations [7], are non-uniform (have a maximum at $j = j_0$) and reflect a person's capacity to employ the lessons of his/her life for choosing the confidence of his/her recognition decisions. The very fact of existence of a preferred j_0 means a person's ability to persuade a goal (to understand and follow instructions); its specific magnitude (e.g., Table 1) defines his/her preferred (motivationally biased) rate of confidence of identification decisions.

Table 1 Results of fitting empirical memory ROCs [7] for groups C, H, and H+; N, q, and j_0 (arguments of (3)) are from [3]; $\Delta = \left(\sum p^j_{\text{exp}} - N - 2\right)/(N+2)$, relative accuracy of histograms in Fig. 1; $b = \sum \left|p^j_{\text{exp}} - 1\right|$, a measure of their non-uniformity (all $\left|p^j_{\text{exp}} - 1\right| \geq 0$ and $\Delta b = \Delta b$)

Group	N	q	j_0	Δ	$b \pm \Delta b$
C, control subjects	16	$5/16 \approx 0.31$	12	0.066	3.34 ± 0.22
H, hypoxia patients	15	$3/15 = 0.20$	12	0.075	2.27 ± 0.17
H+, infarct patients	15	$2/15 \approx 0.13$	10	0.084	1.94 ± 0.16

We see the process of recall/recognition of words consists of two *independent but complementary* mechanisms: discerning the features and performing their motivational biasing. For this reason, learning to recognize words should also consist of two processes: (1) acquiring the skill of discerning the word's features or *statistical* ("supervised") *learning* [1] and (2) accumulating the experience of making motivationally biased word-recognition decisions or *Bayesian* ("reinforcement") *learning* [2]. It means, in addition to the capacity to discover the features of things, experience-specific *cognitive learning* [3] should take into account a *quantitative* description (e.g., Fig. 1) of a *subjective* experience (feeling) about the importance of things to be recognized. The only quantitative information that can here be available and used is *the distribution of Bayesian probabilities* of mental (memory) states.

4.3 BSDT PL Measure of Subjectivity

Even patients with most severe brain damages [7] produce a non-uniform posterior Bayesian probability distribution (Fig. 1H+) and, consequently, keep their ability to make motivationally biased decisions. This fact motivates our suggestion to use the non-uniformity of posterior probability distribution, $b = \sum \left|p^i_{\text{exp}} - 1\right|$ (see Table 1), as *a measure of subjectivity* of word-recognition mechanisms or, more generally, of cognitive capacity of humans. This idea is illustrated by plotting the b and the q (an argument of (3) and *the capacity to discern the word's features*) as a function of brain damage degree and as functions of each other (Fig. 2).

Magnitudes of b and q depend respectively on the health of a person's value system and the health of his/her sensory and memory systems. Since these systems are widely distributed and overlapped structures, brain injuries should simultaneously damage all of them. It makes of the q (the curvature of ROCs [3, 4]) a measure of damage affecting the b: the more the q the more the b is, and vice versa. Examples of possible extrapolations of empirically found values of $b(q)$, $0 \le q \le 1$, are shown in Fig. 2. Line 2 seems biologically more plausible than line 1; to be conclusive, further research is needed. If $q > 0.31$ (healthy subjects), it is expected a new information of ROC measurements in subjects with intact brains could be obtained.

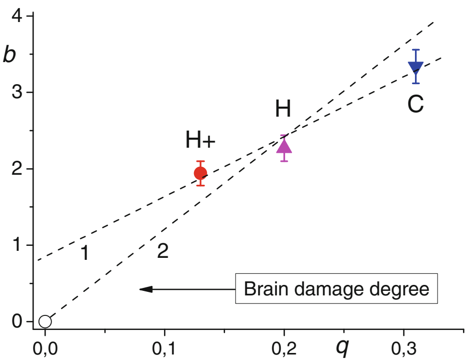


Fig. 2 The capacity of motivational biasing (Bayesian learning) b vs. feature discerning (statistical learning) capacity q. Line 1 gives an extension of the best linear interpolation across three empirical points (filled signs) and line 2 implements the assumption $b(q) = 0$ if $q = 0$ (open circle; simultaneous inability to discern the thing's features and to make motivationally biased decisions). The arrow indicates the degree of brain injury (C, intact brains; H, patients with an intermediate brain injury; H+, most injured brains; data are from Table 1)

5 Conclusions

Quantitative BSDT PL epistemology has been developed from the first principles, coevolution and causality. With its help, a semi-representational network memory model for meaningful words has been constructed and applied to complete quantitative description of memory ROCs measured in healthy humans and patients with injured brains. It has been demonstrated that *everything formal* that can ever be known about *subjective experience* of humans performing word-recognition tasks is a frequency estimation of the distribution of posterior Bayesian probabilities of given the confidence recall/recognition decisions. Samples of such distributions have been found by fitting empirical memory ROCs adopted from [7]. Hence, for the first time, epistemology has been considered as a branch of natural or even applied sciences.

The non-uniformity of Bayesian probability distributions has been proposed to use to measure the capacity to make motivationally biased recall/recognition decisions. It has been demonstrated that *cognitive learning* we introduced [3] is a complementary mix of statistical learning [1] (the ability to discern features of things) and Bayesian learning [2] (the ability to make motivationally biased decisions).

Results could for example be used in engineering (e.g., for designing robots or computer codes imitating experience-dependent behaviors of animals/humans), medicine (e.g., for estimating functional mind deficit or brain damage degree of patients with injured brains), psychology and cognitive sciences (e.g., for explaining the mind, mind-related behavioral and neurobiological phenomena or for developing most efficient learning methodologies), and probability theory (e.g., for clarifying the relations between definitions of probabilities by Bernoulli and Bayes/Laplace).

References

1. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
2. Neapolitan, R.E.: Learning Bayesian Networks. Prentice Hall, Upper Saddle River (2003)
3. Gopych, P., Gopych, I.: BSDT ROC and cognitive learning hypothesis. In: Herrero, Á. et al. (eds.) CISIS 2010. AISC 85, pp. 13–23. Springer, Berlin-Heidelberg (2010)
4. Gopych, P.M.: Elements of the binary signal detection theory, BSDT. In: Yoshida, M., Sato, H. (eds.) New Research in Neural Networks, pp. 55–63. Nova Science, New York (2008)
5. Gopych, P.: Beyond the Zermelo-Fraenkel axiomatic system: BSDT primary language and its perspective applications. Int. J. Adv. Intell. Syst. **5**, 493–517 (2012)
6. Gopych, P.: Biologically plausible BSDT recognition of complex images: the case of human faces. Int. J. Neural Syst. **18**, 527–545 (2008)
7. Yonelinas, A.P., Kroll, N.E., Quamme, J.R. et al.: Effects of extensive temporal lobe damage or mild hypoxia on recollection and familiarity. Nat. Neurosci. **5**, 1236–1241 (2002)
8. Laplace, P.S.: A philosophical essay on probabilities. In: Truscott, F.W., Emory, F.L. (translated from the 6th French edn.). Wiley, New York (1902)
9. Gopych, P.: Minimal BSDT abstract selectional machines and their selectional and computational performance. In: Yin, H., Tino, P., Corchado, E., Byrne, W., Yao, X. (eds.) Ideal 2007. LNCS, vol. 4881, pp. 198–208. Springer, Berlin-Heidelberg (2007)
10. Gopych, P.: Thinking machines versus thinking organisms. In: Iliadis, L., Papadopoulos, H., Jane, C. (eds.) EANN2013, Part I. CCIS, vol. 383, pp. 71–80. Springer, Berlin-Heidelberg (2013)
11. Rizzolatti, G., Craighero, L.: The Mirror-neuron system. Ann. Rev. Neurosci. **27**, 169–192 (2004)
12. Keysers, C., Kaas, J.H., Gazzola, V.: Somatosensation in social perception. Nat. Rev. Neurosci. **11**, 417–428 (2010)
13. Youk, H., Lim, W.A.: Secreting and sensing the same molecules allows cells to achieve versatile social behaviors. Science **343**, 1242782 (2014)
14. Stolk, A., Noordzij, M.L., Verhagen, L., et al.: Cerebral coherence between communicators marks the emergence of meaning. Proc. Natl. Acad. Sci. U.S.A. **111**, 18183–18188 (2014)
15. Rhodes, G., Jeffery, L.: Adaptive norm-based coding of facial identity. Vision. Res. **46**, 2977–2987 (2006)
16. Edelman, G.M.: Wider than the Sky. Yale University Press, New Haven (2004)

Scaled Conjugate Gradient Learning for Complex-Valued Neural Networks

Călin-Adrian Popa

Abstract In this paper, we present the full deduction of the scaled conjugate gradient method for training complex-valued feedforward neural networks. Because this algorithm had better training results for the real-valued case, an extension to the complex-valued case is a natural way to enhance the performance of the complex backpropagation algorithm. The proposed method is exemplified on well-known synthetic and real-world applications, and experimental results show an improvement over the complex gradient descent algorithm.

Keywords Complex-valued neural networks · Scaled conjugate gradient algorithm · Time series prediction

1 Introduction

Over the last few years, the domain of complex-valued neural networks has received an increasing interest. Popular applications of this type of networks include antenna design, estimation of direction of arrival and beamforming, radar imaging, communications signal processing, image processing, and many others (for an extensive presentation, see [11]).

These networks appear as a natural choice for solving problems such as channel equalization or time series prediction in the signal processing domain, because some signals are naturally expressed in complex-valued form. Several methods, which include different network architectures and different learning algorithms, have been proposed to increase the efficiency of learning in complex-valued neural networks (see, for example, [18]). Some of these methods are specially designed for these networks, while others are extended from the real-valued case.

C.-A. Popa (✉)
Department of Computer and Software Engineering, Polytechnic University Timişoara, Blvd. V. Pârvan, No. 2, 300223 Timişoara, Romania
e-mail: calin.popa@cs.upt.ro

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_18

One such method, which has proven its efficiency in many applications, is the scaled conjugate gradient learning method. First proposed by [15], the scaled conjugate gradient method has become today a very known and used algorithm to train feedforward neural networks. Taking this fact into account, it seems natural to extend this learning algorithm to complex-valued neural networks, also.

In this paper, we present the deduction of the scaled conjugate gradient method. We also give a general formula to calculate the gradient of the error function that works both for fully complex, and for split complex activation functions, in the context of a multilayer feedforward complex-valued neural network. We test the proposed scaled conjugate gradient method on both synthetic and real-world applications. The synthetic applications include two fully complex function approximation problems and one split complex function approximation problem. The real-world application is a nonlinear time series prediction problem.

The remainder of this paper is organized as follows: Sect. 2 gives a thorough explanation of the conjugate gradient methods for the optimization of an error function defined on the complex plane. Then, Sect. 3 presents the scaled conjugate algorithm for complex-valued feedforward neural networks. The experimental results of the four applications of the proposed algorithms are shown and discussed in Sect. 4, along with a detailed description of the nature of each problem. Section 5 is dedicated to presenting the conclusions of the study.

2 Conjugate Gradient Algorithms

Conjugate gradient methods belong to the larger class of line search algorithms. For minimizing the error function of a neural network, a series of steps through the weight space are necessary to find the weight for which the minimum of the function is attained. Each step is determined by the search direction and a real number telling us how far in that direction we should move. In the classical gradient descent, the search direction is that of the negative gradient and the real number is the learning rate parameter. In the general case, we can consider some particular search direction, and then determine the minimum of the error function in that direction, thus yielding the real number that tells us how far in that direction we should move. This represents the line search algorithm, and constitutes the basis for a family of methods that have better performance than the classical gradient descent. Our deduction of conjugate gradient algorithms follows mainly the one presented in [3], which too follows that in [13].

Let's assume that we have a complex-valued neural network with an error function denoted by E, and an $2N$-dimensional weight vector denoted by $\mathbf{w} = (w_1^R, w_1^I, \dots, w_N^R, w_N^I)^T$. We must find the weight vector denoted by $\mathbf{w}^*$ that minimizes the function $E(\mathbf{w})$. Suppose we are iterating through the weight space to find the value of $\mathbf{w}^*$ or a very good approximation of it. Further, let's assume that at step k in the iteration, we want the search direction to be $\mathbf{p}_k$, where $\mathbf{p}_k$ is obviously an $2N$-dimensional vector.

In this case, the next value for the weight vector is given by $\mathbf{w}_{k+1} = \mathbf{w}_k + \lambda_k \mathbf{p}_k$, where the parameter λ_k is a real number telling us how far in the direction of $\mathbf{p}_k$ we want to go, which means that λ_k should be chosen to minimize $E(\lambda) = E(\mathbf{w}_k + \lambda \mathbf{p}_k)$.

This is a real-valued function in one real variable, so its minimum is attained when $\frac{\partial E(\lambda)}{\partial \lambda} = \frac{\partial E(\mathbf{w}_k + \lambda \mathbf{p}_k)}{\partial \lambda} = 0$. By the chain rule, we can write that

$$\begin{aligned}\frac{\partial E(\mathbf{w}_k + \lambda \mathbf{p}_k)}{\partial \lambda} &= \sum_{i=1}^{N} \frac{\partial E(\mathbf{w}_k + \lambda \mathbf{p}_k)}{\partial (w_i^{k,R} + \lambda p_i^{k,R})} \frac{\partial (w_i^{k,R} + \lambda p_i^{k,R})}{\partial \lambda} \\ &\quad + \sum_{i=1}^{N} \frac{\partial E(\mathbf{w}_k + \lambda \mathbf{p}_k)}{\partial (w_i^{k,I} + \lambda p_i^{k,I})} \frac{\partial (w_i^{k,I} + \lambda p_i^{k,I})}{\partial \lambda} \\ &= \sum_{i=1}^{N} \frac{\partial E(\mathbf{w}_{k+1})}{\partial w_i^{k+1,R}} p_i^{k,R} + \frac{\partial E(\mathbf{w}_{k+1})}{\partial w_i^{k+1,I}} p_i^{k,I} \\ &= \langle \nabla E(\mathbf{w}_{k+1}), \mathbf{p}_k \rangle, \end{aligned} \tag{1}$$

where $\langle \mathbf{a}, \mathbf{b} \rangle$ is the Euclidean scalar product in $\mathbb{R}^{2N} \simeq \mathbb{C}^N$, given by $\langle \mathbf{a}, \mathbf{b} \rangle = \left(\sum_{i=1}^{N} a_i \overline{b_i} \right)^R = \sum_{i=1}^{N} a_i^R b_i^R + a_i^I b_i^I$, for all $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{2N} \simeq \mathbb{C}^N$, and by a^R and a^I we denoted the real and imaginary part of the complex number a, and by $\overline{a}$ the conjugate of the complex number a.

So, if we denote $\mathbf{g} := \nabla E$, then (1) can be written in the form

$$\langle \mathbf{g}(\mathbf{w}_{k+1}), \mathbf{p}_k \rangle = 0. \tag{2}$$

The next search direction $\mathbf{p}_{k+1}$ is chosen so that the component of the gradient parallel to the previous search direction $\mathbf{p}_k$ remains zero. As a consequence, we have that $\langle \mathbf{g}(\mathbf{w}_{k+1} + \lambda \mathbf{p}_{k+1}), \mathbf{p}_k \rangle = 0$. By the Taylor series expansion to the first order, we have that $\mathbf{g}(\mathbf{w}_{k+1} + \lambda \mathbf{p}_{k+1}) = \mathbf{g}(\mathbf{w}_{k+1}) + \nabla \mathbf{g}(\mathbf{w}_{k+1})^T \lambda \mathbf{p}_{k+1}$, and then, if we take (2) into account, we obtain that $\lambda \langle \nabla \mathbf{g}(\mathbf{w}_{k+1})^T \mathbf{p}_{k+1}, \mathbf{p}_k \rangle = 0$, which is equivalent to $\langle \mathbf{H}^T(\mathbf{w}_{k+1}) \mathbf{p}_{k+1}, \mathbf{p}_k \rangle = 0$, or, further to

$$\langle \mathbf{p}_{k+1}, \mathbf{H}(\mathbf{w}_{k+1}) \mathbf{p}_k \rangle = 0, \tag{3}$$

where we denoted by $\mathbf{H}(\mathbf{w}_{k+1})$ the Hessian of the error function $E(\mathbf{w})$, because $\mathbf{g} = \nabla E$, and thus $\nabla \mathbf{g}$ is the Hessian.

The search directions that satisfy equation (3) are said to be conjugate directions. The conjugate gradient algorithm builds the search directions $\mathbf{p}_k$, such that each new direction is conjugate to all the previous ones.

Next, we will explain the linear conjugate gradient algorithm. For this, we consider an error function of the form

$$E(\mathbf{w}) = E_0 + \mathbf{b}^T \mathbf{w} + \frac{1}{2} \mathbf{w}^T \mathbf{H} \mathbf{w}, \tag{4}$$

where $\mathbf{b}$ and $\mathbf{H}$ are constants, and the matrix $\mathbf{H}$ is assumed to be positive definite. The gradient of this function is given by

$$\mathbf{g}(\mathbf{w}) = \mathbf{b} + \mathbf{H}\mathbf{w}. \tag{5}$$

The weight vector $\mathbf{w}^*$ that minimizes the function $E(\mathbf{w})$ satisfies the equation

$$\mathbf{b} + \mathbf{H}\mathbf{w}^* = 0. \tag{6}$$

As we saw earlier from equation (3), a set of $2N$ nonzero vectors $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{2N}\} \subset \mathbb{R}^{2N}$ is said to be conjugate with respect to the positive definite matrix $\mathbf{H}$ if and only if

$$\langle \mathbf{p}_i, \mathbf{H}\mathbf{p}_j \rangle = 0, \forall i \neq j. \tag{7}$$

It is easy to show that in these conditions, the set of $2N$ vectors is also linearly independent, which means that they form a basis in $\mathbb{R}^{2N} \simeq \mathbb{C}^N$. If we start from the initial point $\mathbf{w}_1$ and want to find the value of $\mathbf{w}^*$ that minimizes the error function given in (4), taking into account the above remarks, we can write that

$$\mathbf{w}^* - \mathbf{w}_1 = \sum_{i=1}^{2N} \alpha_i \mathbf{p}_i. \tag{8}$$

Now, if we set

$$\mathbf{w}_k = \mathbf{w}_1 + \sum_{i=1}^{k-1} \alpha_i \mathbf{p}_i, \tag{9}$$

then (8) can be written in the iterative form

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{p}_k, \tag{10}$$

which means that the value of $\mathbf{w}^*$ can be determined in at most $2N$ steps for the error function (4), using the above iteration. We still have to determine the real parameters α_k that tell us how much we should go in any of the $2N$ conjugate directions $\mathbf{p}_k$.

For this, we will multiply equation (8) by $\mathbf{H}$ to the left, and take the Euclidean $\mathbb{R}^{2N}$ scalar product with $\mathbf{p}_k$. Taking into account equation (6), we obtain that $-\langle \mathbf{p}_k, \mathbf{b} + \mathbf{H}\mathbf{w}_1 \rangle = \sum_{i=1}^{2N} \alpha_i \langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_i \rangle$. But, because the directions $\mathbf{p}_k$ are conjugate with respect to matrix $\mathbf{H}$, we have from (7) that $\langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_i \rangle = 0, \forall i \neq k$, so the above equation yields the following value for α_k:

$$\alpha_k = -\frac{\langle \mathbf{p}_k, \mathbf{b} + \mathbf{H}\mathbf{w}_1 \rangle}{\langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_k \rangle}. \tag{11}$$

Now, if we multiply equation (9) by $\mathbf{H}$ to the left, and take the Euclidean $\mathbb{R}^{2N}$ scalar product with $\mathbf{p}_k$, we have that: $\langle \mathbf{p}_k, \mathbf{H}\mathbf{w}_k\rangle = \langle \mathbf{p}_k, \mathbf{H}\mathbf{w}_1\rangle + \sum_{i=1}^{k-1} \alpha_i \langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_i\rangle$, or, taking into account that $\langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_i\rangle = 0, \forall i \neq k$, we get that $\langle \mathbf{p}_k, \mathbf{H}\mathbf{w}_k\rangle = \langle \mathbf{p}_k, \mathbf{H}\mathbf{w}_1\rangle$, and so the relation (11) for calculating α_k becomes:

$$\alpha_k = -\frac{\langle \mathbf{p}_k, \mathbf{b} + \mathbf{H}\mathbf{w}_k\rangle}{\langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_k\rangle} = -\frac{\langle \mathbf{p}_k, \mathbf{g}_k\rangle}{\langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_k\rangle}, \tag{12}$$

where $\mathbf{g}_k := \mathbf{g}(\mathbf{w}_k) = \mathbf{b} + \mathbf{H}\mathbf{w}_k$, as relation (5) shows.

Finally, we need to construct the mutually conjugate directions $\mathbf{p}_k$. For this, the first direction is initialized by the negative gradient of the error function at the initial point $\mathbf{w}_1$, i.e. $\mathbf{p}_1 = -\mathbf{g}_1$. We have the following update rule for the conjugate directions:

$$\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{p}_k. \tag{13}$$

Taking the Euclidean $\mathbb{R}^{2N}$ scalar product with $\mathbf{H}\mathbf{p}_k$, and imposing the conjugacy condition $\langle \mathbf{p}_{k+1}, \mathbf{H}\mathbf{p}_k\rangle = 0$, we obtain that

$$\beta_k = \frac{\langle \mathbf{g}_{k+1}, \mathbf{H}\mathbf{p}_k\rangle}{\langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_k\rangle}. \tag{14}$$

It can be easily shown by induction that repeated application of the relations (13) and (14), yield a set of mutually conjugate directions with respect to the positive definite matrix $\mathbf{H}$.

So far, we have dealt with a quadratic error function that has a positive definite Hessian matrix $\mathbf{H}$. But in practical applications, the error function may be far from quadratic, and so the expressions for calculating α_k and β_k that we deduced above, may not be as accurate as in the quadratic case. Furthermore, these expressions need the explicit calculation of the Hessian matrix $\mathbf{H}$ for each step of the algorithm, because the Hessian is constant only in the case of the quadratic error function. This calculation is computationally intensive and should be avoided. In what follows, we will deduce expressions for α_k and β_k that do not need the explicit calculation of the Hessian matrix, and do not even assume that the Hessian is positive definite.

First of all, let's consider the expression for α_k, given in (12). Because of the iterative relation (10), we can replace the explicit calculation of α_k with an inexact line search that minimizes $E(\mathbf{w}_{k+1}) = E(\mathbf{w}_k + \alpha_k \mathbf{p}_k)$, i.e. a line minimization along the search direction $\mathbf{p}_k$, starting at the point $\mathbf{w}_k$. In our experiments, we used the golden section search, which is guaranteed to have linear convergence, see [4, 14].

Now, let's turn our attention to β_k. From (5), we have that

$$\mathbf{g}_{k+1} - \mathbf{g}_k = \mathbf{H}(\mathbf{w}_{k+1} - \mathbf{w}_k) = \alpha_k \mathbf{H}\mathbf{p}_k,$$

and so the expression (14) becomes:

$$\beta_k = \frac{\langle \mathbf{g}_{k+1}, \mathbf{g}_{k+1} - \mathbf{g}_k\rangle}{\langle \mathbf{p}_k, \mathbf{g}_{k+1} - \mathbf{g}_k\rangle}.$$

This is known as the *Hestenes-Stiefel* update expression, see [10].
Similarly, we obtain the *Polak-Ribiere* update expression (see [17]):

$$\beta_k = \frac{\langle \mathbf{g}_{k+1}, \mathbf{g}_{k+1} - \mathbf{g}_k \rangle}{\langle \mathbf{g}_k, \mathbf{g}_k \rangle}. \tag{15}$$

We then have that $\langle \mathbf{g}_k, \mathbf{g}_{k+1} \rangle = 0$, and so expression (15) becomes:

$$\beta_k = \frac{\langle \mathbf{g}_{k+1}, \mathbf{g}_{k+1} \rangle}{\langle \mathbf{g}_k, \mathbf{g}_k \rangle}.$$

This expression is known as the *Fletcher-Reeves* update formula, see [19].

3 Scaled Conjugate Gradient Algorithm

As we have seen above, in real world applications, the Hessian matrix can be far from being positive definite. Because of this, Møller proposed in [15] the scaled conjugate algorithm which uses the model trust region method known from the Levenberg-Marquardt algorithm, combined with the conjugate gradient method presented above. To ensure the positive definiteness, we should add to the Hessian matrix a sufficiently large positive constant λ_k multiplied by the identity matrix. With this change, the formula for the step length given in (12), becomes

$$\alpha_k = -\frac{\langle \mathbf{p}_k, \mathbf{g}_k \rangle}{\langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_k \rangle + \lambda_k \langle \mathbf{p}_k, \mathbf{p}_k \rangle}. \tag{16}$$

Let us denote the denominator of (16) by $\delta_k := \langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_k \rangle + \lambda_k \langle \mathbf{p}_k, \mathbf{p}_k \rangle$. For a positive definite Hessian matrix, we have that $\delta_k > 0$. But if $\delta_k \leq 0$, then we should increase the value of δ_k in order to make it positive. Let $\overline{\delta_k}$ denote the new value of δ_k, and, accordingly, let $\overline{\lambda_k}$ denote the new value of λ_k. It is clear that we have the relation

$$\overline{\delta_k} = \delta_k + (\overline{\lambda_k} - \lambda_k)\langle \mathbf{p}_k, \mathbf{p}_k \rangle, \tag{17}$$

and, in order to have $\overline{\delta_k} > 0$, we must have $\overline{\lambda_k} > \lambda_k - \delta_k / \langle \mathbf{p}_k, \mathbf{p}_k \rangle$. Møller in [15] chooses to set $\overline{\lambda_k} = 2\left(\lambda_k - \frac{\delta_k}{\langle \mathbf{p}_k, \mathbf{p}_k \rangle}\right)$, and so the expression for the new value of δ_k given in (17), becomes $\overline{\delta_k} = -\delta_k + \lambda_k \langle \mathbf{p}_k, \mathbf{p}_k \rangle = -\langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_k \rangle$, which is now positive and will be used in (16) to calculate the value of α_k.

Another problem signaled above is the quadratic approximation for the error function E. The scaled conjugate gradient algorithm addresses this problem by considering a comparison parameter defined by

$$\Delta_k = \frac{E(\mathbf{w}_k) - E(\mathbf{w}_k + \alpha_k \mathbf{p}_k)}{E(\mathbf{w}_k) - E_Q(\mathbf{w}_k + \alpha_k \mathbf{p}_k)}, \tag{18}$$

where $E_Q(\mathbf{w})$ represents the local quadratic approximation of the error function in the neighborhood of the point $\mathbf{w}_k$, given by

$$E_Q(\mathbf{w}_k + \alpha_k \mathbf{p}_k) = E(\mathbf{w}_k) + \alpha_k \langle \mathbf{p}_k, \mathbf{g}_k \rangle + \frac{1}{2}\alpha_k^2 \langle \mathbf{p}_k, \mathbf{H}\mathbf{p}_k \rangle. \tag{19}$$

We can easily see that Δ_k measures how good the quadratic approximation really is. Plugging relation (19) into relation (18), and taking into account expression (12) for α_k, we have that $\Delta_k = \frac{2(E(\mathbf{w}_k) - E(\mathbf{w}_k + \alpha_k \mathbf{p}_k))}{\alpha_k \langle \mathbf{p}_k, \mathbf{g}_k \rangle}$.

The value of λ_k is then updated in the following way

$$\lambda_{k+1} = \begin{cases} \lambda_k/2, & \text{if } \Delta_k > 0.75 \\ 4\lambda_k, & \text{if } \Delta_k < 0.25 \\ \lambda_k, & \text{else} \end{cases},$$

in order to ensure a better quadratic approximation.

Thus, there are two stages of updating λ_k: one to ensure that $\delta_k > 0$ and one according to the validity of the local quadratic approximation. The two stages are applied successively after each weight update.

In order to apply the scaled conjugate gradient algorithm to a complex-valued feedforward neural network, we only need to calculate the gradient of the error function at different steps. In what follows, we will give a method for calculating such gradients, using the well-known backpropagation scheme.

Let's assume that we have a fully connected complex-valued feedforward network that has L layers, where layer 1 is the input layer, layer L is the ouput layer, and the layers denoted by $\{2, \ldots, L-1\}$ are hidden layers. The error function $E : \mathbb{R}^{2N} \simeq \mathbb{C}^N \to \mathbb{R}$ for such a network is

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{c} [(y_i^{L,R} - t_i^R)^2 + (y_i^{L,I} - t_i^I)^2],$$

where $(y_i^L)_{1 \le i \le c}$ represent the outputs of the network, $(t_i)_{1 \le i \le c}$ represent the targets, and $\mathbf{w}$ represents the vector of the weights and biases of the network. As a consequence, in order to compute the gradient $\mathbf{g}(\mathbf{w}) := \nabla E(\mathbf{w})$, we must calculate all the partial derivatives of the form $\frac{\partial E}{\partial w_{jk}^{l,R}}(\mathbf{w})$ and $\frac{\partial E}{\partial w_{jk}^{l,I}}(\mathbf{w})$, where w_{jk}^l denotes the weight connecting neuron j from layer l to neuron k from layer $l-1$, for all $l \in \{2, \ldots, L\}$. We further denote

$$s_j^l := \sum_k w_{jk}^l x_k^{l-1} + w_{j0}^l,$$

$$y_j^l := G^l(s_j^l),$$

where G^l is the activation function of layer $l \in \{2, \dots, L\}$, $(x_k^1)_{1 \le k \le d}$ are the network inputs, and $x_k^l := y_k^l$, $\forall l \in \{2, \dots, L-1\}$, $\forall k$, because x_k^1 are the inputs, y_k^L are the outputs, and $y_k^l = x_k^l$ are the outputs of layer l, which are also inputs to layer $l+1$.

By calculations, we obtain the following formula for computing the components of the gradient of the error function:

$$\frac{\partial E}{\partial w_{jk}^{l,R}}(\mathbf{w}) + i\frac{\partial E}{\partial w_{jk}^{l,I}}(\mathbf{w}) = \delta_j^l \overline{x_k^{l-1}}, \forall l \in \{2, \dots, L\},$$

where

$$\delta_j^l = \begin{cases} \left(\sum_m \overline{w_{mj}^{l+1}} \delta_m^{l+1}\right)^R \left(\frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,R}} + i\frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,I}}\right) \\ + \left(\sum_m \overline{w_{mj}^{l+1}} \delta_m^{l+1}\right)^I \left(\frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,R}} + i\frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,I}}\right), & l \le L-1 \\ (y_j^{l,R} - t_j^R)\left(\frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,R}} + i\frac{\partial G^{l,R}(s_j^l)}{\partial s_j^{l,I}}\right) \\ +(y_j^{l,I} - t_j^I)\left(\frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,R}} + i\frac{\partial G^{l,I}(s_j^l)}{\partial s_j^{l,I}}\right), & l = L \end{cases}.$$

The above formula works both for fully complex, and for split complex activation functions, and represents a unitary writing of the fully complex and split complex variants of the complex-valued backpropagation algorithm found in the literature.

4 Experimental Results

4.1 Fully Complex Synthetic Function I

The first synthetic fully complex function we test the proposed algorithm on is the two variable quadratic function $f_1(z_1, z_2) = \frac{1}{6}\left(z_1^2 + z_2^2\right)$. Fully complex functions treat complex numbers as a whole, and not the real and imaginary parts separately, as the split complex functions do. This problem was used as a benchmark to test the performance of different complex-valued neural network architectures and learning algorithms, for example in [20–22, 25].

To train the networks, we randomly generated 3000 training samples, with each sample having the inputs z_1, z_2 inside the disk centered at the origin and with radius 2.5. For testing, we generated 1000 samples with the same characteristics. All the networks had one hidden layer comprised of 15 neurons and were trained for 5000 epochs. The activation function for the hidden layer was the fully complex hyperbolic tangent function, given by $G^2(z) = \tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, and the activation function for the output layer was the identity $G^3(z) = z$.

In our experiments, we trained complex-valued feedforward neural networks using the classical gradient descent algorithm (abbreviated GD), the gradient descent algorithm with momentum (GDM), the conjugate gradient algorithm with Hestenes-Stiefel updates (CGHS), the conjugate gradient algorithm with Polak-Ribiere updates (CGPR), the conjugate gradient algorithm with Fletcher-Reeves updates (CGFR), and the scaled conjugate gradient algorithm (SCG).

Table 1 Experimental results for the function f_1

Algorithm	Training	Testing
GD	5.23e-5±9.29e-6	5.84e-5±1.16e-5
GDM	5.05e-5±9.23e-6	5.65e-5±1.10e-5
CGHS	1.78e-6±3.59e-7	2.07e-6±5.06e-7
CGPR	1.10e-5±2.56e-6	1.26e-5±3.23e-6
CGFR	9.90e-7±2.11e-7	1.16e-6±2.60e-7
SCG	**7.19e-9±2.74e-9**	**8.77e-9±3.34e-9**
FC-RBF [20, 21]	3.61e-6	9.00e-6
FC-RBF with KMC [21]	2.01e-6	1.87e-6
Mc-FCRBF [22]	2.50e-5	2.56e-6
CSRAN [25]	9.00e-6	9.00e-6
CMRAN [20, 21]	4.60e-3	4.90e-3

Training was repeated 50 times for each algorithm, and the resulted mean and standard deviation of the mean squared error (MSE) are given in Table 1.

The best algorithm was clearly SCG, followed by the conjugate gradient algorithms. The table also gives the MSE of other algorithms used to learn this function, together with the references in which these algorithms and network architectures first appeared. We can see that the proposed algorithm was better in terms of performance than all these other algorithms.

4.2 *Fully Complex Synthetic Function II*

A more complicated example is given by the following function: $f_2(z_1, z_2, z_3, z_4) = \frac{1}{1.5}\left(z_3 + 10z_1z_4 + \frac{z_2^2}{z_1}\right)$, which was used as a benchmark in [1, 22–24]. We generated 3000 random training samples and 1000 testing samples, all having the inputs inside the unit disk. Variable z_1 was chosen so that its radius is bigger than 0.1, because its reciprocal appears in the expression of the function, and otherwise it could have led to very high values of the function in comparison with the other variables. The networks had a single hidden layer with 25 neurons, the same activation functions as the ones used in the previous experiment, and were trained for 5000 epochs. Table 2 shows the results of running each one of the algorithms 50 times.

The table also presents the values of the MSE for different learning methods and architectures found in the literature.

In this experiment also, SCG had better results than the conjugate gradient algorithms, but poorer than some other types of architectures used to learn this problem.

Table 2 Experimental results for the function f_2

Algorithm	Training	Testing
GD	5.32e-4±4.35e-5	6.26e-4±1.40e-4
GDM	5.42e-4±5.05e-5	6.69e-4±2.17e-4
CGHS	1.49e-4±2.35e-6	1.66e-4±5.24e-6
CGPR	1.70e-4±5.16e-6	1.87e-4±8.86e-6
CGFR	1.48e-4±1.71e-6	1.64e-4±3.83e-6
SCG	1.37e-4±3.52e-6	1.61e-4±3.42e-6
FCRN [24]	9.00e-4	3.60e-3
FC-RBF [20, 21]	3.84e-4	2.28e-3
FC-RBF with KMC [21]	1.29e-4	8.26e-3
Mc-FCRBF [22]	**8.10e-7**	**8.10e-7**
CSRAN [25]	6.40e-5	4.00e-4
CMRAN [20, 21]	6.60e-4	2.50e-1

4.3 Split Complex Synthetic Function I

We now test the proposed algorithm on a split complex function. The function, also used in [2, 5, 12], is $f_3(x + iy) = \sin x \cosh y + i \cos x \sinh y$.

The training set had 3000 samples and the test set had 1000 samples randomly generated from the unit disk. The neural networks had 15 neurons on a single hidden layer. The activation functions were split hyperbolic tangent for the hidden layer: $G^2(x + iy) = \tanh x + i \tanh y = \frac{e^x - e^{-x}}{e^x + e^{-x}} + i\frac{e^y - e^{-y}}{e^y + e^{-y}}$, and the identity function for the output layer: $G^3(z) = z$.

The mean and standard deviation of the mean squared error (MSE) over 50 runs are presented in Table 3. The performances of the algorithms were similar to the ones in the previous experiments.

4.4 Nonlinear Time Series Prediction

The last experiment deals with the prediction of complex-valued nonlinear signals. It involves passing the output of the autoregressive filter given by $y(k) = 1.79y(k - 1) - 1.85y(k - 2) + 1.27y(k - 3) - 0.41y(k - 4) + n(k)$, through the nonlinearity given by $z(k) = \frac{z(k-1)}{1+z^2(k-1)} + y^3(k)$, which was proposed in [16], and then used in [7–9].

Table 3 Experimental results for the function f_3

Algorithm	Training	Testing
GD	7.29e-4±1.71e-4	7.72e-4±1.78e-4
GDM	9.20e-4±2.15e-4	9.67e-4±2.20e-4
CGHS	8.83e-6±2.57e-6	9.82e-6±2.83e-6
CGPR	1.88e-4±4.14e-5	2.04e-4±4.42e-5
CGFR	8.02e-6±1.85e-6	8.83e-6±2.15e-6
SCG	**5.61e-7±1.12e-7**	**6.17e-7±1.24e-7**

Table 4 Experimental results for nonlinear time series prediction

Algorithm	Prediction gain
GD	3.64±3.49e-1
GDM	3.68±4.40e-1
CGHS	8.35±8.34e-4
CGPR	8.31±2.59e-2
CGFR	8.34±7.49e-4
SCG	**8.35±3.51e-4**
CLMS [26]	1.87
CNGD [9]	2.50
CRTRL [6]	3.76

The complex-valued noise $n(k)$ was chosen so that the variance of the signal as a whole is 1, taking into account the fact that $\sigma^2 = (\sigma^R)^2 + (\sigma^I)^2$. The tap input of the filter was 4, and so the networks had 4 inputs, 4 hidden neurons and one output neuron. They were trained for 5000 epochs with 5000 training samples.

After running each algorithm 50 times, the results are given in Table 4. In the table, we presented a measure of performance called *prediction gain*, defined by $R_p = 10 \log_{10} \frac{\sigma_x^2}{\sigma_e^2}$, where σ_x^2 represents the variance of the input signal and σ_e^2 represents the variance of the prediction error. The prediction gain is given in dB. It is obvious that, because of the way it is defined, a bigger prediction gain means better performance. It can be easily seen that in this case, SCG, CGHS, and CGFR gave approximately the same results, with CGPR performing slightly worse, and these results were better than those of some classical algorithms and network architectures found in the literature.

5 Conclusions

The full deductions of the scaled conjugate gradient algorithm and of the most known variants of the conjugate gradient algorithm for training complex-valued feedforward neural networks were presented. A method for computing gradients of the error function was given, which can be applied both for fully complex and for split complex activation functions. The three variants of the conjugate gradient algorithm with different update rules and the scaled conjugate gradient algorithm for optimizing the error function were applied for training networks used to solve four well-known synthetic and real-world problems.

Experimental results showed that the scaled conjugate gradient method performed better on the proposed problems than the classical gradient descent and gradient descent with momentum algorithms, in some cases as much as four orders of magnitude better in terms of training and testing mean squared error.

The scaled conjugate gradient algorithm was generally better than the classical variants of the conjugate gradient algorithm. This order of the algorithms in terms of performance is consistent with the one observed in the real-valued case, yet another argument for the extension of these learning methods to the complex-valued domain.

As a conclusion, it can be said that the scaled conjugate gradient algorithm represents an efficient and fast method for training feedforward complex-valued neural networks, as it was shown by its performance in solving very heterogeneous synthetic and real-world problems.

References

1. Amin, M., Savitha, R., Amin, M., Murase, K.: Complex-valued functional link network design by orthogonal least squares method for function approximation problems. In: International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 1489–1496 (2011)
2. Arena, P., Fortuna, L., Re, R., Xibilia, M.: Multilayer perceptrons to approximate complex valued functions. Int. J. Neural Syst. **6**(4), 435–446 (1995)
3. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press Inc, New York (1995)
4. Brent, R.: Algorithms for Minimization Without Derivatives. Prentice-Hall Inc, Englewood Cliffs, New Jersey (1973)
5. Buchholz, S., Sommer, G.: On clifford neurons and clifford multi-layer perceptrons. Neural Netw. **21**(7), 925–935 (2008)
6. Goh, S., Mandic, D.: A class of low complexity and fast converging algorithms for complex-valued neural networks. In: IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing, pp. 13–22 (2004)
7. Goh, S., Mandic, D.: A complex-valued rtrl algorithm for recurrent neural networks. Neural Comput. **16**(12), 2699–2713 (2004)
8. Goh, S., Mandic, D.: Nonlinear adaptive prediction of complex-valued signals by complex-valued prnn. IEEE Trans. Signal Process. **53**(5), 1827–1836 (2005)
9. Goh, S., Mandic, D.: Stochastic gradient-adaptive complex-valued nonlinear neural adaptive filters with a gradient-adaptive step size. IEEE Trans. Neural Netw. **18**(5), 1511–1516 (2007)
10. Hestenes, M., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**(6), 409–436 (1952)
11. Hirose, A.: Complex-Valued Neural Networks: Advances and Applications. Wiley, New York (2013)
12. Huang, G.B., Li, M.B., Chen, L., Siew, C.K.: Incremental extreme learning machine with fully complex hidden nodes. Neurocomputing **71**(4–6), 576–583 (2008)
13. Johansson, E., Dowla, F., Goodman, D.: Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method. Int. J. Neural Syst. **2**(4), 291–301 (1991)

14. Luenberger, D., Ye, Y.: Linear and nonlinear programming. In: International Series in Operations Research & Management Science, vol. 116. Springer, Berlin (2008)
15. Møller, M.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw. **6**(4), 525–533 (1993)
16. Narendra, K., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Trans. Neural Netw. **1**(1), 4–27 (1990)
17. Polak, E., Ribiere, G.: Note sur la convergence de méthodes de directions conjuguées. Rev. Fr. d'Informatique Rech. Opérationnelle **3**(16), 35–43 (1969)
18. Popa, C.A.: Enhanced gradient descent algorithms for complex-valued neural networks. In: International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), IEEE, pp. 272–279 (2014)
19. Reeves, C., Fletcher, R.: Function minimization by conjugate gradients. Comput. J. **7**(2), 149–154 (1964)
20. Savitha, R., Suresh, S., Sundararajan, N.: Complex-valued function approximation using a fully complex-valued rbf (fc-rbf) learning algorithm. In: International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 2819–2825 (2009)
21. Savitha, R., Suresh, S., Sundararajan, N.: A fully complex-valued radial basis function network and its learning algorithm. Int. J. Neural Syst. **19**(4), 253–267 (2009)
22. Savitha, R., Suresh, S., Sundararajan, N.: A self-regulated learning in fully complex-valued radial basis function networks. In: International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 1–8 (2010)
23. Savitha, R., Suresh, S., Sundararajan, N.: A fast learning complex-valued neural classifier for real-valued classification problems. In: International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 2243–2249 (2011)
24. Savitha, R., Suresh, S., Sundararajan, N.: A meta-cognitive learning algorithm for a fully complex-valued relaxation network. Neural Netw. **32**, 209–218 (2012)
25. Suresh, S., Savitha, R., Sundararajan, N.: A sequential learning algorithm for complex-valued self-regulating resource allocation network-csran. IEEE Trans. Neural Netw. **22**(7), 1061–1072 (2011)
26. Widrow, B., McCool, J., Ball, M.: The complex lms algorithm. Proc. IEEE **63**(4), 719–720 (1975)

Off-Grid Parameters Analysis Method Based on Dimensionality Reduction and Self-organizing Map

Tomas Burianek, Tomas Vantuch, Jindrich Stuchly and Stanislav Misak

Abstract Off-Grid systems are energetic objects independent of an external power supply. It uses primarily renewable sources what causes low and variable short-circuit power that must be controlled. Also in the Off-Grid system is a need to keep power quality parameters in requested limits. This requires a power quality parameters forecast and also a forecast of electric energy production from renewable sources. For these forecasts a complex analysis of Off-Grid parameters is an important task. This paper proposes method that processes data set from the Off-Grid system using Dimensionality Reduction and the Self-organizing Map to obtain relations and dependencies between power quality parameters, electric power parameters and meteorological parameters.

Keywords Self-organizing map · Principal component analysis · Time-series · Clustering · Off-grid system

1 Introduction

The Off-Grid System is a system independent of the power supply from external grids. Its specific characteristic is a low and variable short-circuit power [8] in comparison with the On-Grid systems. This is given mainly by a character of a source part because the Off-Grid systems use primarily renewable sources (RES). These

T. Burianek (✉) · T. Vantuch · J. Stuchly · S. Misak
Department of Computer Science, Department of Electrical Power Engineering,
VSB - Technical University of Ostrava, FEECS,
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
e-mail: tomas.burianek.st1@vsb.cz

T. Vantuch
e-mail: tomas.vantuch@vsb.cz

J. Stuchly
e-mail: jindrich.stuchly@vsb.cz

S. Misak
e-mail: stanislav.misak@vsb.cz

RES have a stochastic character and in a conjunction with storage devices are only sources of a short-circuit power. This problem with a stochastic character of RES power supply is possible to solve by a storage device, whereas the power management in the Off-Grid is controlled by using sophisticated artificial methods for example by Demand Side Management (DSM) [17]. However, another problem related to variable and a low short-circuit power is keeping power quality parameters (PQP) in requested limits. PQP are defined by the national and international standards, norms [1, 9]. (i) Level of supplied voltage, (ii) supplied voltage frequency, (iii) periodical fluctuation of voltage as well as (iv) total harmonic distortion (THD) is possible to mention from a primary PQP. These above mentioned PQP is needful to keep in defined limits for a restraint of a reliable and safe running all of the appliances in the system. PQP out of desired limits may cause reduction of a service life of the appliances and in the worst case may cause damage. This brings a need of PQP controlling implementation to the DSM conception. This is possible to realize only with understanding of a PQP progress in a time interval (PQP forecasting) and relations between PQP and other Off-Grid parameters such as electrical parameters and meteorological conditions. The developing of tools for PQP forecasting (predictors) is very important for restraint of reliable and safe Off-Grid system running whereas the developing of predictors is possible to realize only in a case of detailed analysis of relations between PQP and other Off-Grid parameters, how was mentioned.

An analysis of a time-series data is an important field of a data mining [5]. Each time-series that consists of many data points could be seen as a single object. In retrieval of relations between those complex objects a clustering methods are important tools for analysis. There are many algorithms used for a time-series clustering. The basic k-means clustering algorithm is one of the most used for this task. The k-means assigns objects to k clusters, where k is a user defined parameter [2]. Another is the Hierarchical clustering algorithm that creates a nested hierarchy of corresponding groups based on the Euclidean distance between pairs of objects [19]. The Self-organizing Map is a clustering technique that provides a mapping from a high dimensional input space to a two dimensional grid [12].

This paper proposes method for the Off-Grid parameters analysis based on the clustering that employs the Self-organizing Map (SOM). Each parameter time-series results to high-dimensional data with redundant information. Therefore a dimensionality reduction method is applied on the data to extract only important features before presenting to the Self-organizing Map. The Self-organizing Map then provides mapping from the reduced data set to the two-dimensional grid. To obtain relevant clusters from map the k-means clustering algorithm adapted to the SOM is then involved.

Paper is organized as follows: Sect. 2 describes the dimensionality reduction with the Principal Component Analysis method in more details, Sect. 3 describes the Self-organizing Map method and in Sect. 4 the k-means clustering of the Self-organizing Map is explained. Method for the Off-Grid parameters analysis is proposed in Sect. 5 and the experiments and results obtained by proposed method are discussed in Sect. 6. The conclusion about results and future work are given in Sect. 7.

2 Dimensionality Reduction

The problem of dimensionality reduction for a set of variables $X = \{x_1, \dots, x_n\}$ where $x_i \in \mathbb{R}^D$ is to find a lower dimensional representation of this set $Y = \{y_1, \dots, y_n\}$ where $y_i \in \mathbb{R}^d$ and where $d < D$ (often $d \ll D$) in such a way to preserve content of the original data as much as possible [3]. For dimensionality reduction task several methods may be used such as Principal Component Analysis (PCA), Stochastic Neighbor Embedding (SNE), Factor Analysis (FA), Diffusion Maps, Sammon Mapping, Autoencoder, Neighborhood Preserving Embedding (NPE) [15, 21]. In the next subsection the Principal Component Analysis is described in more details.

2.1 Principal Component Analysis

Principal component analysis (PCA) is a statistical linear method that provides the dimensionality reduction. Dimensionality reduction is done by embedding the original data into linear subspace with lower dimension in a way to preserve as much of a relevant information as possible [10, 18]. It finds a mapping M to lower dimension data with maximal variance. This is done by solving equation of a eigenproblem [7].

$$cov(X)M = \lambda M, \tag{1}$$

where $cov(X)$ is the covariance matrix of a input data X. The mapping matrix M is orthogonal and is formed by principal components - eigenvectors of the covariance matrix $cov(M)$. The matrix λ contains eigenvalues of the covariance matrix on diagonal. To provide the dimensionality reduction a columns of the mapping matrix M are sorted according to the decreasing eigenvalues in the matrix λ. The mapping matrix is then truncated to keep only first d principal components. A new data set Y reduced into the dimension d is then computed by Eq. 2.

$$Y = XM_d. \tag{2}$$

Principal Component Analysis was successfully applied in the area of feature extraction and signal processing [20].

3 Self Organizing Map

The Self-organizing Map (SOM) was introduced in 1982 by Kohonen [12]. This method performs a nonlinear projection mapping from a higher dimensional input space into a lower dimensional output grid of prototype nodes called map [11]. It is related to the classical Vector Quantization (VQ) [13]. The output map is usually two-dimensional that is easy for a graphical visualization and analysis.

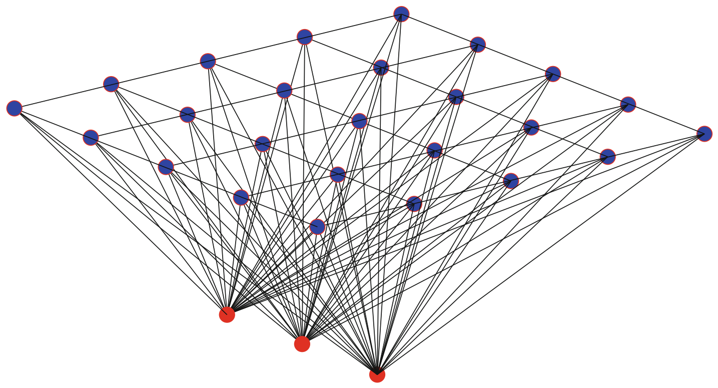


Fig. 1 Self-organizing Map structure with 5×5 map grid and 3 input nodes connected with map by weighted connections

The idea of method is that a similar data items corresponds to the prototype nodes closely related in the grid and a less similar data models are farther to each other. The Self-organizing map is based on an artificial neural network trained by an unsupervised competitive learning process. The model structure consists of map of prototype nodes, where each node has its position in the map grid and has a weigh vector of the same size as the input data space. The weight vectors preserves mapping from the high dimensional input space into the lower dimensional grid space. The map grid is usually organized into a hexagonal or a rectangular shape. Whole structure of the SOM could be presented by Fig. 1 as a two-layer neural network, where an output layer forms the grid of nodes and an input layer are input nodes, where each grid node is connected with an input node by the weighted connection.

The primary aim of the SOM learning is to optimize the weights of nodes. Thereby the map structure is organized in a way to respond to the similar input data items in the close related map prototype nodes.

Process of the training starts with initialization of all weights. It could be proceed by initialization to small random values or by initialization by random samples from the input data set. After initialization there is a repeated process of adaptation. From the input data set with the size n an input vector x_i where $0 \leq i < n$ is presented to the map by computing Euclidean distance to each node's weight vector. The input item i could be presented in ordered manner or by another method, for example could be randomly picked from the input data set. The node b with a least distance to the input vector is considered as a best matching unit (BMU). Then weights of the BMU node and nodes close to it on the map grid are adapted according to the input vector. The weight adaptation process of each node j in map with m nodes is depicted by Eq. 3.

$$w_j(t+1) = w_j(t) + \alpha(t)\eta(t, j, b)(x_i - w_j(t)). \quad (3)$$

The new weight vector for the next step $t+1$ is computed from the previous weight in the step t that is adjusted towards the input vector x_i. A magnitude of this adaptation depends on a monotonically decreasing learning coefficient $\alpha(t)$ and a neighborhood function $\eta(t, j, b)$. The neighborhood function $\eta(t, j, b)$ captures influence of the BMU node b to the current node j in the step t based on a distance between nodes on the output map grid. For the neighborhood function could be used several functions but the most common is the Gaussian function. The neighborhood influence is decreasing during the adaptation process. The adaptation process stops after selected number of steps.

4 SOM Clustering Using K-Means Algorithm

Clustering of the Self-organzing Map could be done by a hierarchical or partitive clustering algorithms [22]. In this work the partitive clustering algorithm called k-means is involved.

The k-means algorithm that was first introduced in [16] is well-known clustering method. This algorithm minimizes an error function:

$$E = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2, \tag{4}$$

where k is a number of clusters and c_i is the center of cluster S_i. Computation of error is done by sum of squared distances of each data point x in each cluster S. The algorithm of k-means iteratively computes partitioning for the data and updates cluster centers based on the error function.

In Self-organizing map the prototype nodes are used for clustering instead of all input data set. In this approach, the number of clusters k is unknown. Therefore k-means algorithm is run multiple times for each $k \in \langle 2, \sqrt{N} \rangle$ where N is number of samples and the best k settings is selected based on the Davies-Boulding index [4] calculated for each k clustering.

5 Proposed Off-Grid Parameters Analysis Method

To analyze relations and dependencies of parameters in the Off-Grid system, an algorithm consisting of several steps is proposed. A block diagram in Fig. 2 describes four main steps.

In the first step a data set is constructed from parameters, where each row is a time-series of each parameter. Each parameter has different range of values. Then each row is normalized to the same range in $\langle 0, 1 \rangle$.

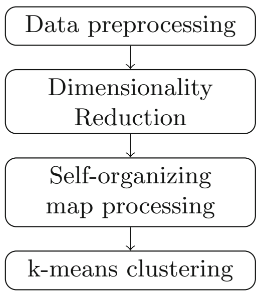


Fig. 2 Block diagram of proposed algorithm

The second step consist of dimensionality reduction of the data set to a specific dimension. In this step only important features are extracted from the original high-dimensional data and speeds up the following SOM adaptation. The Principal Component Analysis is used for task of dimensionality reduction. The specific dimension could be obtained by computing intrinsic dimension of the data set. The Maximum Likelihood Estimation [14], the eigenvalues of PCA [6] or other method may be used for this task.

In the next step the reduced data set is processed by the Self-organizing Map. Each map prototype node is labeled by parameter names, where each node may have 0, 1 or more labels. Labeled map could be used for visual analysis with the idea that parameters of the same or near nodes in the map are related to each other. This visual analysis of groups of parameters could be supported by visualization of the U-matrix, which shows distances between map nodes in the input space by coloring associated neighboring cells.

The last step performs identification of parameter groups - clusters. For this task the k-means clustering algorithm adapted to the Self-organizing Map from Sect. 4 is involved. Obtained clusters are then analyzed to reveal important relations in the Off-Grid parameter data set.

6 Experiments and Results

The experiments of the Off-Grid Parameters Analysis method were performed on the data set obtained from the Off-Grid system. This data set consists of 117 power quality, meteorological and electric power parameter time-series. Each time-series is formed by 14400 one-minute ticks.

Input settings of proposed method are showed in Table 1. Selected dimensionality reduction method was PCA described in Sect. 2.1. Setting of other parameters of the SOM are based on designed map structure and performed experiments. The results obtained by the proposed method correspond with the original idea of relations between meteorological parameters, power parameters and basic power quality parameters. Figure 3 shows on the right side the Self-organizing Map labeled by

Table 1 Table of settings

Dimensionality Reduction Method	PCA
Intrinsic Dimensionality Method	MLE
SOM Map Size	10×10
SOM Lattice	Hexa
SOM neighborhood	Gaussian
SOM learning coefficient α	0.5
SOM epochs	1000

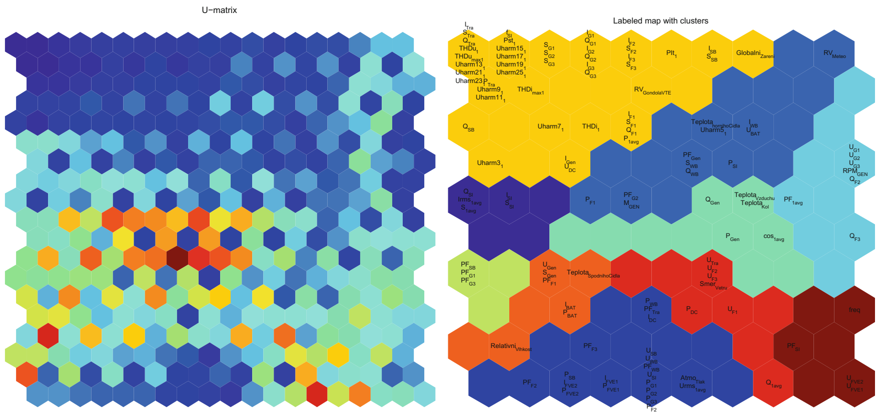


Fig. 3 Left figure represents U-matrix visualization, right figure shows clusters and labels in map

names of all parameters and on the left side the visualized U-matrix that gives notion about distances between prototype nodes in the input space and its distribution. Clusters computed by the k-means algorithm are highlighted in the right figure by distinct colors of cells. The resulting groups of parameters are shown in Table 2.

Cluster 1 associates voltage on each part of Photovoltaic power lant (PV) string U_{FVE_1} and U_{FVE_2} which are in relation with frequency *freq* in the Off-Grid System and subsequently affect PF_{SI} - a power factor of Off-Grid inverter. Management of Off-Grid Inverter (SI) has ability to limit output power from the renewable sources. If parameter *freq* rises to defined bound 50.5 Hz(50 Hz is nominal), Photovoltaic power plant Inverter (PVI) overpasses out of an ideal point of Maximum power point tracking (MPPT) and increases voltage on its DC inputs to value U_{OC} what is an open circuit voltage of PV arrays. In this case when consumption of electric energy in the Off-Grid system is minimal an appliances are in the stand-by mode and the parameter PF_{SI} converges to zero, because connected appliances that have almost pure capacitive character caused by switching sources. Results in this cluster correspond with this fact.

Cluster 2 collects important power quality parameters together with power and meteorological variables. These parameters associate internal linkages of the system

Table 2 Table of clusters

Cluster 1	$PF_{SI}, freq, U_{FVE2}, U_{FVE1}$
Cluster 2	$I_{Tra}, S_{Tra}, Q_{Tra}, THDu_1, THDu_{max1}, Uharm13_1, Uharm21_1, Uharm23_1, P_{Tra}, Uharm9_1, Uharm11_1, Q_{SB}, Uharm3_1, f_{SI}, Pst_1, Uharm15_1, Uharm17_1, Uharm19_1, Uharm25_1, THDi_{max1}, S_{G1}, S_{G2}, S_{G3}, Uharm7_1, I_{Gen}, U_{DC}, I_{G1}, Q_{G1}, I_{G2}, Q_{G2}, I_{G3}, Q_{G3}, THDi_1, I_{F2}, S_{F2}, I_{F3}, S_{F3}, RV_{GondolaVTE}, I_{F1}, S_{F1}, Q_{F1}, P_{1avg}, PIt_1, I_{SB}, S_{SB}, Globalni_{Zareni}$
Cluster 3	$Q_{SI}, Irms_{1avg}, S_{1avg}, I_{SI}, S_{SI}$
cluster 4	$PF_{F2}, P_{SB}, I_{FVE2}, P_{FVE2}, PF_{F3}, I_{FVE1}, P_{FVE1}, P_{WB}, PF_{Tra}, I_{DC}, U_{SB}, U_{WB}, PF_{WB}, U_{SI}, P_{G1}, P_{G2}, P_{G3}, P_{F2}, P_{F3}, Atmo_{Tlak}, Urms_{1avg}$
Cluster 5	$P_{F1}, PF_{G2}, M_{GEN}, PF_{Gen}, S_{WB}, Q_{WB}, Teplota_{hornihoCidla}, Uharm5_1, P_{SI}, I_{WB}, U_{BAT}, RV_{Meteo}$
Cluster 6	$PF_{1avg}, U_{G1}, U_{G2}, U_{G3}, RPM_{GEN}, Q_{F2}, Q_{F3}$
Cluster 7	$Q_{Gen}, P_{Gen}, Teplota_{Vzduch}, Teplota_{Kol}, cos_{1avg}$
Cluster 8	$PF_{SB}, PF_{G1}, PF_{G3}$
Cluster 9	$Rel_{Vlhkost}, U_{Gen}, S_{Gen}, PF_{F1}, I_{BAT}, P_{BAT}, Tep_{SpoCidla}$
Cluster 10	$P_{DC}, U_{Tra}, U_{F2}, U_{F3}, Smer_{Vetru}, U_{F1}, Q_{1avg}$

in a complex plane. The reason of this are relations between meteorological parameters($Globalni_{Zareni}$ and $RV_{Gondola-VTE}$) and power variables such as reactive and apparent power components (Q_{SB} and S_{SB}). The system reacts to the power variables by change of the power quality parameters such as a negative changes of harmonics of higher orders (U_{Harm7} - U_{Harm25}), overall harmonic voltage and current distortion ($Thdu$ and $Thdi$) together with short term and long term flicker severity(Pst_1 and Plt_1). Clusters such as Cluster 3, Cluster 4, Cluster 5 and other confirm previously stated ideas about mutual interactions of parameters and even revealed relations that in common system parameters evaluation are not evident. Example for this behavior could be group of parameters in Cluster 4 where PV power output (P_{FVE2} and P_{FVE1}) and current of individual PV strings (I_{FVE2} and I_{FVE1}) and AC power output of PV inverter (P_{SB}) affect atmospheric pressure ($Atmo_{Tlak}$) what makes sense in context with change of weather. Change of the atmospheric pressure leads to the change of energy production from PV. With the high atmospheric pressure a good meteorologic conditions are expected (sun shining) and PV production is near to a maximal possible power.

7 Conclusion and Future Work

In this work the Off-Grid Parameter Analysis Method was proposed. This method has four main steps: preprocessing of the data, dimensionality reduction by Principal Component Analysis, processing by the Self-organizing Map and clustering of map by the k-means algorithm. Results are presented in Fig. 3 that shows the labeled map with highlighted clusters. List of 10 obtained clusters with named parameters are showed in Table 2.

These results corresponds with the original idea of authors, that measured parameters in the Off-Grid system have an effect on each other. It means that there exist direct links between meteorological parameters, electric power parameters and parameters of power quality. One of the main objectives was analysis and verification of these relations. Results of this analysis will be important base for power quality parameters forecasting, forecasting of electric energy production from the renewable sources and also as a tool for prediction of consumer behavior in given energetic objects. All these tools will be associated in sophisticated dispatching system called the Active Demand Side Management for a complex control of energy flows in the Off-Grid systems.

Acknowledgments This paper was conducted within the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070, project ENET CZ.1.05/2.1.00/03.0069, Students Grant Competition project reg. no. SP2015/142, SP2015/146, SP2015/170, SP2015/178, project LE13011 Creation of a PROGRES 3 Consortium Office to Support Cross-Border Cooperation (CZ.1.07/2.3.00/20.0075) and project TACR: TH01020426.

References

1. IEEE Recommended Practice for Monitoring Electric Power Quality. IEEE Std 1159, c1–81 (2009) (Revision of IEEE Std 1159–1995)
2. Bradley, P.S., Fayyad, U.M.: Refining initial points for k-means clustering. ICML **98**, 91–99 (1998)
3. Cunningham, P.: Dimension reduction. In: Machine Learning Techniques for Multimedia, pp. 91–112. Springer, Berlin (2008)
4. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2), 224–227 (1979)
5. Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surv. **45**(1), 12:1–12:34 (2012)
6. Fan, M., Gu, N., Qiao, H., Zhang, B.: Intrinsic dimension estimation of data by principal component analysis. CoRR abs/1002.2050 (2010)
7. Francis, J.G.F.: The qr transformation a unitary analogue to the lr transformation part 1. Comput. J. **4**(3), 265–271 (1961)
8. Goksu, O., Teodorescu, R., Bak-Jensen, B., Iov, F., Kjr, P.: An iterative approach for symmetrical and asymmetrical short-circuit calculations with converter-based connected renewable energy sources. Application to wind power. In: Power and Energy Society General Meeting, 2012 IEEE, pp. 1–8 (2012)
9. Ji, N.: Steady-state signal generation compliant with iec61000-4-30: 2008. In: 22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013), pp. 1–4 (2013)
10. Jolliffe, I.: Principal Component Analysis. Springer Series in Statistics, Springer, Berlin (2002)
11. Kohonen, T.: The self-organizing map. Proc. IEEE **78**(9), 1464–1480 (1990)
12. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biol. Cybern. **43**(1), 59–69 (1982)
13. twenty-fifth Anniversay Commemorative Issue Essentials of the self-organizing map. Neural Networks. **37**(0), 52–65 (2013)
14. Levina, E., Bickel, P.J.: Maximum Likelihood Estimation of Intrinsic Dimension. In: NIPS (2004)

15. van der Maaten, L.J., Postma, E.O., van den Herik, H.J.: Dimensionality reduction: A comparative review. Journal of Machine Learning Research **10**(1–41), 66–71 (2009)
16. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Cam, L.M.L., Neyman, J. (eds.) Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability. vol. 1, pp. 281–297. University of California Press (1967)
17. Matallanas, E., Castillo-Cagigal, M., Gutirrez, A., Monasterio-Huelin, F., Caamao-Martn, E., Masa, D., Jimnez-Leube, J.: Neural network controller for active demand-side management with PV energy in the residential sector. Applied Energy **91**(1), 90–97 (2012)
18. Partridge, M., Calvo, R.A.: Fast dimensionality reduction and simple pca. Intelligent Data Analysis **2**(3), 203–214 (1998)
19. Rodrigues, P., Gama, J., Pedroso, J.: Hierarchical clustering of time-series data streams. Knowledge and Data Engineering, IEEE Transactions on **20**(5), 615–627 (2008)
20. Siuly, S., Li, Y.: Designing a robust feature extraction method based on optimum allocation and principal component analysis for epileptic EEG signal classification. Computer Methods and Programs in Biomedicine **119**(1), 29–42 (2015)
21. Sorzano, C.O.S., Vargas, J., Pascual-Montano, A.D.: A survey of dimensionality reduction techniques. CoRR abs/1403.2877 (2014)
22. Vesanto, J., Alhoniemi, E.: Clustering of the self-organizing map. Neural Networks, IEEE Transactions on **11**(3), 586–600 (2000)

Matrix-Valued Neural Networks

Călin-Adrian Popa

Abstract This paper introduces matrix-valued feedforward neural networks, for which the inputs, outputs, weights and biases are all square matrices. This type of networks represents a natural generalization of the complex-, hyperbolic-, quaternion- and Clifford-valued neural networks that have been intensively studied over the last few years. The full deduction of the gradient descent algorithm for training such networks is presented. The proposed networks are tested on three synthetic function approximation problems, with promising results for the future.

Keywords Clifford-valued neural networks · Backpropagation algorithm · Matrix-valued neural networks

1 Introduction

In the last few years, there has been an increasing interest in the study of neural networks with values in multidimensional domains. The simplest form of multidimensional neural networks are complex-valued neural networks, which were first introduced in the 1970s (see, for example, [24]), but have received more attention in the 1990s and in the past decade, because of their numerous applications, starting from those in complex-valued signal processing and continuing with applications in telecommunications and image processing (see, for example, [6, 9]). It has also been proven, that because they have orthogonal decision boundaries, complex-valued neural networks can solve the XOR problem and the detection of symmetry problem using a single complex-valued neuron, which is impossible using a single real-valued neuron, see [13–16].

Besides complex-valued neural networks, an extension of the neural networks in the real domain are the hyperbolic-valued neural networks, defined on the 2-dimensional algebra of hyperbolic numbers, see [4, 8, 21]. An interesting prop-

C.-A. Popa
Department of Computer and Software Engineering, Polytechnic University Timişoara,
Blvd. V. Pârvan, No. 2, 300223 Timişoara, Romania
e-mail: calin.popa@cs.upt.ro

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems
and Computing 378, DOI 10.1007/978-3-319-19824-8_20

erty exhibited by these networks is that their decision boundary consists of two hypersurfaces which can have any angle, depending on the weights of the network. If the complex-valued neural networks always have orthogonal decision boundaries, hyperbolic-valued neural networks can have anywhere from orthogonal to parallel decision boundaries, feature that might prove very effective in applications.

An extension of the complex and hyperbolic numbers are the quaternion numbers, which are defined by four real numbers, thus the quaternion algebra is 4-dimensional. Quaternion-valued neural networks were introduced in the 1990s also, in the beginning as a generalization of the complex-valued neural networks, see [1–3, 12, 17]. But soon, interesting applications were found, from chaotic time series prediction, to the 4-bit parity problem, and, in recent times, to quaternion-valued signal processing. Three dimensional objects can be represented using quaternions, and so applications have emerged in 3-dimensional image processing, also.

All the above generalizations of the real-valued neural networks fall into the wider category of Clifford-valued neural networks, which have dimension 2^n, $n \geq 1$. Clifford algebras or geometric algebras have many applications in physics and engineering, which recommends them for use in the field of neural newtorks, also. Clifford-valued neural networks were defined in [22, 23], and later discussed in, for example [5, 7, 8]. They can offer, in the future, a way to solve many problems arising in the design of intelligent systems, because of the close relation between Clifford algebras and geometry, allowing them to process different geometric objects and apply different geometric models to data.

Vector-valued neural networks are a somewhat different approach, that has no direct connection with the Clifford algebras discussed above, see [10, 11, 18]. These networks process three dimensional input vectors in two ways: one based on the vector product, which have three dimensional vectors as weights, and one which have orthogonal matrices as weights (i.e. matrices that satisfy $AA^T = A^TA = I$). This last variant was further generalized to N-dimensional vectors, and thus the N-dimensional neural networks (see [19, 20]), have N-dimensional vector inputs and outputs, but orthogonal matrix weights. Both three-dimensional and N-dimensional neural networks were used successfully in applications, the first in geometric transformations, and the second in the N-bit parity problem, which was solved using a single N-dimensional neuron.

All the considerations above, combined with the fact that complex, hyperbolic, quaternion and Clifford numbers can be written in matrix form (for example, a complex number $a + ib$, $i = \sqrt{-1}$, can be written in the form $\begin{pmatrix} a & -b \\ b & a \end{pmatrix}$), led to the natural idea of defining multidimensional neural networks with matrix inputs, outputs, weights and biases. Matrix-valued neural networks are thus a generalization of all the above neural networks, because each of the algebras on which these neural networks were defined, can be seen as a subalgebra of the algebra of square matrices, with the natural addition and multiplication of the matrices. Because of their degree of generality, these neural networks are bound to have many applications in the future at solving problems at which traditional neural networks have failed or performed poorly.

They can be used to solve matrix equations, to compute matrix averages, or to learn any functions defined on the algebra of matrices. Matrix-valued neural networks might also have interesting applications in computer vision, image processing, and all other areas related to geometric transformations of objects, but they might also perform better on some n-dimensional problems than the solutions available at this time.

The remainder of this paper is organized as follows: Sect. 2 gives the full deduction of the gradient descent algorithm for training matrix-valued feedforward neural networks. The experimental results of three applications of the proposed algorithm are shown and discussed in Sect. 3. Section 4 is dedicated to presenting the conclusions of the study.

2 Matrix-Valued Neural Networks

Consider the algebra $\mathcal{M}_n$ of square matrices of order n.

In what follows, we will define feedforward neural networks for which the inputs, outputs, weights and biases are all from $\mathcal{M}_n$, which means that they are square matrices. Let's assume we have a fully connected feedforward neural network with values from $\mathcal{M}_n$, with L layers, where 1 is the input layer, L is the output layer, and the layers denoted by $\{2, \dots, L-1\}$ are hidden layers. The error function $E : \mathcal{M}_n^N \to \mathbb{R}$ for such a network is

$$E(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{c} ||Y_i^L - T_i||^2, \tag{1}$$

where $||Y||$ is the Frobenius norm of matrix Y defined by $||Y|| := \sqrt{\mathrm{Tr}(YY^T)}$, and $\mathrm{Tr}(Y)$ is the trace of matrix Y. $(Y_i^L)_{1\le i\le c} \in \mathcal{M}_n^c$ represent the outputs of the network, $(T_i)_{1\le i\le c} \in \mathcal{M}_n^c$ represent the targets of the network, and $\mathbf{W} \in \mathcal{M}_n^N$ is the vector of all the N weights and biases of the network, which is a vector whose components are matrices from $\mathcal{M}_n$.

If we denote by $W_{jk}^l \in \mathcal{M}_n$ the weight connecting neuron j from layer l with neuron k from layer $l-1$, for all $l \in \{2, \dots, L\}$, we can define the update step of weight W_{jk}^l in epoch t as being

$$\Delta W_{jk}^l(t) = W_{jk}^l(t+1) - W_{jk}^l(t).$$

With this notation, the gradient descent method has the following update rule for the weight $W_{jk}^l \in \mathcal{M}_n$:

$$\Delta W_{jk}^l(t) = -\varepsilon \left(\frac{\partial E}{\partial [W_{jk}^l]_{ab}}(t) \right)_{1\le a,b\le n},$$

where ε is a real number representing the learning rate and we denoted by $\frac{\partial E}{\partial [W_{jk}^l]_{ab}}(t)$ the partial derivative of the error function E with respect to each element $[W_{jk}^l]_{ab}$ of the matrix $W_{jk}^l \in \mathcal{M}_n$, with $1 \leq a, b \leq n$. Thus, to minimize the function E, we need to compute the partial derivatives $\frac{\partial E}{\partial [W_{jk}^l]_{ab}}(t)$. Now, we will make the following notations

$$S_j^l = \sum_k W_{jk}^l X_k^{l-1}, \tag{2}$$

$$Y_j^l = G^l(S_j^l), \tag{3}$$

where (2) shows that the multiplication from the real-valued case is replaced with matrix multiplication, G^l represents the activation function for the layer $l \in \{2, \ldots, L\}$, $(X_k^1)_{1 \leq k \leq d} \in \mathcal{M}_n^d$ are the inputs of the network, and we have that $X_k^l := Y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$, because X_k^1 are the inputs, Y_k^L are the outputs, and $Y_k^l = X_k^l$ are the outputs of layer l, which are also inputs to layer $l+1$. The activation function is considered to be defined element-wise. For instance, for the matrix $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \in \mathcal{M}_3$, an example of activation function is the element-wise hyperbolic tangent function defined by

$$G\left(\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}\right) = \begin{pmatrix} \tanh a_{11} & \tanh a_{12} & \tanh a_{13} \\ \tanh a_{21} & \tanh a_{22} & \tanh a_{23} \\ \tanh a_{31} & \tanh a_{32} & \tanh a_{33} \end{pmatrix}.$$

We will first compute the update rule for the weights between layer $L-1$ and output layer L, i.e.

$$\varDelta W_{jk}^L(t) = -\varepsilon \left(\frac{\partial E}{\partial [W_{jk}^L]_{ab}} \right)_{1 \leq a, b \leq n}.$$

Using the chain rule, we can write the following set of relations $\forall 1 \leq a, b \leq n$:

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \sum_{1 \leq c, d \leq n} \frac{\partial E}{\partial [S_j^L]_{cd}} \frac{\partial [S_j^L]_{cd}}{\partial [W_{jk}^L]_{ab}}. \tag{4}$$

We will first handle $\frac{\partial [S_j^L]_{cd}}{\partial [W_{jk}^L]_{ab}}$. To compute this derivative, we need an explicit formula for $[S_j^L]_{cd}$, which can be easily deduced from (2):

$$[S_j^L]_{cd} = \sum_k \sum_{m=1}^{n} [W_{jk}^L]_{cm} [X_k^{L-1}]_{md}, \tag{5}$$

$\forall 1 \le c, d \le n$. Now, we can see that

$$\frac{\partial [S_j^L]_{cd}}{\partial [W_{jk}^L]_{ab}} = \begin{cases} [X_k^{L-1}]_{bd}, & \text{if } c = a \\ 0, & \text{else} \end{cases}, \tag{6}$$

and so relation (4) can be written in the form

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \sum_{d=1}^{n} \frac{\partial E}{\partial [S_j^L]_{ad}} [X_k^{L-1}]_{bd}, \tag{7}$$

or, equivalently

$$\frac{\partial E}{\partial [W_{jk}^L]_{ab}} = \sum_{d=1}^{n} \frac{\partial E}{\partial [S_j^L]_{ad}} [(X_k^{L-1})^T]_{db}. \tag{8}$$

Next, by denoting $\varDelta_j^L := \frac{\partial E}{\partial S_j^L}$, we have from the chain rule that

$$[\varDelta_j^L]_{cd} = \frac{\partial E}{\partial [S_j^L]_{cd}} = \sum_{1 \le e,f \le n} \frac{\partial E}{\partial [Y_j^L]_{ef}} \frac{\partial [Y_j^L]_{ef}}{\partial [S_j^L]_{cd}}, \tag{9}$$

$\forall 1 \le c, d \le n$. Taking into account notation (3), and the expression of the error function given in (1), we have that

$$\begin{aligned} [\varDelta_j^L]_{cd} &= \sum_{1 \le e,f \le n} ([Y_j^L]_{ef} - [T_j]_{ef}) \frac{\partial [G^L(S_j^L)]_{ef}}{\partial [S_j^L]_{cd}} \\ &= ([Y_j^L]_{cd} - [T_j]_{cd}) \frac{\partial [G^L(S_j^L)]_{cd}}{\partial [S_j^L]_{cd}}, \end{aligned} \tag{10}$$

$\forall 1 \le c, d \le n$, because $[G^L(S_j^L)]_{ef}$ depends upon $[S_j^L]_{cd}$ only for $e = c$ and $f = d$, which means that $\frac{\partial [G^L(S_j^L)]_{ef}}{\partial [S_j^L]_{cd}} = 0, \forall (e,f) \ne (c,d)$. If we denote by $\odot$ the element-wise multiplication of two matrices, the above relation gives

$$\varDelta_j^L = (Y_j^L - T_j) \odot \frac{\partial G^L(S_j^L)}{\partial S_j^L}, \tag{11}$$

where $\frac{\partial G^L(S_j^L)}{\partial S_j^L}$ represents the matrix of element-wise derivatives of the activation function G^L. For instance, if $S = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \in \mathcal{M}_3$, then

$$\frac{\partial G(S)}{\partial S} = \begin{pmatrix} \operatorname{sech}^2 a_{11} & \operatorname{sech}^2 a_{12} & \operatorname{sech}^2 a_{13} \\ \operatorname{sech}^2 a_{21} & \operatorname{sech}^2 a_{22} & \operatorname{sech}^2 a_{23} \\ \operatorname{sech}^2 a_{31} & \operatorname{sech}^2 a_{32} & \operatorname{sech}^2 a_{33} \end{pmatrix},$$

with the function G defined as in the above example.

Finally, from (8), we get the expression for the desired update rule in the form

$$\Delta W_{jk}^L(t) = -\varepsilon \Delta_j^L (X_k^{L-1})^T,$$

where the matrix $\Delta_j^L \in \mathcal{M}_n$ is given by relation (11).

Now, we will compute the update rule for an arbitrary weight W_{jk}^l, where $l \in \{2, \ldots, L-1\}$. First, we can write that

$$\Delta W_{jk}^l(t) = -\varepsilon \left(\frac{\partial E}{\partial [W_{jk}^l]_{ab}} \right)_{1 \le a,b \le n},$$

and then, from the chain rule, we have that

$$\frac{\partial E}{\partial [W_{jk}^l]_{ab}} = \sum_{1 \le c,d \le n} \frac{\partial E}{\partial [S_j^l]_{cd}} \frac{\partial [S_j^l]_{cd}}{\partial [W_{jk}^l]_{ab}}. \tag{12}$$

$\forall 1 \le a, b \le n$. Applying the chain rule again, we obtain that

$$\frac{\partial E}{\partial [S_j^l]_{cd}} = \sum_r \sum_{1 \le e,f \le n} \frac{\partial E}{\partial [S_r^{l+1}]_{ef}} \frac{\partial [S_r^{l+1}]_{ef}}{\partial [S_j^l]_{cd}}, \tag{13}$$

$\forall 1 \le c, d \le n$, where the sum is taken for all neurons r in layer $l+1$ to which neuron j from layer l sends connections. We further have that

$$\frac{\partial [S_r^{l+1}]_{ef}}{\partial [S_j^l]_{cd}} = \sum_{1 \le g,h \le n} \frac{\partial [S_r^{l+1}]_{ef}}{\partial [Y_j^l]_{gh}} \frac{\partial [Y_j^l]_{gh}}{\partial [S_j^l]_{cd}}, \tag{14}$$

$\forall 1 \le c, d \le n$, $\forall 1 \le e, f \le n$. Again from (2), we can compute

$$\frac{\partial[S_r^{l+1}]_{ef}}{\partial[Y_j^l]_{gh}} = \begin{cases} [W_{rj}^{l+1}]_{eg}, & \text{if } f = h \\ 0, & \text{else} \end{cases}, \tag{15}$$

and then

$$\frac{\partial[S_r^{l+1}]_{ef}}{\partial[S_j^l]_{cd}} = \sum_{g=1}^{n} [W_{rj}^{l+1}]_{eg} \frac{\partial[G^l(S_j^l)]_{gf}}{\partial[S_j^l]_{cd}}, \tag{16}$$

$\forall 1 \le c, d \le n$, $\forall 1 \le e,f \le n$. Now, returning to equation (13), and putting it all together, we have that

$$\begin{aligned} \frac{\partial E}{\partial[S_j^l]_{cd}} &= \sum_r \sum_{1\le e,f\le n} \frac{\partial E}{\partial[S_r^{l+1}]_{ef}} \sum_{g=1}^{n} [W_{rj}^{l+1}]_{eg} \frac{\partial[G^l(S_j^l)]_{gf}}{\partial[S_j^l]_{cd}} \\ &= \sum_r \sum_{1\le e,f,g\le n} [W_{rj}^{l+1}]_{eg} \frac{\partial E}{\partial[S_r^{l+1}]_{ef}} \frac{\partial[G^l(S_j^l)]_{gf}}{\partial[S_j^l]_{cd}} \\ &= \sum_r \left(\sum_{e=1}^{n} [(W_{rj}^{l+1})^T]_{ce} \frac{\partial E}{\partial[S_r^{l+1}]_{ed}} \right) \frac{\partial[G^l(S_j^l)]_{cd}}{\partial[S_j^l]_{cd}} \\ &= \sum_r [(W_{rj}^{l+1})^T \Delta_r^{l+1}]_{cd} \frac{\partial[G^l(S_j^l)]_{cd}}{\partial[S_j^l]_{cd}}, \end{aligned} \tag{17}$$

$\forall 1 \le c, d \le n$, where again we took into account the fact that $\frac{\partial[G^l(S_j^l)]_{gf}}{\partial[S_j^l]_{cd}} = 0, \forall (g,f) \neq (c,d)$. Now, by denoting $\Delta_j^l := \frac{\partial E}{\partial S_j^l}$, we can write the above relation in the form

$$\Delta_j^l = \left(\sum_r (W_{rj}^{l+1})^T \Delta_r^{l+1} \right) \odot \frac{\partial G^l(S_j^l)}{\partial S_j^l}, \tag{18}$$

where $\odot$ denotes the element-wise multiplication of two matrices.

Finally, taking into account the fact that

$$\frac{\partial[S_j^l]_{cd}}{\partial[W_{jk}^l]_{ab}} = \begin{cases} [X_k^{l-1}]_{bd}, & \text{if } c = a, \\ 0, & \text{else} \end{cases}, \tag{19}$$

relation (12) becomes

$$\frac{\partial E}{\partial [W^l_{jk}]_{ab}} = \sum_{d=1}^{n} \frac{\partial E}{\partial [S^l_j]_{ad}} [X^{l-1}_k]_{bd}$$
$$= \sum_{d=1}^{n} \frac{\partial E}{\partial [S^l_j]_{ad}} [(X^{l-1}_k)^T]_{db}$$
$$= [\varDelta^l_j (X^{l-1}_k)^T]_{ab}, \tag{20}$$

$\forall 1 \leq a, b \leq n$. Thus, the update rule for the weight W^l_{jk} can be written in matrix form in the following way:

$$\varDelta W^l_{jk}(t) = -\varepsilon \varDelta^l_j (X^{l-1}_k)^T,$$

which is similar to the formula we obtained for the layer L.

To summarize, we have the following formula for the update rule of the weight W^l_{jk}:

$$\varDelta W^l_{jk}(t) = -\varepsilon \varDelta^l_j (X^{l-1}_k)^T, \forall l \in \{2, \ldots, L\}, \tag{21}$$

where

$$\varDelta^l_j = \begin{cases} \left(\sum_r (W^{l+1}_{rj})^T \varDelta^{l+1}_r\right) \odot \frac{\partial G^l(S^l_j)}{\partial S^l_j}, & l \leq L-1 \\ (Y^l_j - T_j) \odot \frac{\partial G^l(S^l_j)}{\partial S^l_j}, & l = L \end{cases}. \tag{22}$$

3 Experimental Results

3.1 Synthetic Function Approximation 1

In our experiments, we consider square matrices of order 3. The first function we will test the proposed algorithm on is the simple arithmetic mean of two matrices

$$f_1(A, B) = \frac{1}{2}(A + B). \tag{23}$$

For training of the matrix-valued neural network, we generated 3000 training samples with random elements between 0 and 1. The testing set contained 1000 samples generated in the same way. The network had 15 neurons on a single hidden layer. The activation function for the hidden layer was the element-wise hyperbolic tangent function given by

$$G^2\left(\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}\right) = \begin{pmatrix} \tanh a_{11} & \tanh a_{12} & \tanh a_{13} \\ \tanh a_{21} & \tanh a_{22} & \tanh a_{23} \\ \tanh a_{31} & \tanh a_{32} & \tanh a_{33} \end{pmatrix},$$

and for the output layer, the activation function was the identity function: $G^3(S) = S$.

The experiment showed that the neural network converges, and the mean squared error (MSE) for both the training and testing sets was approximately the same, and equal to 0.009030. Training was done for 1000 epochs. Although the result is not spectacular, we must take into account the fact that each matrix is formed of 9 real numbers, as opposed to two or four real numbers that form the complex, hyperbolic, and respectively quaternion numbers.

3.2 Synthetic Function Approximation 2

A more complicated function is the two variable quadratic function

$$f_2(A, B) = A^2 + B^2. \tag{24}$$

This example was inspired by the synthetic functions on which the complex- and quaternion-valued neural networks were tested. The square matrices of order 3 that compose the training and testing sets were randomly generated with elements between 0 and 1, 3000 for the training set, and 1000 for the test set. The activation functions were the same as the ones above. The architecture had 15 neurons on a single hidden layer, and the network was trained for 1000 epochs.

Like in the above experiment, the training and testing MSE had similar values, equal in this case with 0.009897. The performance is slightly worse than the one obtained in the previous experiment, but in this case, the function was more complicated. These results give reasons for hope that in the future these networks can be optimized to perform better on matrix-valued function approximation problems.

3.3 Synthetic Function Approximation 3

The last experiment involves the inverse of a single matrix argument:

$$f_3(A) = A^{-1}. \tag{25}$$

This experiment was done having in mind the fact that matrix inverse is a complicated operation, which is necessary in many applications. We wanted to see if the matrix inversion can be learned by a neural network, specifically a matrix-valued neural network.

The results were not as good as with the above two experiments, but gave reason for optimism. The network was trained using 6000 square matrices of order 3, generated in the same way as the ones in the above experiments. A network architecture with a single hidden layer comprised of 15 neurons was trained for 1000 epochs. The training and testing mean squared error (MSE) were 0.021971, bigger than in the two experiments above, but promising taking into account the complexity of the problem to be learned, and the relatively small number of training samples.

4 Conclusions

The full deduction of the gradient descent algorithm for training matrix-valued neural networks was presented. Matrix-valued neural networks are a generalization of the complex-, hyperbolic-, quaternion- and Clifford-valued neural networks, since each of the algebras these networks is defined on admits a representation as a matrix subalgebra of the square matrix algebra with the natural operations of addition and multiplication of the matrices.

Since in recent years, many applications have emerged in the field of complex- and quaternion-valued neural networks, it is natural to think that the future will bring even more applications, both for these types of networks, and for their direct generalization, namely the Clifford-valued neural networks. But taking into account the fact that Clifford algebras have dimension 2^n, $n \geq 1$, it is possible that the applications do not need such a large dimension for the values of the input data. This is where matrix-valued neural networks might come into play, because their dimension is only n^2, and thus it allows more efficient use of memory than in the case of Clifford networks.

Three synthetic function approximation problems were used to test the matrix-valued backpropagation algorithm. Although not spectacular, the performances of the networks in terms of training and testing mean squared error were promising, leaving place for future developments in the topic of matrix-valued neural networks. Because of the dimension of the input space, more accurate results should be obtained by increasing the number of training samples, and optimizing the operations done by the gradient descent algorithm.

The present work represents only a first step done towards a more general framework for neural networks, which could benefit not only from increasing the number of hidden layers and making the architecture ever more complicated, but also from increasing the dimensionality of the data that is handled by the network.

References

1. Arena, P., Baglio, S., Fortuna, L., Xibilia, M.: Chaotic time series prediction via quaternionic multilayer perceptrons. In: International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1790–1794. IEEE (1995)
2. Arena, P., Fortuna, L., Muscato, G., Xibilia, M.: Multilayer perceptrons to approximate quaternion valued functions. Neural Netw. **10**(2), 335–342 (1997)

3. Arena, P., Fortuna, L., Occhipinti, L., Xibilia, M.: Neural networks for quaternion-valued function approximation. In: International Symposium on Circuits and Systems (ISCAS), vol. 6, pp. 307–310. IEEE (1994)
4. Buchholz, S., Sommer, G.: A hyperbolic multilayer perceptron. In: International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 129–133. IEEE (2000)
5. Buchholz, S., Sommer, G.: On clifford neurons and clifford multi-layer perceptrons. Neural Netw. **21**(7), 925–935 (2008)
6. Hirose, A.: Complex-Valued Neural Networks, Studies in Computational Intelligence, vol. 400. Springer, Berlin (2012)
7. Kuroe, Y.: Models of clifford recurrent neural networks and their dynamics. In: International Joint Conference on Neural Networks (IJCNN), pp. 1035–1041. IEEE (2011)
8. Kuroe, Y., Tanigawa, S., Iima, H.: Models of hopfield-type clifford neural networks and their energy functions - hyperbolic and dual valued networks -. In: International Conference on Neural Information Processing. pp. 560–569. No. 7062 in Lecture Notes in Computer Science (2011)
9. Mandic, D., Goh, S.: Complex Valued Nonlinear Adaptive Filters Noncircularity, Widely Linear and Neural Models. Wiley, New York (2009)
10. Nitta, T.: A back-propagation algorithm for neural networks based on 3d vector product. In: International Joint Conference on Neural Netwoks (IJCNN), vol. 1, pp. 589–592. IEEE (1993)
11. Nitta, T.: Generalization ability of the three-dimensional back-propagation network. In: International Conference on Neural Networks, vol. 5, pp. 2895–2900. IEEE (1994)
12. Nitta, T.: A quaternary version of the back-propagation algorithm. In: International Conference on Neural Networks, vol. 5, pp. 2753–2756. IEEE (1995)
13. Nitta, T.: An extension of the back-propagation algorithm to complex numbers. Neural Netw. **10**(8), 1391–1415 (1997)
14. Nitta, T.: An analysis of the fundamental structure of complex-valued neurons. Neural Process. Lett. **12**(3), 239–246 (2000)
15. Nitta, T.: On the inherent property of the decision boundary in complex-valued neural networks. Neurocomputing **50**, 291–303 (2003)
16. Nitta, T.: Solving the xor problem and the detection of symmetry using a single complex-valued neuron. Neural Netw. **16**(8), 1101–1105 (2003)
17. Nitta, T.: A solution to the 4-bit parity problem with a single quaternary neuron. Neural Inf. Process. Lett. Rev. **5**(2), 33–39 (2004)
18. Nitta, T.: Three-dimensional vector valued neural network and its generalization ability. Neural Inf. Process. Lett. Rev. **10**(10), 237–242 (2006)
19. Nitta, T.: N-dimensional vector neuron. In: IJCAI Workshop on Complex-Valued Neural Networks and Neuro-Computing: Novel Methods, Applications and Implementations, pp. 2–7 (2007)
20. Nitta, T.: Complex-Valued Neural Networks: Advances and Applications, chap. N-Dimensional Vector Neuron and its Application to the N-Bit Parity Problem, pp. 59–74. Wiley, New York (2013)
21. Nitta, T., Buchholz, S.: On the decision boundaries of hyperbolic neurons. In: International Joint Conference on Neural Networks (IJCNN), pp. 2974–2980. IEEE (2008)
22. Pearson, J., Bisset, D.: Back propagation in a clifford algebra. Int. Conf. Artif. Neural Netw. **2**, 413–416 (1992)
23. Pearson, J., Bisset, D.: Neural networks in the clifford domain. In: International Conference on Neural Networks, vol. 3, pp. 1465–1469. IEEE (1994)
24. Widrow, B., McCool, J., Ball, M.: The complex lms algorithm. Proc. IEEE **63**(4), 719–720 (1975)

A Feature Clustering Approach for Dimensionality Reduction and Classification

Kotte VinayKumar, R. Srinivasan and Elijah Blessing Singh

Abstract Dimensionality reduction is one of the primary challenges when handling high dimensional data. Feature clustering is a powerful approach for reducing the dimensionality of the global feature vector when performing classification. In this paper, we discuss the current research issues in handling data streams and high dimensional data and introduce an approach to perform dimensionality reduction by computing the standard deviation of each feature with every transaction or document of the entire dataset. We then rank and cluster the features of the global feature vector to obtain feature-cluster matrix. The feature-cluster matrix so formed is used to perform dimensionality reduction. Then we show how the reduced dimensionality can be used to perform classification after elimination of noise. In this work, we classify the new test document or transaction after reducing dimensionality. In future, the idea is to cluster the features using a kernel measure and perform clustering and classification of text streams dynamically.

Keywords Feature clustering · Classification · Dimensionality reduction

1 Introduction

Clustering is a process of grouping similar entities together. This process is quite challenging as we need to face several challenges to achieve accuracy and efficiency. The fact that accuracy is challenging is because there is no thumb rule or method existing which helps us deciding the correct number of clusters. The challenge in

K.VinayKumar (✉)
Kakatiya Institute of Technology and Science, Warangal, India
e-mail: vinaykumar.kitswgl@gmail.com

R. Srinivasan
Karunya University, Coimbatore, India
e-mail: srini0402@gmail.com

E.B. Singh
School of CSE, Karunya University, Coimbatore, India
e-mail: director_cst@karunya.edu

achieving efficiency is the requirement of accurate similarity measure which can compute the similarity. In the literature several similarity measures are proposed with the aim of computing similarity between two entities [2]. But most of these similarity measures are suitable for computing similarity in low dimensional data space. There is an immediate need for coming up with new and accurate similarity measures applicable for applications related to the high dimensional data space. In this work, we concentrate on evaluating similarity measures for low dimensional and high dimensional data space.

Now the question is:

1. What a high dimensional data space is.
2. Why the similarity measures suitable for low dimensional data space become unsuitable for high dimensional data spaces.
3. How to handle noise in high dimensional data space
4. Evaluating suitability of similarity measure for computing similarity between objects defined over high dimensional data space.

In this work, we restrict our contribution to address 1, 2, 3 above. The idea is to, analytically restrict our contribution to address the problem of dimensionality reduction and extend the work in future to address above 4 by designing a suitable similarity measure for performing clustering and classification of text streams.

Let D be a data object defined over a finite set of attributes. Now any data object defined by less than 10 attributes is treated to be low dimensional and the one which is defined by more than 10 attributes is treated to be high dimensional data object [1].

Clustering high dimensional data may be defined as a search problem of finding the clusters and the spatial dimensional over which these clusters may be generated so as to be reliable [2, 6].

In [4], the authors propose an approach for reducing the dimensionality of document-word matrix using feature clustering approach and then try to classify the test document using the reduced dimension matrix. In [5], the authors introduce a similarity measure for clustering text documents and document sets. The information on various algorithms, data stream models in the literature is explained in [2, 8–10]. Finding frequent patterns in data streams using sliding window approach is contributed in [13]. An approach for clustering data streams is contributed in [11, 12, 14, 15].

2 Research Challenges in Handling High Dimensional Data

In this section we discuss the most important research challenges that require immediate attention that also need to be addressed in the view of clustering, classification of high dimensional data streams which is not addressed in detail in the literature.

2.1 Handling Noise in High Dimensional Data

Consider the customer transaction information in Table 1 for 3 customers with 10 attributes (transaction items). If we now evaluate the suitability of Euclidean measure for computing similarity between any two customers w.r.t transactions to study the behavior of customers then we cannot distinguish between the customers as all the customer have same distance value 1.414. However if we now look at the Table 1, we can arrive at a decision that the customer V and S, share at least some degree of similarity as compared to no similarity between customers V and P. This is where the Euclidean distance measure fails to be a similarity measure for high dimensional data spaces. The reason for this is the noisy data which downplays the Euclidean measure to be accurate and affective to suit for high dimensional data space applications.

Table 1 Customer transaction table

	A1	A2	A3	A4	A4	A5	A6	A7	A8	A9	A10
Vinay (V)	1	0	0	0	0	0	0	0	0	0	0
Prudvi (P)	1	0	0	0	0	0	0	0	0	0	0
Srinivas (S)	1	0	0	0	0	0	0	0	0	0	0

From the above table, using Euclidean measure we have the distance between customers (V, P), (V, S) and (P, S) as 1.414.

Table 2 Distance of records of customer transaction table

Euclidean distance measure	Prudvi	Srinivas
Vinay	1.414	1.414
Prudvi	0	1.414

From Table 2, it is evident that the traditional similarity measure is not effective for high dimensional data. The main reason for this is the noise coining out of the high dimensions. So, finding clusters without eliminating the noise data leads to inaccurate and unreliable clusters. These clusters are hence meaningless and hence cannot be considered.

2.2 Identifying Representatives for Clusters

The second challenge before data mining researchers is finding the unique representative for each cluster. This means that apart from handling noise in high dimensional data, another important challenge for researchers is finding the representatives for each cluster. These representatives must define each of the clusters uniquely, accurately and effectively.

As goes from the old saying, always find where you have lost, we need to verify and validate the suitability of existing measures by properly applying and making use of them suitably or design a new measure or approach to suit the need.

2.3 Reduce Dimensionality to Handle High Dimensional Data

Any chance of reducing high dimensionality is always good in terms of cost of processing, resources and time. One of the immediate challenges while when handling the high dimensional data is the dimensionality itself which must be reduced as much as possible. Designing and validating the suitable approach and method requires immediate attention from the researchers in the current situation and near future as huge data is generated from applications continuously and is now coined as Big Data.

2.4 Individuate Clusters and Noise to Handle Outliers

This is the most challenging task before data mining researchers. Assume we have a database containing the records of customers for an insurance company. When clustered all the customers with similar behavior are grouped. In the process of clustering, we may miss the outlier(s) who show unusual behavior such as trying to fraud the company which requires further investigation also called fraud detection. Handling outliers is also another important challenge which needs to be addressed.

2.5 Defining Suitable Distance Measure

Identifying suitable distance measures or defining new distance measure which can help to effectively reduce the dimensions of high dimensional data to its equivalent low dimensional representation is also one of the most significant challenges before data mining researchers. Reducing the dimension must also be in view of retaining important outliers without missing them. This makes to retain features which show unusual behavior as compared other entities.

3 Proposed Approach

In this section, we discuss the algorithm for clustering features of the transaction database. The cluster-feature matrix so formed is then used to reduce the dimensionality of the original matrix which is the equivalent representation in low dimensional space. The purpose of eliminating the features is to handle the noise which makes the decision incorrect in applications related to classification. Here noise is the set of features which make the similarity get deviated or decision deviated from legitimate one. Section 3.1 outlines the algorithm for feature clustering and Sect. 3.2 discusses a sample case study with a simple example database.

3.1 Feature Clustering Algorithm for Dimensionality Reduction Algorithm 1

// Database: Set of transactions of the entire dataset denoted by Tf
// Transaction: Set of items from the itemset denoted by T
// Itemset: Set of all items or features denoted by I = I_1,I_2,I_3,... I_n

1. For each feature from the global feature vector of the transaction database denoted by DB, obtain its corresponding standard deviation over the entire database. The transaction may be binary or weighted. For feature clustering con- sider binary form of transactions.

Let T_i be any given transaction denoted by T_i = I = I_1,I_2,I_3,... I_m defined over items I1 , I2 , I3 . Im.

The standard deviation for feature I_m can be found by

$$\sigma(feature_i) = \sqrt{\frac{\sum_{j=1}^{j=n}(I_{ij} - \mu_{ij})^2}{\mid DB \mid}} \tag{1}$$

2. Cluster the features based on their respective standard deviation with error rate defined by the user specified threshold denoted by $U_{min-dev}$.

3. Form the cluster-feature matrix, say G.

```
For each feature $f_1$ ∈ set or Itemset do
if ( feature $f_1$ ∈ Cluster $G_h$) then
G[h, i] =1 // here feature is same as item from itemset
else G[h, i] =0
endif
```

4. Transpose the cluster-feature matrix G to get the feature cluster matrix say, G'.

5. Reduce the dimensions of the initial transaction-feature matrix by multiplying the initial transaction-feature matrix with G obtained in step-4. Call it Tf_{new}.

6. The resulting matrixTf_{new}, is the low dimensional representation of the initial transaction-feature matrix in different subspace.

3.2 Case Study

We give an example of how our method of dimensionality reduction works. For sake of simplicity we consider a simple high dimensional customer transaction table defined for 9 customers C1, C2, C3, C9 with 10 features A1, A2, A3, A10. The reason we say high dimensional is because the customer table has 10 features which makes it high dimensional.

Table 3 A simple customer transaction table

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
C1	0	1	0	0	1	1	0	0	0	1
C2	0	0	0	0	0	1	1	1	0	0
C3	0	0	0	0	0	0	1	0	0	0
C4	0	0	1	0	1	1	1	1	0	1
C5	0	0	0	1	0	1	0	0	1	0
C6	1	1	1	0	0	1	0	0	1	0
C7	1	1	1	1	0	1	0	1	1	0
C8	1	0	1	1	0	1	0	0	0	0
C9	1	1	1	1	0	0	0	0	0	0

Step-1: For each item of the transaction, we find its corresponding standard deviation over the entire database.

Table 4 Standard deviation of features of Table 3

Feature	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
$\sigma(SD)$	0.527	0.527	0.527	0.527	0.441	0.441	0.5	0.5	0.5	0.441

Step-2: Cluster the features based on their standard deviation. In this case, we get three clusters say G1, G2, G3 (Tables 4 and 5).

Table 5 Feature clusters

Clusters	Features or transaction items
G1	A1, A2, A3, A4
G2	A5, A6, A10
G3	A7, A8, A9

Step-3: Form the cluster-feature matrix.

This is done by assigning a 1 to the corresponding cell of the matrix if the feature belongs to the cluster and a 0 if the feature does not belong to the cluster. I.e if feature i belongs to cluster k, then the element denoted by M (k, i) of the matrix is assigned 1 otherwise 0. This is shown in the Table 6 below.

Table 6 Cluster-feature matrix

Cluster/feature	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
G1	1	1	1	1	0	0	0	0	0	0
G2		0	0	0	1	1	0	0	0	1
G3	0	0	0	0	0	0	1	1	1	0

Step-4: Transpose the cluster-feature matrix obtained in step-3. This gives the feature cluster matrix which is then suitable for reducing the dimensionality of the initial customer-transaction matrix in tabular form as in Table 3

Step-5: Reduce the dimension of initial customer transaction matrix by multiplying matrix of step-1 with one obtained in step-4.

This may be shown mathematically as

$$C_{Reduce} = C_{9X10} * CF_{10X3} \tag{2}$$

Table 7 Feature-cluster matrix

Feature/cluster	G1	G2	G3
A1	1	0	0
A2	1	0	0
A3	1	0	0
A4	1	0	0
A5	0	1	0
A6	0	1	0
A7	0	0	1
A8	0	0	1
A9	0	0	1
A10	0	1	0

The high dimensional data of order 10 is now transformed into its equivalent low dimensional form of order 3 which may now be used for performing clustering, classification or any data mining related analysis (Table 7).

The final customer transaction matrix with reduced dimensionality into different subspace is shown below in Table 8.

Table 8 Low dimensional matrix representation of matrix of Table 3

Feature	A11	A12	A13
C1	1	3	0
C2	0	1	2
C3	0	0	1
C4	1	3	2
C5	1	1	1
C6	3	1	1
C7	4	1	2
C8	3	1	0
C9	4	0	0

4 Classification - Case Study

4.1 Classification Approach Using Feature Cluster Matrix - Algorithm2

Begin of algorithm: Algorithm as follows
// Database: transaction-item matrix or doc-word matrix in the binary form
// Transaction: Set of items from the itemset denoted by T
// Itemset: Set of all items or features denoted by T

1. Transform each feature of the transaction database into binary form by making non-zero feature values equal to 1 and zero valued features as 0. Then obtain the corresponding standard deviation of each feature over the entire database.

Let T_i be any given transaction denoted by $T_i = \mathrm{I} = I_1, I_2, I_3, \ldots I_m$ defined over items I1 , I2 , I3 . Im.

The standard deviation for feature I_m can be found by

$$\sigma(feature_i) = \sqrt{\frac{\sum_{j=1}^{j=n} (I_{ij} - \mu_{ij})^2}{\mid DB \mid}} \tag{3}$$

2. Cluster the features based on their respective standard deviation with error rate defined by the user specified threshold denoted by $U_{min-dev}$. In the example considered for case study in Sect. 4.2 we assume zero error rate

3. Rank the features based upon the standard deviation (S.D) values from lowest to highest. A low value of S.D implies high rank and a high value indicates low rank. We can choose top-k rank features as per user requirement.

4. Obtain the matrix in low dimensional form by multiplying the feature-cluster matrix obtained from Algorithm 1 of Sect. 3.1.

5. Reduce the new test document or transaction by multiplying with the feature cluster matrix.

6. Obtain the similarity value of new test document w.r.t each document in the input dataset. We can use the similarity measure available in the literature such as Euclidean; Cosine etc. or we can have our similarity measure defined.

7. Classify the test document to belong to the category of the document with highest similarity value.

End of Algorithm

4.2 Classification of a New Transaction or Document

We now demonstrate the use of noise elimination from the transaction-feature database for classification. Consider the Tables 9 and 10 below for Transactions or documents D1 to D9 in binary and frequency form. We use the binary form of document word Table 10 for performing dimensionality reduction.

In Tables 9 and 10, the last column indicates class of respective documents.

The Table 11 below shows the standard deviation and rank values of each feature and the Table 12 depicts the cluster-word matrix and Table 13 shows the low dimensional equivalent representation of Table 9 obtained after performing dimensionality reduction.

If a new document D10 denoted by D10 = [1 1 1 1 1 1 0 1 0 1] is to be classified then we need to find the similarity of this transaction with documents D1 to D9. The transformed document D10 is represented as shown in Table 14.

Here, we use cosine function [1, 4] and the distance of D10 with remaining set of documents as shown in Table 15 below

Table 9 A simple document-word table in binary form

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	Class
D1	1	1	1	1	1	1	1	1	0	1	Easy
D2	1	1	1	1	1	1	1	1	1	1	Easy
D3	1	1	1	0	0	0	1	1	1	0	Easy
D4	1	1	1	1	1	1	1	1	1	0	Medium
D5	1	0	1	1	1	1	1	0	1	1	Medium
D6	1	1	0	1	1	1	1	1	0	0	Medium
D7	0	0	1	1	1	1	1	1	1	1	Hard
D8	1	1	1	0	0	1	1	1	1	0	Hard
D9	1	1	0	1	1	1	1	1	1	1	Hard

Table 10 A simple document-word table in frequency form

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	Class
C1	4	6	2	3	3	1	1	1	0	1	easy
C2	5	5	3	1	1	2	3	3	2	1	easy
C3	2	3	4	0	0	0	2	1	2	0	easy
C4	2	2	3	5	6	4	3	2	1	0	medium
C5	1	0	1	2	3	2	2	0	2	1	medium
C6	3	2	0	5	6	5	4	3	0	0	medium
C7	0	0	2	3	2	3	5	4	6	1	hard
C8	2	3	3	0	0	3	5	5	4	0	hard
C9	1	1	0	3	3	2	4	3	3	1	hard

Table 11 Standard deviation and rank of features of Table 9

Feature	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
σ(SD)	0.333	0.4409	0.4409	0.4409	0.4409	0.333	0	0.333	0.4409	0.5
Rank	1	2	2	2	2	1	0	1	2	3

Table 12 Cluster-feature matrix for Table 11

Cluster/feature	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
G1	1	0	0	0	0	1	0	1	0	0
G2	0	1	1	1	1	0	0	0	1	0
G3	0	0	0	0	0	0	0	0	0	1

Table 13 Document-word matrix after dimensionality reduction

	G1	G2	C3
D1	6	14	1
D2	10	12	3
D3	3	9	2
D4	8	17	3
D5	3	8	2
D6	11	13	4
D7	7	13	5
D8	10	10	5
D9	6	10	4

Table 14 Test document after reduction

	C1	C2	C3
D10	9	14	0

Table 15 Similarity before eliminating noise

Clusters	Similarity with document D10 (binary form)	Similarity with document D10 (non-binary form)
D1	0.942809	0.877145
D2	0.894427	0.860698
D3	0.57735	0.761929
D4	0.824958	0.755929
D5	0.75	0.536111
D6	0.801784	0.646686
D7	0.75	0.449359
D8	0.668153	0.620387
D9	0.824958	0.525577

Table 16 Similarity after eliminating noise

Clusters	Similarity with document D10
D1	0.984061
D2	0.974585
D3	0.948173
D4	0.978966
D5	0.951765
D6	0.965174
D7	0.944328
D8	0.921291
D9	0.945453

The Table 16 shows that the document D10 is most similar to document D1 after elimination of noise and hence is classified to category 1, which is 'easy' in both the cases with and without noise.

5 Conclusions

In this paper, we have presented the feature clustering approach by computing standard deviation and rank of each feature which is then used to cluster the features. These clusters are denoted by feature cluster matrix. This feature cluster matrix is then used to perform dimensionality reduction of the initial high dimensional document or transaction dataset. This is then followed by demonstration of the classification process w.r.t a new document received. The approach is validated using a simple but effective case study considering the worst case scenario. We may improve the feature clustering to make it dynamic so as to cluster the features in a self clustering manner.

In future, Reduced Dimensionality may, then be used to perform clustering and classification of text streams. The idea is to verify and validate a suitable distribution function, which can act as an efficient kernel measure.

References

1. Jiawei Han, M., Kamber, J.P.: Data Mining Concepts and Techniques, 3rd edn. (2012)
2. Agarwal, C.: Data Streams Models and Algorithms. Springer Publications (2007)
3. Gama, J.: Knowledge Discovery from Databases. CRC Press (2013)
4. Jiang, J.-Y., et al.: A Fuzzy self constructing feature clustering algorithm for text classification. In: IEEE Transactions of Knowledge and Data Engineering, pp. 335–349 (2011)
5. Lin, Y.-S., et al.: A similarity measure for text classification and clustering. In: IEEE Transactions of Knowledge and Data Engineering (2013)
6. Han, J., Kamber, M.: Data mining: concepts and techniques. In: Kacprzyk, J., Jain, L.C. (eds.) vol. 54, 2nd edn. Morgan Kaufmann (2006)
7. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams, a review, SIGMODC Record, vol. 34, No 2 (2005)
8. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of PODS (2002)
9. Tatbul, N., Zdonik, S.: A subset-based load shedding approach for aggregation queries over data streams. In: Proceedings of International Conference on very Large Data Bases (VLDB) (2006)
10. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Towards an adaptive approach for mining data streams in resource constrained environments. In: The Proceedings of Sixth International Conference on Data Warehousing and Knowledge Discovery. Lecture Notes in Computer Science (LNCS), Springer (2004)
11. Charikar, M., O'Callaghan, L., Panigrahy, R.: Better streaming algorithms for clustering problems. In: Proceedings of 35th ACM Symposium on Theory of Computing (2003)
12. Aggarwal C., Han, J., Wang, J., Yu, P.: A framework for clustering evolving data streams. In: VLDB Conference (2003)
13. Chang, J.H., Lee, W.S.: estWin: online data stream stream mining of recent frequent item sets by sliding window method. J. Inf. Sci. **31**(2), 7690 (2005)
14. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining Data Streams, a Review. SIGMODC Record, vol. 34, No 2 (2005)
15. Phridviraj, M.S.B., Srinivas, C., GuruRao, C.V.: Clustering text data streams a tree based approach with ternary function and ternary feature vector. Proc. Comput. Sci. **31**, 976–984

An Application of ANNs Method for Solving Fractional Fredholm Equations

Ahmad Jafarian and S. Measoomy Nia

Abstract For the last decade, several authors demonstrated the performance of artificial neural network models over other traditional testing methods. The current research, aimed to present a global optimization technique based on combination of neural networks approach and power series method for the numerical solution of a fractional Fredholm type integro-differential equation involving the Caputo derivative. In other words, an accurate truncated power series representation of the solution function is achieved when a suitable learning algorithm is used for the suggested neural architecture. As applications of the present iterative approach, some kinds of integro-differential equations are investigated. The achieved simulations are compared with the results obtained by some existing algorithms.

Keywords Fractional Fredholm type integro-differential equation • Generalized power series expansion • ANNs approach • Caputo fractional derivative • Approximate solution

1 Introduction

Fractional Fredholm integro-differential equations are extensively appeared in mathematical modeling of real life problems. On the other hand, the analytical solutions of these kinds of equations cannot be found easily, thus there has been growing interest in the proposition of alternative numerical methods. The most commonly used ones are, fixed point method [1, 19], Adomian decomposition method [13, 17], upper and lower solutions method [12], Chebyshev wavelet

A. Jafarian (✉)
Department of Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran
e-mail: jafarian5594@yahoo.com

S. Measoomy Nia
Department of Mathematics, Science and Research Branch, Islamic Azad University, Urmia, Iran

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_22

method [20], Variational iteration and Homotopy perturbation methods [14], Taylor expansion method [5]. Among these methods, the series expansion technique is more attractive and gives a closed form solution for a linear fractional integro-differential equation.

In this study, a combinative iterative procedure will be proposed for the solution of fractional Fredholm type integro-differentialequations. The present technique uses a modification of the power series method for transforming the origin problem to non-linear algebraic equations system that can be solved with an usual method. There are a lot of optimization algorithms attainable to solve the resulting system. Among diversity of the theoretical studies, we employ an implementation of one-hidden-layer feed-forward neural net architecture to complete this procedure. In our recent works, different architecture of artificial neural networks (ANNs) have been widely applied to approximate solutions of various types of integral equations (see [6–10]). To begin the optimization process, a training set of collocation points is built by discretization of the differential interval into arbitrary length subintervals. Assuming that the equation has an unique solution, the unknown series coefficients are quantified randomly to obtain a primary approximate solution. The gradient descent based back-propagation learning algorithm is then used to achieve the unknown coefficients which were considered as network parameters. The organization of this paper proceeds as follows. In the following section, we first review some basic concepts and definitions of neural networks and then extend in detail the ANNs approach for solving the mentioned type integro-differential equation. Some numerical examples with comparison to some offered algorithms are given in Sect. 4. Obtained simulative experimental results show that the approach has the potentiality to become an effective method. The final section contains conclusions and directions for future research.

2 Illustration of the Method

In this section, we will mainly concentrate on the solution of both linear and nonlinear fractional Fredholm type integro-differential equations (F-FIDEs). As we know, all of the methods discussed in the last section are also valid, but we are interested in the proposition of a general scheme in which can be easily implemented for various classes of fractional equations. For this aim, the approximate solution is represented by means of a modification of the so called power series method, whose coefficients are estimated by training an appropriate neural network architecture. This work yields a truncated power series representation of the solution function, which usually is enough for obtaining approximation to it. We start by presenting some commonly used fundamental concepts.

2.1 A Brief Introduction to ANNs

In this part we deal with essential concepts in which will be used further on. It is not an exaggeration to say that research in the field of artificial neural networks has been growing attention in the last ten years than ever before. First of all, we make a cursory review to neural network surrogate modeling for the numerical solution of a given problem. For more information in this issue, refer to [3, 4, 16].

The proposed three-layer feed-forward neural network framework shown in Fig. 1, contains one external input, n hidden neurons and an output signal. This architecture shows how a power series expansion can be performed as a neural net. Let v and w denote the $1 \times n$ weight vectors, and also b is bias term. In the present architecture, the input signal which is represented by the mathematical symbol x, when multiplied by connection weight v_i, gives a weighted input. In this case, this product is fed through the activation function f to generate a result. Net output $N(x)$ is computed by multiplying the output of neuron i in the hidden layer with the weight parameter w_i and then adding to the bias signal b. The input-output relation of each unit in the proposed neural architecture can be summarized as follows:

- Input unit:

$$O^1 = \chi.$$

- Hidden unit:

$$O_i^2 = f(net_i),$$

$$net_i = x\ .\ v_i, i = 1, \ldots, n.$$

- Output unit:

$$N(x) = \sum_{i=1}^{n} \left(o_i^2 \cdot w_i\right) + b.$$

3 Solving the F-FIDE Problem

In this study, we restrict our attention to the nonlinear initial value fractional integro-differential equation of the form:

$${}_cD_\chi^\alpha[u(x)] = \phi_1(x, u(x)) + \lambda \int_0^1 \xi(x,t)\phi_2(u(t))dt, \quad 0 \le x \le 1,$$

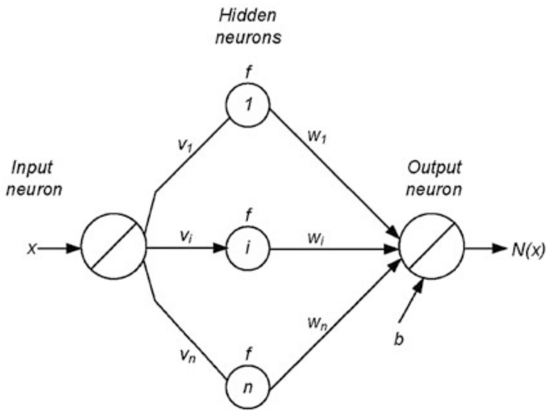


Fig. 1 Illustration of the represented neural architecture

subject to the initial condition $u(0) = \beta$. Here, ${}_cD_x^\alpha$ denotes the Caputo differential operator of order $\alpha \in (0, 1]$, λ is a real known parameter, $\phi_1, \phi_2 \in L^2([0,1])$ and $\xi \in L^2\left([0.1]^2\right)$ are given functions, and $u(x)$ id the unknown to be determined.

Definition 1 Let $u(x)$ is continuously differentiable function on finite interval $[a,b]$ up to order k. The Caputo fractional derivative operator ${}_cD_x^\alpha$ of order $\alpha > 0$ is defined by:

$$
{}_cD_x^\alpha[u(x)] = \begin{cases} \dfrac{d^k u(x)}{dx^k}, & \alpha = k \in N, \\ \dfrac{1}{\Gamma(k-\alpha)} \displaystyle\int_c^x \dfrac{u^{(k)}(\tau)}{(x-\tau)^{\alpha-k+1}} d\tau, & x > c, \quad 0 \le k-1 < \alpha < k, \end{cases}
$$

where $\Gamma(.)$ denotes the Gamma function. Recall that for the Caputo sense, derivative of a constant is zero and the following useful property holds:

$$
{}_cD_x^\alpha\left[x^k\right] = \begin{cases} 0, \quad k \in N, \quad k < \lceil \alpha \rceil, \\ \dfrac{\Gamma(k+1)}{\Gamma(k+1-\alpha)} x^{k-\alpha}, \quad x > c, \; k \in N, k \ge \lceil \alpha \rceil, \end{cases}
$$

where the ceiling function $\lceil \alpha \rceil$ denotes smallest integer greater than or equal to α. For more details and mathematical properties of fractional calculus, please refer to [18]. The reminder of this paper is organized into two segments:

the solution function is represented by a truncated power series expansion, and using the purposed neural network configuration to determine the values of the series coefficients. Sufficient and necessary conditions for existence and uniqueness of solutions of the mentioned fractional problems are given in [2].

Discretization of the Problem.
The generalized power series expansion is a powerful tool which has been developed for the numerical solution of different kinds of fractional equations. This method decomposes the solution $u(x)$ into a rapidly convergent series of solution components. In our study, we will approximate the solution function $u(x)$ by following series:

$$u(x) = \sum_{i=0}^{\infty} a_i x^{ia}, \tag{1}$$

for fractional order $\alpha \in (0,1]$ (see, [11, 15]). Under initial condition, we take $a_0 = \beta$. Thus, Eq. (1) is written as:

$$u(x) = \beta + \sum_{i=1}^{\infty} a_i x^{ia}, \tag{2}$$

For positive integer m, consider a partition of interval [0,1] with the node points $x_j = \frac{j}{m}, (for\ j = 0, \ldots, m)$. Now, let us put the collocation point x_j into the Eq. (3). After some simplifications and grouping, we obtain the following relation:

$$\begin{gathered}\sum_{i=1}^{\infty} a_i \frac{\Gamma(i+1)}{\Gamma(i+1-a)} x_j^{(i-1)a} = \phi_1\left(x_j, \beta + \sum_{i=1}^{\infty} a_i x_j^{ia}\right) + \\ \lambda \int_0^1 \xi(x_j, t)\phi_2\left(\beta + \sum_{i=1}^{\infty} a_i t^{ia}\right) dt, \quad j = 0, \ldots, m.\end{gathered} \tag{3}$$

Now, we look at the technique which will allow us to approximate desired values of the coefficients a_i (for $i \in N$).
Proposed Criterion Function.
To achieve a particular goal, only the first (n + 1) terms of defined power series (8) are used. As a result, if we express the solution function as a truncated generalized power series, so it can be easily approximated by finding the free coefficients:

$$a_i, (for\ i = 1, 2, \ldots, n).$$

This means, we intend to approximate solution u(x) by the truncated series:

$$u_n(x) = \beta + \sum_{i=1}^{n} a_i x^{ia}.$$

Below, we use the notations:

$$u_i = x^{i-1}, w_i = a_i, \quad f(x) = x^a$$

and b = β. Whit these assumptions, it can easily be argued that the output of the designed neural network is equivalent to the mentioned truncated generalized power series.

In order to train this network over the space of connection weights, the parameter a_i must be changed in such a way that the network error is reduced.

Throughout next part, an attempt will be made to see how the defined error function can be minimized over the set of node points. For this aim, the weights are to be optimized via a gradient descent optimization process. This supplement, known as back error propagation in which will be considered in following. Further details in this respect can be found in [15].

Proposed Learning Algorithm.
Before offering specific learning rule, it should be noted that learning in an artificial neural network implements a local search mechanism to obtain optimal weight values which decrease global network error. Now, we try to train the neural architecture by modifying the connecting weights according to the defined set points that can be formulated as an learning algorithm. Throughout this part, an attempt is made to construct an incremental learning process as a self learning mechanism for minimizing the predefined criterion function. Since, this function is analytical one, the gradient descent method will naturally lead to a capable learning rule. We illustrate the above idea by deriving a unsupervised back-propagation learning algorithm for adjusting the weight parameters such that the above target is minimized over the space of weight settings. The performance of this algorithm is well summarized in the following paragraph.

First, he initial weight parameters a_i (for $i = 1,\ldots, n$) are selected randomly to begin the procedure. Then, the set of node points are used to successively adjust the connection weights by moving a small step in direction in which correctly optimize the objective function. To complete the derivation of back-propagation for the output layer weights, we perform gradient de- scent rule on the criterion function (11). This standard algorithm works as follows:

$$a_i(r+1) = a_i(r) + \Delta a_i(r), \quad i = 1, \ldots, n, \tag{4}$$

$$\Delta a_i(r) = -\eta \cdot \frac{\partial E_j}{\partial a_i} + \gamma \cdot \Delta a_i(r-1), \tag{5}$$

where η and γ are the small constant learning rate and the momentum term constant, respectively. Due to its gradient descent nature, back propagation is very sensitive to the values of learning rate and momentum constant. If the choice of these initial quantities, then the learning convergence will be slow. Here, the index r in $a_i(r)$ refers to the iteration number and the subscript j in x_j is the label of the training node point.

4 Numerical Examples

In order to show the performance of the proposed neural networks approach in solving fractional integro-differential equation problems, two numerical examples are carried out in this section A comparison is made between the proposed combinative algorithm and other methods presented in []. All calculations are achieved by using the mathematical software Mat lab v7.10. Below, we use the specifications as following:

1. Learning rate: $\eta = 0.1$,
2. Momentum constant: $\gamma = 0.05$.

Also, for better comparison the root mean square error is employed as:

$$E_{mid} = \| u - u_n \|_2 = \left(\frac{1}{m+1} \sum_{j=0}^{m} \left(u\left(_{xj}\right) - u_n\left(x_j\right) \right)^2 \right)^{\frac{1}{2}}.$$

Example 3.1 Consider the following integro-differential equation:

$$_{c}D_{x}^{a}[u(x)] + 3u(x) = \phi_1(x, u(x)) + \frac{3\pi}{2} \int_0^1 xt^2 \sin(\pi u(t))dt, \quad 0 \le x \le 1,$$

Where

$$\phi_1(x, u(x)) = 3x^3 + \frac{\Gamma(4)}{\Gamma\left(\frac{7}{2}\right)} x^{\frac{5}{2}} - x,$$

with the initial condition $u(0) = 0$ and the exact solution $u(x) = x^3$. As indicated, the solution function can be uniformly approximated on that interval by (8) to any degree of accuracy. The objective here is to adaptively update the hidden layer weights a_i (for i = 1,…, 10); by attempting to optimize the associated criterion function over the set of m + 1 training points $X = \{x_0, x_1, \ldots, x_m\}$. For this aim, the weights are first normally initialized to small random values and then the learning rule considered in previous section is employed. The root mean square errors for different number of iterations and node points are presented in Table 1. Figure 2 shows the cost functions on the different number of iterations. It is noticeable that by increasing the learning steps, the criterion function goes to zero. The approximate and exact solutions are plotted and compared in Fig. 3, and also absolute error are plotted in Fig. 4 for r = 100.

Table 1 Numerical results for Example 3.1

r	E_{mid}			
	m = 10	m = 20	m = 30	m = 40
100	0.0008745	0.0007344	0.0006383	0.0005617
200	0.0006194	0.0004625	0.0003832	0.00003463
300	0.0004083	0.0002940	0.0002334	0.0002172
400	0.0003140	0.0002077	0.0001835	0.0001702
500	0.0002801	0.0001756	0.0001587	0.0001427

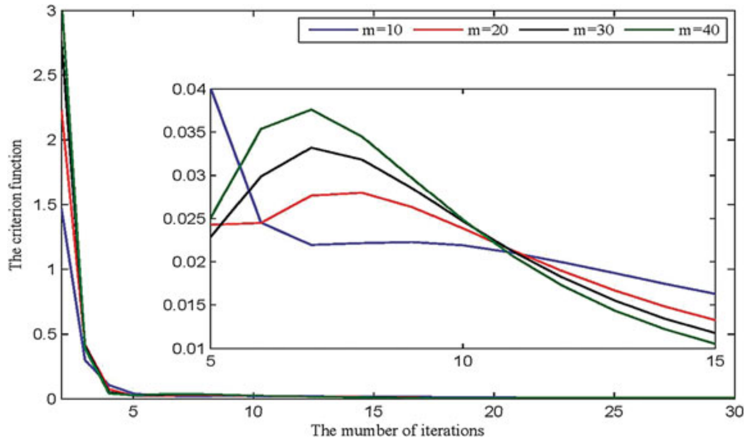


Fig. 2 The cost curves with different step sizes for Example 3.1

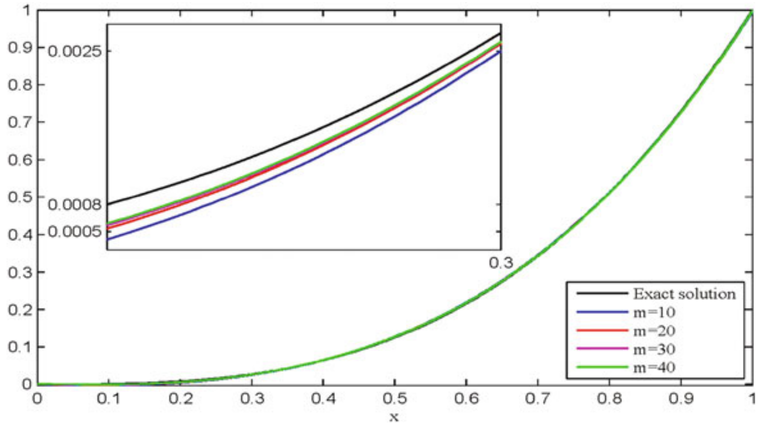


Fig. 3 The exact and approximate solutions for Example 3.1

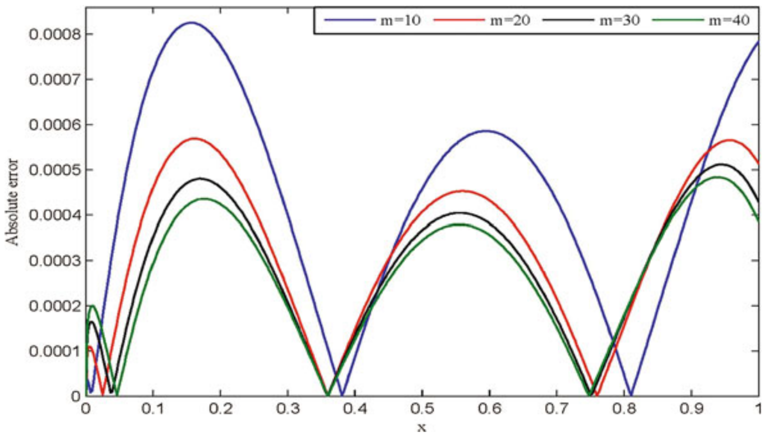


Fig. 4 The absolute error between exact and approximate solutions for Example 3.1

Example 3.2 The following fractional equation is considered:

$$_{c}D_{x}^{\alpha}[u(x)] = 1 - \frac{x}{4} + \int_{0}^{1} xt[u(t)]^{2}dt, \quad 0 \leq x \leq 1,$$

with initial condition $u(0) = 0$. Note that the exact solution of this problem for $\alpha = 1$, is $u(x) = x$. This equation is studied in [16] by using the rationalized Haar functions (RHF). The numerical results for $m = 10$ and $r = 1000$ with comparison are presented in Table 2. Based on the results, it can be concluded that our numerical simulations are in good agreement with the solutions reported in the literature. Therefore, it easily can be seen the approximate solutions for $\alpha = 0.25$, $\alpha = 0.5$ and $\alpha = 0.75$ are reliable. Figures 5 and 6 simulate the cost function and absolute error criterion for $\alpha = 1$, respectively. Since the exact solution of the above

Table 2 Numerical results for Example 3.2

x	$a = 0.25$		$a = 0.5$	
	ANN	RHF	ANN	RHF
0.1	0.650960	0.650962	0.362259	0.362260
0.2	0.821677	0.821678	0.525360	0.525361
0.3	0.959522	0.959520	0.657181	0.657181
0.4	1.084528	1.084520	0.774341	0.774336
0.5	1.203268	1.203260	0.883181	0.883175
0.6	1.131835	1.131827	0.986274	0.986267
0.7	1.431320	1.431310	1.085718	1.085710
0.8	1.543181	1.543170	1.182468	1.182460
0.9	1.654432	1.654420	1.277279	1.277270
1.0	1.765433	1.765420	1.370730	1.370720

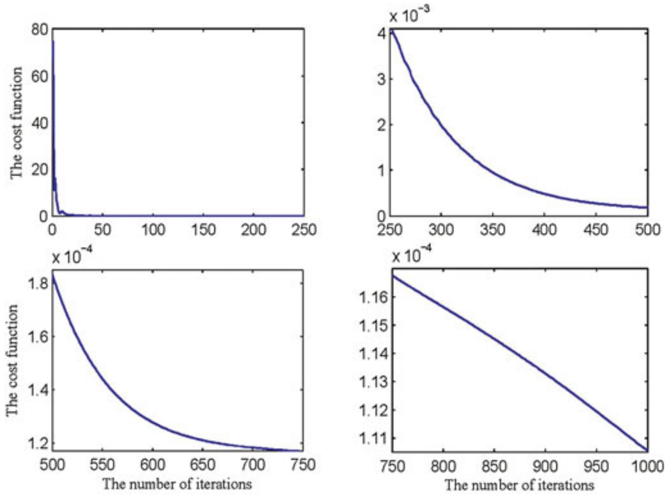


Fig. 5 The cost curve for Example 4.2

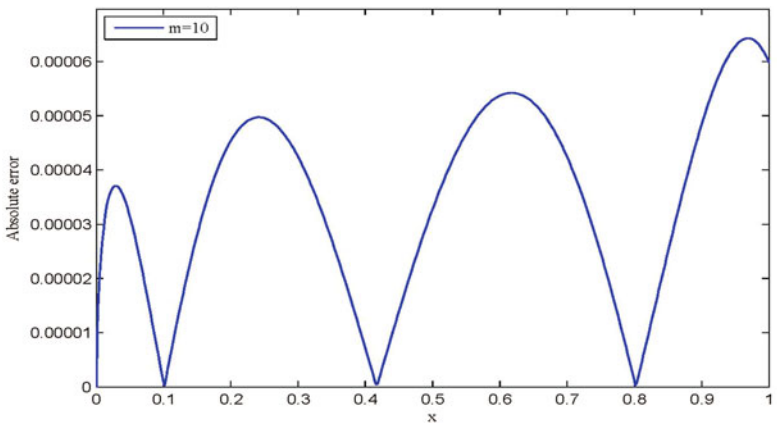


Fig. 6 The absolute error between exact and approximate solutions for Example 3.2

problem is available when $\alpha = 1$, so we are only able to show the criteria of mean absolute error for this value. As can be seen, the estimated solution is in high concurrence with the exact solution.

5 Conclusion

Fractional equations have gained increasing importance due to their several applications in the field of real world problems. As indicated, these kind of equations are very difficult to handle analytically, so we have to usually get approximate solutions. This paper explained how a combination of the power series method and neural networks approach can be developed as an iterative technique for the numerical solution of a fractional Fredholm type integro-differential equation. From a practical point of view, tow numerical examples were investigated to demonstrate the applicability of the presented iterative method. Moreover, the obtained simulation results were compared with exact solutions and also with the solutions obtained by two another works. For the future works, we are going to develop our proposed method for solving high order integro-differential equations.

References

1. Anguraj, A., Karthikeyan, P., Rivero, M., Trujillo, J.J.: On new existence results for fractional integro-differential equations with impulsive and integral conditions **66**(12), 2587–2594 (2014)
2. Bandyopadhyay, B., Kamal, S.: Stabilization and Control of Fractional Order Systems: A Sliding Mode Approach, vol. 317 (2015)
3. Graupe, D.: Principles of Artificial Neural Networks, 2nd edn. World Scientific, River Edge (2007)
4. Hanss, M.: Applied Fuzzy Arithmetic: An Introduction with Engineering Applications. Springer, Berlin (2005)
5. Huang, L., Li, X.F., Zhao, Y.L., Duan, X.Y.: Approximate solution of fractional integro-differential equations by Taylor expansion method. Comput. Math Appl. **62**(3), 1127–1134 (2011)
6. Jafarian, A., Measoomy, S., Nia, S.: Abbasbandy, artificial neural networks based modeling for solving linear Volterra integral equations system. Appl. Soft Comput. **27**, 391–395 (2015)
7. Jafarian, A., Measoomy Nia, S.: Artificial neural network approach to the fuzzy Abel integral equation problem. J. Intell. Fuzzy Syst. doi:10.3233/IFS-130980
8. Jafarian, A., Measoomy Nia, S.: New itrative method for solving linear Fredholm fuzzy integral equations of the second kined. Int. J. Ind. Math. **5**(3), 10 pp (2013)
9. Jafarian, A., Measoomy Nia, S.: Feed-back neural network method for solving linear Volterra integral equations of the second kind. Int. J. Math. Modell. Numer. Optim. **4**(3), 225–237 (2013)
10. Jafarian, A., Measoomy Nia, S.: Utilizing feed-back neural network approach for solving linear Fredholm integral equations system. Appl. Math. Modell. **37**(7), 5027–5038 (2013)
11. Jumarie, G.: Modi_ed Riemann-Liouville derivative and fractional Taylor series of nondifferentiable functions further results. Comput. Math. Appl. **51** (2006)
12. Momani, S.M., Hadid, S.B.: Some comparison results for integro-fractional differential inequalities. J. Fract. Calc. **24**, 37–44 (2003)
13. Momani, S., Noor, M.: Numerical methods for fourth order fractional integro-differential equations. Appl. Math. Comput. **182**, 754–760 (2006)
14. Nawaz, Y.: Variational iteration method and homotopy perturbation method for fourth-order fractional integro-differential equations. Comput. Math Appl. **61**(8), 2330–2341 (2011)

15. Odibat, Z., Shawagfeh, N.: Generalized Taylors formula. Appl. Math. Comput. **186**(1), 286–293 (2007)
16. Ordokhani, Y., Rahimi, N.: Solving fractional nonlinear Fredholm integro-differential equations via hybrid of rationalized Haar functions. J. Inf. Comput. Sci. **9**(3), 169–180 (2014)
17. Ray, S.S.: Analytical solution for the space fractional diffusion equation by two-step Adomian decomposition method. Commun. Nonlinear. Sci. Numer Simul. **14**, 129–306 (2009)
18. Yang, X.J.: Advanced Local Fractional Calculus and Its Applications. World Science Publisher, New York, USA (2012)
19. Zhanga, L., Ahmadb, B., Wanga, G., Agarwal, R.P.: Nonlinear fractional integro-differential equations on unbounded domains in a Banach space. J. Comput. Appl. Math. **249**, 51–56 (2013)
20. Zhu, L., Fan, Q.: Solving fractional nonlinear Fredholm integro-differential equations by the second kind Chebyshev wavelet. Commun. Nonlinear Sci. Numer. Simul. **17**, 2333–2341 (2012)

Solving Circle Packing Problem by Neural Gas

Jiri Pospichal

Abstract This paper considers the problem of finding the densest packing of N ($N = 1, 2, \ldots$) equal non-overlapping circles in a circle. This and similar packing problems in 2D or 3D space can be considered as a simplified version of various real world problems as container loading, sensor network layout or placing of facilities and therefore it has been thoroughly studied by mathematicians, computer scientists and in operations research. For most problems with smaller N, whether for packing in a circle, square or a triangle, or packing of spheres into three dimensional objects, there have been found provably optimal solutions, and a fierce competition exists to find the most effective algorithm and solutions for higher values of N. In this paper we are not trying to compete with these achievements, but we are trying to experimentally examine a possibility to use a special type of neural network, specifically the neural gas method, to solve such a problem. Experiments show, that the neural gas approach is applicable to this kind of problem and provides reasonable though slightly suboptimal results.

Keywords Circle packing • Heuristic • Optimization • Neural gas • Clustering

1 Introduction

A circle packing is an arrangement of circles within a given boundary, where no two circles overlap. Packing Equal Circles in a Circle (PECC) problem consists in finding a dense solution which can place all N non-overlapping circles or discs of unit-diameter into a smallest circle of diameter r, or alternatively, for a given circumscribing circle of unit-diameter and N equal circles to find the greatest

J. Pospichal (✉)
Faculty of Natural Sciences, University of SS. Cyril and Methodius in Trnava, 917 01 Trnava, Slovak Republic
e-mail: jiri.pospichal@gmail.com

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_23

diameter of the packed circles. Due to the requirement of highest density for a solution, some (or all) of the packed circles are mutually tangent. Similar packing problems of equal circles exist for square, rectangle, triangle, semicircle, and circular quadrant for two-dimensional containers, or e.g. for spheres in a sphere or a cube. Similarly, unequal circles with integer radii or equivalent unequal spheres are studied as well.

The problem in unbounded two dimensional Euclidean plane was already studied by Lagrange, who proved in 1773 that the highest-density arrangement of circles has the centers of the circles arranged in a hexagonal lattice (like a honeycomb), where each circle is surrounded by 6 other circles. All the mentioned problems including packing within boundaries are extensively studied by mathematicians for the beauty of the problem in order to find a provable optimum or at least upper and lower bounds and by computer scientists competing to find the most efficient heuristic. In operation research they are studied to deal with practical problems [4] in production and packing for the textile, container loading in naval, cylinder packing or dashboard layout problems in automobile and aerospace industries, and cutting and facility dispersion in food industries, and for sensor network layout or for communication networks data transfer in optimization of Quadrature amplitude modulation, where a modem transmits data as a series of points in a 2-dimensional phase-amplitude plane.

All these packing problems are NP hard optimization problems, which have been tackled using various techniques, from computer-aided optimality proofs to branch-and-bound procedures, constructive approaches, multi-start non-convex minimization, billiard simulation, and metaheuristics and multiphase heuristics including local search optimization, e.g. [18], or more recently [1–3, 6, 9, 13, 18], based typically on a sort of spring model. Overview of the methods can be found in [7]. Even monographs have been already devoted to such packing problems [5, 15, 16]. Likely, the most current results can be found in internet resources [14, 17, 19]. Even though neural networks have not been yet used for circle packing, the inverse is true; circle packing was used for cortical mapping [10].

Circle packing problems are being solved even for thousands of circles, although improvements were found within last five years also for slightly more than a hundred circles. Even though the problems are NP hard, the combined heuristics, analogously to Traveling Salesman Problem, are very fast and very good.

Similarly to the Traveling salesman problem, where Hopfield used his neural network to solve a 10-city problem [8], the present paper tries to use a neural network approach to solve a circle packaging problem, although the approach is substantially slower and results suboptimal compared to specialized heuristics. The goal here is to prove feasibility of the concept, not its effectiveness. In the further presented approach there will be therefore provided no comparison of the effectiveness of the proposed algorithm against other techniques; only a very basic comparison regarding the quality of the results compared with ideal results shall be provided.

2 Neural Gas Approach to Circle Packing

Neural gas is an artificial neural network approach, related to self-organizing map (SOM), which was introduced in 1991 [12]. In a similar way to SOM, it can be used for clustering or putting together related data. Expressed in another way, it finds optimal data representations based on feature vectors, and is typically used in pattern recognition, more specifically in speech recognition or image compression.

Like SOM and some other artificial neural networks, neural gas adaptation includes two phases - training and mapping. "Training" creates a "map" by a competitive process called vector quantization from input examples. "Mapping" classifies a new input vector. Neural gas is composed of N neurons (their number is fixed in advance), and with each node i there is associated a weight vector w_i of the same dimension as the input data vectors. These weight or feature vectors flit during the training within data space as gas molecules would in space, therefore the algorithm was named neural gas.

In our case, each weight vector just corresponds to coordinates of its neuron in 2D space. After the learning, its coordinates should ideally correspond to a centre of a "packed" circle and, from a user defined number of N neurons we shall have packing of N circles within the containing circle. The radius of the circles will be defined as the minimum of either half of the smallest Euclidean distance between all couples of centers of "packed" circles (i.e. Euclidean distance of weight vectors of neurons) or the distance of centers of "packed" circles from a boundary of circumscribing circle. This unfortunately means that only two circles are sure to be tangent, either two packed circles or a packed circle and the circumscribing circle; if other circles are close enough to touch, it is due to the quality of optimization.

Initially, each neuron is associated with a random point within a containing circle. Then a randomly selected point $\boldsymbol{x}$ from within the containing circle is presented to the neural gas network. (Actually, we generated a 10000 random points within the circle in advance, and these are then randomly selected one by one and fed and "clustered" by neural gas; instead of generating 50000 points, we iterated the 10000 five times). Euclidean distances of the selected point $\boldsymbol{x}$ to all the weight vectors of neurons, i.e. centers of the embedded circles, are calculated, and these centers are sorted by their distance from the selected point, from closest i_1 to most distant i_N. Then each weight vector of the ordered sequence is adapted by

$$w_i^{new} = w_i^{old} + \varepsilon \cdot e^{-k_i/\lambda}\left(x - w_i^{old}\right), \quad i = 1, \ldots, N \tag{1}$$

where k_i is the number of neurons with weight vector closer to the current point $\boldsymbol{x}$ than the current weight $\boldsymbol{w}_i$ (i.e. index of its vector in ordered sequence, minus 1), ε is an adaptation step size and λ is neighborhood range. The parameters ε and λ are reduced with increasing number of presented points by equations

$$\varepsilon_{iter} = \varepsilon_{start} \cdot \left(\frac{\varepsilon_{final}}{\varepsilon_{start}}\right)^{\left(\frac{iter}{itermax}\right)} \tag{2}$$

$$\lambda_{iter} = \lambda_{start} \cdot \left(\frac{\lambda_{final}}{\lambda_{start}}\right)^{\left(\frac{iter}{itermax}\right)} \tag{3}$$

where *iter* is number of points presented so far, *itermax* is total number of presented points. Initial value of ε_{start} was set to $\varepsilon_{start} = 0.2$, its final value was set to $\varepsilon_{final} = 0.00005$, and for λ these parameters were $\lambda_{start} = 10$, $\lambda_{final} = 0.01$. The adaptation steps are actually similar to gradient descent method in classical neural

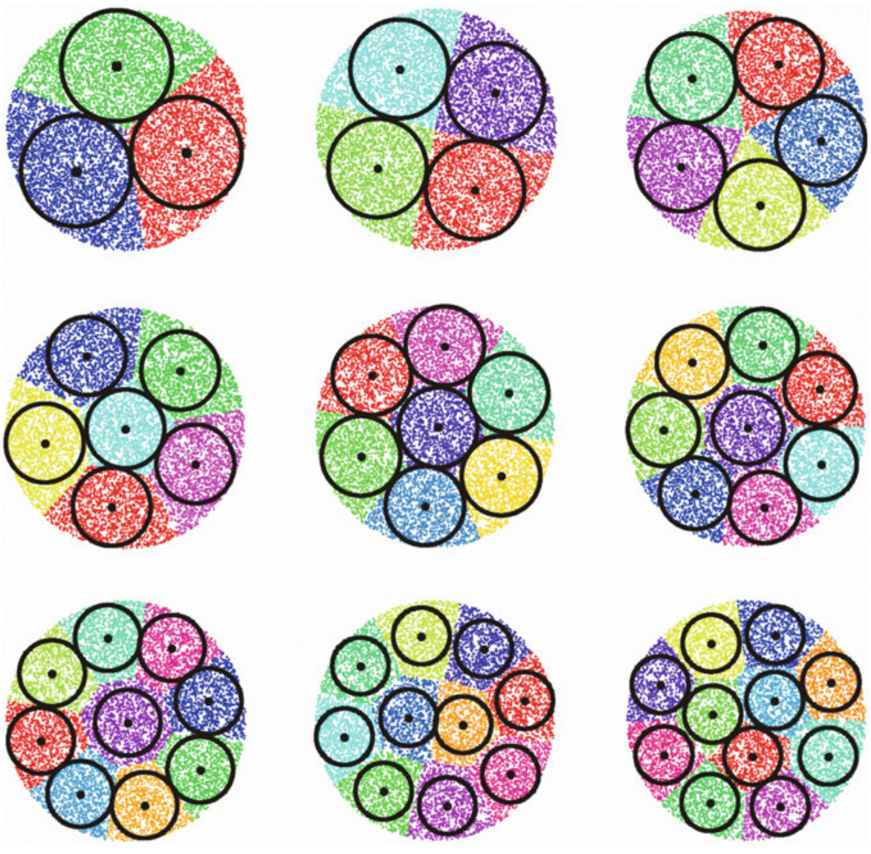


Fig. 1 Examples of circle packing for 3 up to 11 circles, the small dots represent points, which neural gas used for adaptation of the centers of circles

networks, where the more the result differs from the ideal output, the more are the weights adapted, and to SOM, where not only winning neuron changes its parameters, but its neighbors change them as well, even though to a lesser extent.

3 Experimental Results

The neural gas was used to pack from 3 up to 20 circles. As an illustrative example, Fig. 1 shows the results for packing from 3 up to 11 circles. The points within a circumscribing circle, which were randomly generated and used for adaptation of the neural gas are marked by their hue as to which of the centers of packed circles

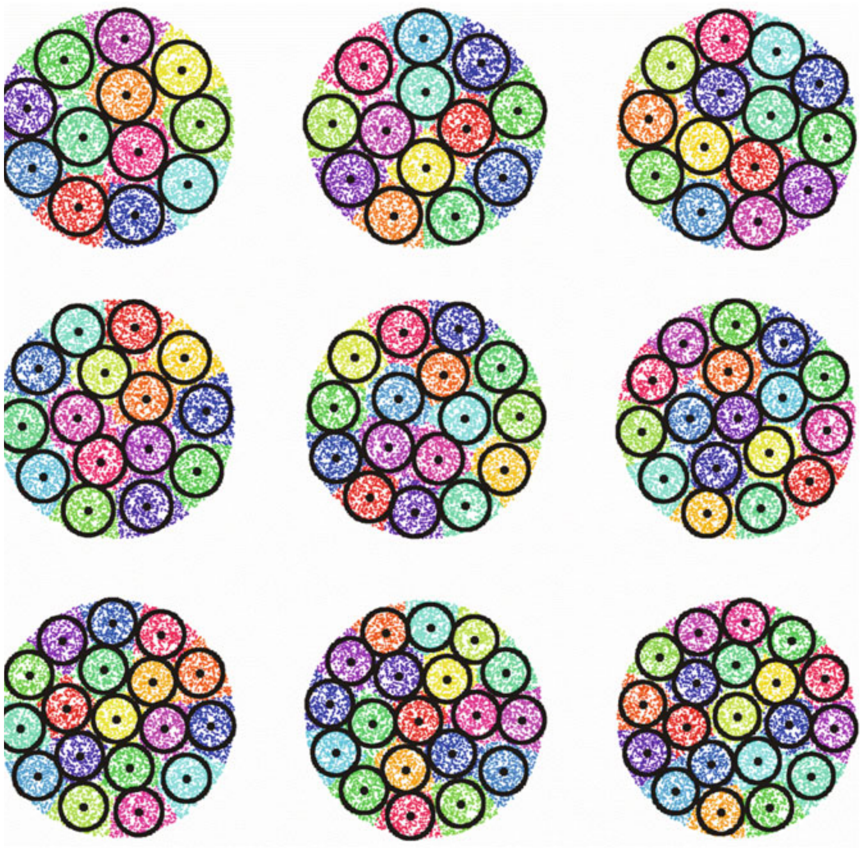


Fig. 2 Examples of circle packing for 12 up to 20 circles, the small dots represent points, which neural gas used for adaptation of the centers of circles

they finally belong. The examples for 3–9 circles are highly symmetrical, but it is not always the case – the symmetrical cases can be usually solved exactly by purely mathematical approach. As you can see in the last two cases for 10 and 11 packed circles, these cases are not so much symmetrical and they also show the greatest deviation from ideal packing. One can see in the example for packing 10 circles, that the sets of points belonging to the central two circles have pentagonal shapes (corresponding to Voronoi cells), but the sides of the pentagons are unequal, edges in the centre are much longer. The points along the longer edges pull the centers of the central two circles too close; therefore they do not touch the "outer" packed circles.

Sometimes, there can be two or more solutions of the packing of the same quality, e.g. 6, 11 and 13 packed circles have 2 possible optimal solutions, and 18 packed circles have even 3 optimal solutions. However, neural gas mostly produces only one of them, like those shown in Figs. 1 and 2. Basic placements of all the solutions produced by neural gas roughly corresponds to optimal ones, save for 17 packed circles, where in Fig. 2 the last case in the second row should have 6 inner packed circles surrounding the central one, not 5. Any solution may also be rotated or reflected, so there are formally many equivalent optimal solutions, in fact a continuum of them. In the examples provided in Figs. 1 and 2 the circles do not always touch their neighbors. In our case those are mostly imperfections of our method. Even though there can be sometimes circles that do not have to touch their

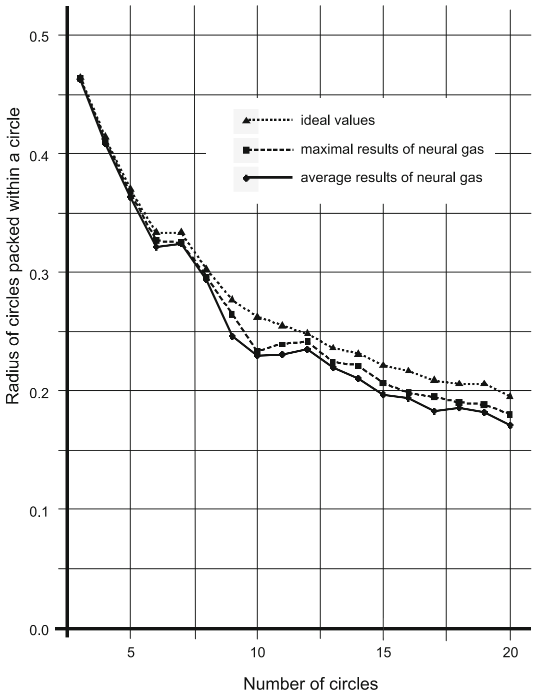


Fig. 3 Results of neural gas packing compared to ideal values

neighbors, in optimal solutions for N up to 20 circles such cases occur only for one circle placed in the centre for $N = 8$, 9 and 20.

In Fig. 3 you can see three graphs, ideal values of radius depending on number of packed circles $N = 3\ldots20$, the best of 10 results for each value N found by neural gas and the averages of those 10 trials.

To provide the neural gas with enough training data, in the previous figures we used 10000 randomly generated points within the circumscribing circle. To estimate, if such a great number was necessary, we also collected data for number of randomly generated points ranging from 100 up to 12800, always doubling their amount. The results from 100 repeated measurements are presented as a box plot showing interquartile range as a box with median as thick line, whiskers showing lowest datum still within 1.5 IQR of the lower quartile, and the highest datum still within 1.5 IQR of the upper quartile, and circles presenting outliers. The Fig. 4 shows that 10000 training points were not an extreme amount, even though a tenth of this amount would also provide reasonable results.

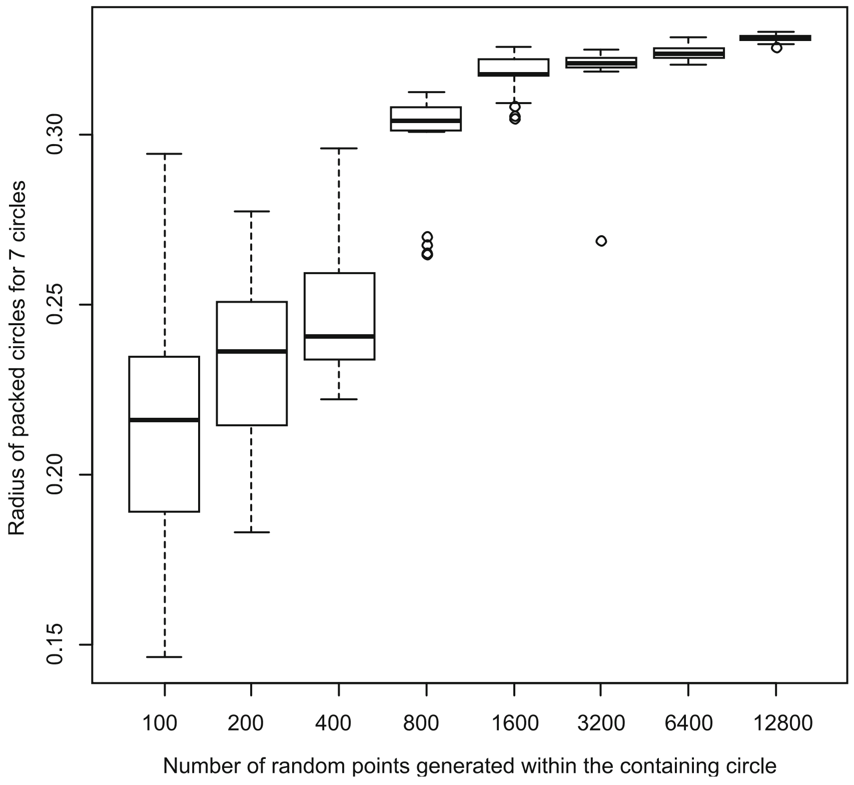


Fig. 4 Box plot showing the performance against the amount of training data

4 Conclusions

The neural gas provided acceptable solutions for the packing problem, even though the method was not designed or ever before used for such a purpose. Although only packing within a circle was tested, other types of boundaries and packing in 3D space can be expected to provide acceptable results as well. The method was used only to prove the possibility of its application. Its effectiveness is far beyond the many dedicated heuristics, but feasibility of its application may improve, if we would require packing within less regular boundaries. Future improvements of this type of algorithm might employ search for an empty space within the containing boundary, but outside the packed circle, where future points used to shift neural gas centers should be concentrating. Since such a space would be highly irregular, a stochastic kind of exploration would be needed; the most promising such method can be Rapidly Exploring Random Tree method [1, 11].

Acknowledgments This contribution was supported by Grant Agency VEGA SR under the grant 1/0458/13.

References

1. Abbadi, A., Matoušek, R.: RRTs review and statistical analysis. Int. J. Math. Comput. Simul. **6**(1), 1–8 (2012)
2. Al-Modahka, I., Hifi, M., M'Hallah, R.: Packing circles in the smallest circle: an adaptive hybrid algorithm. J. Oper. Res. Soc. **62**(11), 1917–1930 (2011)
3. Carrabs, F., Cerrone, C., Cerulli, R.: A Tabu search approach for the circle packing problem. In: Network-Based Information Systems (NBiS), pp. 165–171 (2014)
4. Castillo, I., Kampas, F.J., Pintér, J.D.: Solving circle packing problems by global optimization: numerical results and industrial applications. Eur. J. Oper. Res. **191**(3), 786–802 (2008)
5. Conway, J.H., Sloane, N.J.: Sphere Packings, Lattices, and Groups. Springer, New York (1987)
6. Flores, J.J., Martínez, J., Calderón, F.: Evolutionary computation solutions to the circle packing problem. Soft Comput. (2015). ISSN 1433-7479. doi:10.1007/s00500-015-1603-y
7. Hifi, M., M'Hallah, R.: A literature review on circle and sphere packing problems: models and methodologies. Adv. Oper. Res. (2009) Article ID 150624, 22 pp. doi:10.1155/2009/150624
8. Hopfield, J.J., Tank, D.W.: "Neural" computation of decisions in optimization problems. Biol. Cybern. **52**(3), 141–152 (1985)
9. Huang, W., Fu, Z., Xu, R.: Tabu search algorithm combined with global perturbation for packing arbitrary sized circles into a circular container. China Inf. Sci. **56**(9), 1–14 (2013)
10. Hurdal, K.M., Stephenson, K.: Cortical cartography using the discrete conformal approach of circle packings. NeuroImage **23**(Suppl 1), S119–28 (2004)
11. LaValle, S M., Kuffner. Jr. J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. (IJRR) **20**(5) (2001). doi:10.1177/02783640122067453
12. Martinetz, T., Schulten, K.: A "neural gas" network learns topologies. In: Kohonen, T. et al. (eds): Artificial Neural Networks. Elsevier, Amsterdam, pp. 397–402 (1991)
13. Shi, Y.-J., Liu, Z.-C., Ma, S.: An improved evolution strategy for constrained circle packing problem. In: Advanced Intelligent Computing Theories and Applications. Springer, Berlin, pp. 86–93 (2010)

14. Specht E.: Packomania web site. http://www.packomania.com/, 1999. last visit 2.4.2015
15. Stephenson, K.: Introduction to Circle Packing: The Theory of Discrete Analytic Functions. Cambridge University Press, Cambridge (2005)
16. Szabó, P.G., Markót, M.C., Csendes, T., Specht, E., Casado, L.G., García, I.: New Approaches to Circle Packing in a Square: With Program Codes. In: Springer Optimization and Its Applications, Vol. 6. New York, NY, USA (2007)
17. Weisstein, E.W.: Circle Packing http://mathworld.wolfram.com/CirclePacking.html. Accessed 2 April 2015
18. Yan-Jun, S., Yi-Shou, W., Long, W., Hong-Fei, T.: A layout pattern based particle swarm optimization for constrained packing problems. Inf. Technol. J. **11**, 1722–1729 (2012)
19. Zhang, D.F., Deng, A.S.: An effective hybrid algorithm for the problem of packing circles into a larger containing circle. Comput. Oper. Res. **32**(8), 1941–1951 (2005)
20. Zimmermann, A.: Al Zimmermann's Programming Contests—Circle Packing, http://recmath.com/contest/CirclePacking/index.php. Accessed 2 April 2015

Cost Functions Based on Different Types of Distance Measurements for Pseudo Neural Network Synthesis

Zuzana Kominkova Oplatkova and Roman Senkerik

Abstract This research deals with a novel approach to classification. New classifiers are synthesized as a complex structure via evolutionary symbolic computation techniques. Compared to previous research, this paper synthesizes multi-input-multi-output (MIMO) classifiers with different cost function based on distance measurements. An inspiration for this work came from the field of artificial neural networks (ANN). The proposed technique creates a relation between inputs and outputs as a whole structure together with numerical values which could be observed as weights in ANN. Distances used in cost functions were: Manhattan (absolute distances of output vectors), Euclidean, Chebyshev (maximum distance value), Canberra distance, Bray – Curtis. The Analytic Programming (AP) was utilized as the tool of synthesis by means of the evolutionary symbolic regression. For experimentation, Differential Evolution for the main procedure and also for meta-evolution version of analytic programming was used Iris data (a known benchmark for classifiers) was used for testing of the proposed method.

Keywords Pseudo neural networks • Symbolic regression • Classification • Euclidean distance • Chebyshev distance • Manhattan distance

1 Introduction

This paper deals with a novel method for classification problems, which is based on symbolic regression with evolutionary optimization techniques, namely Analytic Programming [1–5] and Differential evolution [6]. The symbolic regression is able

Z.K. Oplatkova (✉) · R. Senkerik
Faculty of Applied Informatics, Tomas Bata University in Zlin,
Nam. T. G. Masaryka 5555, 760 01 Zlin, Czech Republic
e-mail: oplatkova@fai.utb.cz

R. Senkerik
e-mail: senkerik@fai.utb.cz

to synthesize a complex structure which is optimized by means of evolutionary computation.

The basic case of symbolic regression in the context of evolutionary computation represents a process in which the measured data is fitted and a suitable mathematical formula is synthesized in an analytical way. This process is widely known for mathematicians. They use this process when a need arises for mathematical model of unknown data, i.e. relation between input and output values. The proposed technique is similar to synthesis of analytical form of mathematical model between input and output(s) in training set. Such a structure can be used as a classifier because it can simulate the behaviour of the Artificial Neural Networks (ANN) [7–10], where the inspiration for this work came from. The novel method is called Pseudo neural networks.

Artificial neural networks are based on some relation between inputs and output(s), which utilizes mathematical transfer functions and optimized weights from training process. The training process in ANN means to optimize weight vector for all training patterns to classify correctly according to the required output value. This means that the output error function based on weights is minimized. ANN uses Euclidean or Manhattan function as the usual error function. In the case of Pseudo neural networks, there is no optimization of number of nodes, connections or transfer function in nodes. The training is done by means of optimization procedure in evolutionary symbolic regression on the basis of output error function. The obtained structure is not possible to redraw to a pure ANN structure of nodes and connections.

The first simulation with Pseudo neural networks worked with Manhattan error function (absolute value of difference between actual and required values). As authors have experiences in the field of evolutionary optimization algorithms, they know that the shape and complexity of the cost function affect the final optimization process very much. Therefore the research with other distance measurements in the cost functions were enabled and implemented to observe the speed of convergence to the optimal solution. The other distances include Euclidean, Chebyshev, Bray-Curtis and Canberra measurements.

This paper uses Analytic Programming (AP) [1–5] for evolutionary symbolic regression procedure. Besides AP, other techniques for symbolic regression computation can be found in literature, e.g. Genetic Programming (GP) introduced by John Koza [14, 15] or Grammatical Evolution (GE) developed by Conor Ryan [16].

The above-described tools were recently commonly used for synthesis of artificial neural networks but in a different manner than is presented here. One possibility is the usage of evolutionary algorithms for optimization of weights to obtain the ANN training process with a small or no training error result. Some other approaches represent the special ways of encoding the structure of the ANN either into the individuals of evolutionary algorithms or into the tools like Genetic Programming. But all of these methods are still working with the classical terminology and separation of ANN to neurons and their transfer functions [11].

In this paper, iris plant dataset [17, 18] was used as a benchmark case for classification, which has been introduced by Fisher [17] for the first time. It is a well known dataset with 4 features and 3 classes. The attributes consist of sepal length, sepal width, petal length and petal width, which divides the plants into iris virginica, iris versicolor and iris sentosa. The data set was analysed in a lot of papers by means of supervised and unsupervised neural networks [19–22], variations like distribution based ANN [23], piecewise linear classifier [24] or rough sets [25]. Not only pure ANN were used for classification but also evolutionary algorithms connected with fuzzy theory were employed [26, 27]. The tool from symbolic regression called Gene expression programming was used for classification too [28]. The last mentioned tool was used as a classifier, which contain procedures if-then rules in the evolutionary process. The basic components consist of greater then, less then, equal to, etc. and pointers to attributes.

The proposed technique in this paper is different. It synthesizes the structure without a prior knowledge of transfer functions and inner potentials. It synthesizes the relation between inputs and output of training set items used in neural networks so that the items of each group are correctly classified according the rules for cost function value.

This paper uses a MIMO approach (Multi Input Multi Output) and different kinds of output error based on different distances. It synthesizes more expressions with relation between input and each output separately. Also in the case of artificial neural networks, it can be obtained more expressions between inputs and each output.

Firstly, different distance measurements are depicted in equations and figures. The following section contains Analytic Programming description. After that a brief description of artificial neural networks (ANN) continues. The result section and conclusion finish the paper.

2 Distance Measurements

The paper uses 5 distances which were implemented into the cost function according to which the Pseudo neural networks are created. The basic cost function value is given in Eq. (1). This distance is called Manhattan [31].

$$cv = \sum_{i=1}^{n} |A_i - B_i| \tag{1}$$

where

A_i – required output vector component
B_i – actual current output vector component from the complex structure

Other distance measurements follow in Eq. (2) Euclidean distance, (3) Chebyshev distance or Maximum value distance, (4) Canberra distance and (5) Bray-Curtis distance [32]. There are also the other exotic metrics which can be usable [32].

$$cv = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2} \tag{2}$$

$$cv = \max_i (|A_i - B_i|) \tag{3}$$

$$cv = \sum_{i=1}^{n} \frac{|A_i - B_i|}{|A_i| + |B_i|} \tag{4}$$

$$cv = \frac{\sum_{i=1}^{n} |A_i - B_i|}{\sum_{i=1}^{n} (A_i + B_i)} \tag{5}$$

To have an idea about the complexity and dynamics of the error functions an artificial case was prepared (Fig. 1). There are 4 points divided into two classes.

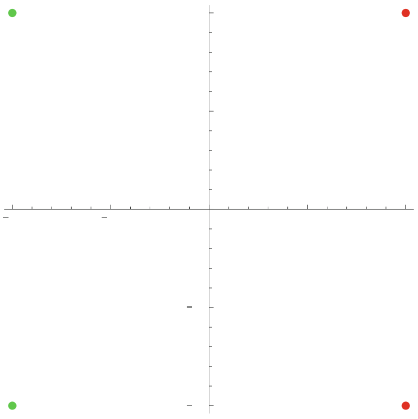


Fig. 1 Artificial classification data case

If one neuron would be used for classification of all 4 points with sigmoid function without the bias, the dependences of the error functions on weights are depicted in following figures (Figs. 2, 3, 4 and 5).

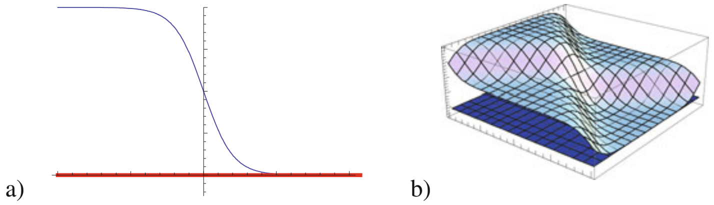


Fig. 2 Manhattan – classical absolute value of output vector differences (a) 2D function (one input and one weight) – the red line is for the distance equal to zero, (b) 3D function (two inputs and two weights) – the blue plane is for the distance equal to zero

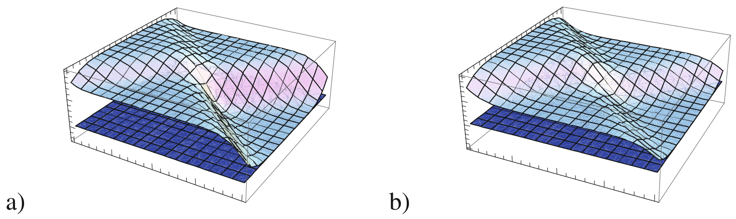


Fig. 3 (a) Euclidean distance, (b) Bray – Curtis distance – maximal distance value, the blue plane is for the distance equal to zero

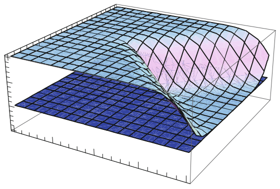


Fig. 4 Chebyshev – the blue plane is for the distance equal to zero

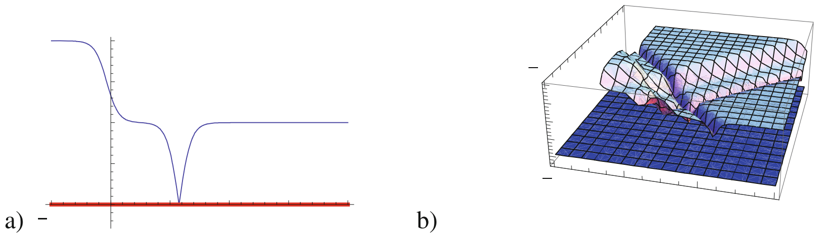


Fig. 5 Canberra distance – (a) 2D – the red line is for the distance equal to zero, (b) 3D – the blue plane is for the distance equal to zero

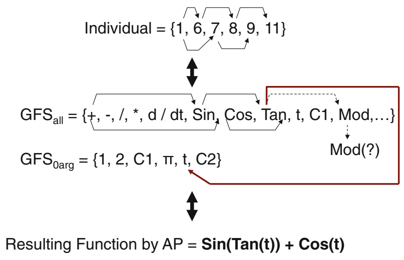
It can be visible from graphs that Euclidean, Manhattan and Bray-Curtis distances are almost the same shape only the output interval is different. In all cases, there is visible an area where combinations of weights can be used to approach the correct training with error equal to zero. Chebyshev and Canberra are different. Canberra is the most complicated and full of local optimas. It can be believed that the Canberra will not work satisfactory for Pseudo neural networks and AP. The Chebyshev distance seems to be good for the purpose of Pseudo neural networks. Unfortunately, simulations proved the opposite. The reason is caused by the different visualization. The graphs use the dependence of the distance on weights. On the contrary, Pseudo neural networks have the structure synthesized completely also with coefficients which are only similar to weights.

3 Analytic Programming

This tool was used for the synthesis of a complex structure which can serve similarly as a supervised ANN and classify items from the training set into specified groups. Basic principles of the AP were developed in 2001 [1–4]. Together with genetic programming (GP) [8, 9] and grammatical evolution (GE) [10], AP belongs to the group of evolutionary symbolic regression tools. GP uses genetic algorithms while AP can be used with any evolutionary algorithm, independently on individual representation.

The core of AP is based on a usage of the set of functions, operators and terminals. This set of variables is usually mixed together and consists of functions with different number of arguments. The structure of so called General functional set (GFS) is created by subsets of functions according to the number of their arguments. For example GFSall is a set of all functions, operators and terminals, GFS3arg is a subset containing functions with only three arguments, GFS0arg represents only terminals, etc. The subset structure presence in GFS is important for AP as it avoids the synthesis of pathological programs, i.e. programs containing functions without arguments, etc. The content of GFS is dependent only on the user [1] and the solved problem.

Fig. 6 Main principles of AP

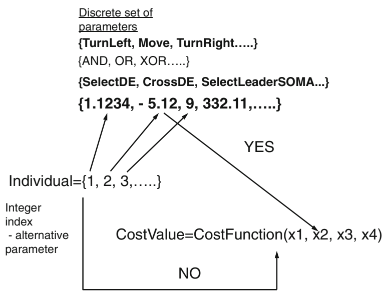

Individual = {1, 6, 7, 8, 9, 11}
GFS_all = {+, -, /, *, d / dt, Sin, Cos, Tan, t, C1, Mod,...}
Mod(?)
GFS_0arg = {1, 2, C1, π, t, C2}
Resulting Function by AP = Sin(Tan(t)) + Cos(t)

The second part of the AP core is a sequence of mathematical operations, which are used for the program synthesis. These operations are used to map an individual of a population into a suitable program, in this case into a suitable complex structure of mathematical functions and operators (Fig. 6). This transformation or mapping consists of two main parts. The first part is called discrete set handling (DSH) [1–5] (Fig. 7) and the second one stands for security procedures which do not allow synthesizing pathological programs. The method of DSH allows handling arbitrary objects and defined functions, not only in AP.

AP needs some evolutionary algorithm that consists of population of individuals for its run. This research used one evolutionary algorithm: Differential Evolution (DE) [16] for main process and also meta-evolutionary process in AP.

AP exists in 3 versions – basic without constant estimation, AP_{nf} – estimation by means of nonlinear fitting package in Mathematica environment (www.wolfram.com) and AP_{meta} – constant estimation by means of another evolutionary algorithms; meta means meta-evolution. AP_{meta} – strategy was used in the case of Pseudo neural network synthesis. The detailed description of AP can be found in [1, 2].

Fig. 7 Discrete set handling


Discrete set of parameters
{TurnLeft, Move, TurnRight.....}
{AND, OR, XOR.....}
{SelectDE, CrossDE, SelectLeaderSOMA...}
{1.1234, - 5.12, 9, 332.11,.....}
YES
Individual={1, 2, 3,.....}
Integer index - alternative parameter
CostValue=CostFunction(x1, x2, x3, x4)
NO

4 Artificial Neural Networks

Artificial neural networks are inspired in the biological neural nets and are used for complex and difficult tasks [7–11]. The most often usage is classification of objects as also in this case. ANNs are capable of generalization and hence the classification is natural for them. Some other possibilities are in pattern recognition, control, filtering of signals and also data approximation and others.

There are several kinds of ANN. Simulations were based on similarity with feedforward net with supervision. ANN needs a training set of known solutions to be learned on them. Supervised ANN has to have input(s) and also required output (s).

The neural network works so that suitable inputs in numbers have to be given on the input nodes. These inputs are multiplied by weights which are adjusted during the training. In the neuron the sum of inputs multiplied by weights are transferred through mathematical function like sigmoid, linear, hyperbolic tangent etc. These single neuron units (Fig. 3) are connected to different structures to obtain different structures of ANN (e.g. Fig. 4), where $\sum\delta = TF[\sum(w_i x_i + bw_b)]$ and $\sum = TF[\sum(w_i x_i + bw_b)]$; TF is for example logistic sigmoid function.

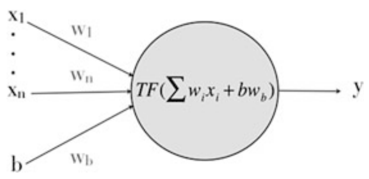


Fig. 8 Neuron model, where TF (transfer function like sigmoid), x_1 - x_n (inputs to neural network), b – bias (usually equal to (1), w_1 – w_n, w_b – weights, y – output

The example of relation between inputs and output can be shown as a mathematical form (1). It represents the case of only one neuron and logistic sigmoid function as a transfer function (Figs. 8 and 9).

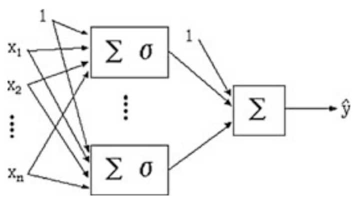


Fig. 9 ANN models with one hidden layer

$$y = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2)}}, \qquad (6)$$

where

y – output
x_1, x_2 – inputs
w_1, w_2 – weights.

The aim of the proposed technique is to find similar relation to (1). This relation is completely synthesized by evolutionary symbolic regression – Analytic Programming.

5 Problem and Iris Dataset Description

For this classification problem, iris plant dataset [17, 18] was used as a benchmark case for classification. It is a well known dataset with 4 features and 3 classes. The attributes consist of sepal length, sepal width, petal length and petal width, which divides the plants into iris *virginica*, iris *versicolor* and iris *sentosa*. This set contains 150 instances. Half amount was used as training data and the second half was used as testing data. The cross validation is planned for future testing. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. The attributes were of real values.

Compared to previous research, this paper uses MIMO approach together with different type of distance measurements in cost function in AP described in Sect. 2. An expression for each output node was synthesized. The data for each simulation was used as follow: first node – data for iris setosa as value higher than 0.5 which was saturated to 1 and the other two groups as data with saturated zero value. The second node expression divided similarly data for iris versicolor as saturated value 1 and the other two as value zero. The last node was the same for iris virginica. The final combination of the output node values gives the same result as in ANN approach.

6 Results

The simulations were carried out with Analytic Programming and Differential Evolution. Settings of EA parameters for both processes (main and metaevolution) were based on performed numerous experiments with chaotic systems and simulations with AP_{meta} (Tables 1 and 2), where CFE means cost function evaluations.

Table 1 DE settings for main process of AP

Pop size	40
F	0.8
Cr	0.8
Generations	50
Max. CFE	2000

Table 2 DE settings for meta-evolution

Pop size	40
F	0.8
Cr	0.8
Generations	150
Max. CFE	6000

The set of elementary functions for AP was inspired in the items used in classical artificial neural nets. The components of input vector x contain values of each attribute (x1, x2, x3, x4). Thus AP dataset consists only of simple functions with two arguments and functions with zero arguments, i.e. terminals. Functions with one argument, e.g. Sin, Cos, etc., were not applied in the case of this paper.

Basic set of elementary functions for AP:

GFS2arg = + , -, /, *, ^, exp
GFS0arg = x_1, x_2, x_3, x_4, K

Total number of cost function evaluations for AP was 1000, for the second EA it was 6000, together 6 millions per each simulation.

Carried simulations proved that Canberra and Chebyshev distances were not satisfactory for Pseudo neural nets synthesis with iris data set. The Chebyshev distance equal to 1 produced structures which had 25 misclassified items which is unacceptable. The Manhattan, Euclidean and Bray-Curtis were useful but none improved the speed of convergence. The notations which works with training error equal to 2 misclassified items (it means 2.66 % error, 97.34 % success from all 75 training patterns) were synthesized for the first and third output node during the first or second generation, for the second node within 10th generation in average.

The output expression can be mixed. The user can choose the suitable shape which is less complicated then others. The suitable node output should be selected also on the basis of the testing error. In some cases the error was equal to 5–7 misclassified items (8 % error, 92 % success).

The obtained training and testing errors are comparable with errors obtained within other approaches as artificial neural networks and others. But other tests with other benchmarks are necessary to carry out.

7 Conclusion

This paper deals with a novel approach – pseudo neural networks. Within this approach, classical optimization of the structure or weights was not performed. The proposed technique is based on symbolic regression with evolutionary computation. It synthesizes a whole structure in symbolic form without a prior knowledge of the ANN structure or transfer functions. It means that the relation between inputs and output(s) is synthesized which serves as a classifier. In the case of this paper, expressions for each output node were synthesized separately and independently based on different types of distance measurement. The simulations showed that Chebyshev and Canberra distances are not suitable for Pseudo neural network synthesis. Euclidean, Manhattan and Bray-Curtis are suitable according to these preliminary results. For further tests some observed critical points have to be taken into consideration. Huge testing with other benchmarks are also in future plans.

Acknowledgment This work was supported by Grant Agency of the Czech Republic - GACR P103/15/06700S, further by financial support of research project NPU I No. MSMT-7778/2014 by the Ministry of Education of the Czech Republic and also by the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

References

1. Zelinka, et al.: Analytical programming - a novel approach for evolutionary synthesis of symbolic structures. In: Kita, E. (ed.) Evolutionary Algorithms, InTech (2011)
2. Oplatkova, Z.: Metaevolution: Synthesis of Optimization Algorithms by means of Symbolic Regression and Evolutionary Algorithms, Lambert Academic Publishing Saarbrücken (2009)
3. Zelinka, I., Varacha, P., Oplatkova, Z.: Evolutionary synthesis of neural network. In: Proceedings of 12th International Conference on Soft Computing – MENDEL 2006. Mendel series, vol. 2006, pp. 25 – 31. Brno, Czech Republic (2006). ISSN: 1803- 3814
4. Zelinka, I.,Oplatkova, Z., Nolle, L.: Boolean symmetry function synthesis by means of arbitrary evolutionary algorithms-comparative study. In: International Journal of Simulation Systems, Science and Technology, vol. 6, pp. 44–56, 9 Aug 2005. ISSN: 1473-8031
5. Lampinen, J., Zelinka, I.: New Ideas in Optimization—Mechanical Engineering Design Optimization by Differential Evolution, vol. 1, 20 p. McGraw-hill, London (1999). ISBN 007-709506-5
6. Gurney, K.: An Introduction to Neural Networks. CRC Press (1997). ISBN: 1857285034
7. Hertz, J., Kogh, A., Palmer, R.G.: Introduction to the Theory of Neural Computation. Addison – Wesley (1991)
8. Wasserman, P.D.: Neural Computing: Theory and Practice. Coriolis Group (1980). ISBN: 0442207433

9. Fausett, L.V.: Fundamentals of Neural Networks: Architectures, Algorithms and Applications. Prentice Hall (1993). ISBN: 9780133341867
10. Volna, E., Kotyrba, M., Jarusek, R.: Multiclassifier based on Elliott wave's recognition. Comput. Math. Appl. **66** (2013). doi:10.1016/j.camwa.2013.01.012
11. Fekiac, J., Zelinka, I., Burguillo, J.C.: A Review of Methods for Encoding Neural Network Topologies in Evolutionary Computation. ECMS 2011, Krakow, Poland. ISBN: 978-0-9564944-3-6
12. Back, T., Fogel, D.B., Michalewicz, Z.: Handbook of Evolutionary Algorithms. Oxford University Press (1997). ISBN 0750303921
13. Koza, J.R., et al.: Genetic Programming III; Darwinian Invention and problem Solving. Morgan Kaufmann Publisher (1999). ISBN 1-55860-543-6
14. Koza, J.R.: Genetic Programming. MIT Press (1998). ISBN 0-262-11189-6
15. O'Neill, M., Ryan, C.: Grammatical Evolution. Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers (2003). ISBN 1402074441
16. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugenics **7**(2), 179–188 (1936). doi:10.1111/j.1469-1809.1936.tb02137.x
17. Machine learning repository with Iris data set. http://archive.ics.uci.edu/ml/datasets/Iris
18. Swain, M., et al.: An approach for iris plant classification using neural network. Int. J. Soft Comput. **3**(1) (2012). doi:10.5121/ijsc.2012.3107
19. Shekhawat, P., Dhande, S.S.: Building and iris plant data classifier using neural network associative classification. Int. J. Adv. Technol. **2**(4), 491–506 (2011). ISSN: 0976-4860
20. Avci, M., Yildirim, T.: Microcontroller based neural network realization and iris plant classifier application. In: Proceedings of the Twelfth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN'2003), Canakkale, Turkey, 2–4 July 2003
21. Osselaer, S.: Iris data analysis using back propagation neural networks. J. Manufact. Syst. **13** (4), 262 (2003)
22. Chen, S., Fang, Y.: A new approach for handling iris data classification problem. Int. J. Appl. Sci. Eng. (2005). ISSN: 1727-2394
23. Kostin, A.: A simple and fast multi-class piecewise linear pattern classifier. Pattern Recogn. **39** (11), 1949–1962 (2006). doi:10.1016/j.patcog.2006.04.022. ISSN 0031-3203
24. Kim, D.: Data classification based on tolerant rough set. Pattern Recogn. **34**(8), 1613–1624 (2001). doi:10.1016/S0031-3203(00)00057-1. ISSN 0031-3203
25. Agustı´n-Blas, L.E., et al.: A new grouping genetic algorithm for clustering problems. Expert Syst. Appl. **39**(10) 9695–9703 (2012). doi:10.1016/j.eswa.2012.02.149. ISSN 0957-4174
26. Zhou, E., Khotanzad, A. Fuzzy classifier design using genetic algorithms. Pattern Recogn. **40** (12), 3401–3414 (2007). doi:10.1016/j.patcog.2007.03.028. ISSN 0031-3203
27. Ferreira, C.: Gene Expression Programming: Mathematical Modelling by an Artificial Intelligence. (2006). ISBN: 9729589054
28. Oplatkova, Z., Senkerik, R.: Evolutionary synthesis of complex structures – pseudo neural networks for the task of iris dataset classification. In: Zelinka, I., Chen, G., Rössler, O.E., Snasel, V., Abraham, A. (eds.) Nostradamus 2013: Prediction, Modeling and Analysis of Complex Systems, vol 210. Advances in Intelligent Systems and Computing, Springer International Publishing, pp. 211–220. doi:10.1007/978-3-319-00542-3_22
29. Kominkova Oplatkova, Z., Senkerik, R.: Lenses classification by means of pseudo neural networks – two approaches. In: MENDEL 2014 - 20th International conference on soft Computing. Brno, Czech Republic, pp. 397–401. ISSN 1803-3814. (2014)
30. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution : A Practical Approach to Global Optimization. (Natural Computing Series), Springer; 1 edition (2005)
31. Rohlf, F.J.: Brenner's Encyclopedia of Genetics (Second Edition) (2013)
32. Matousek, R., Karpisek, Z.: Exotic metrics for function approximation. In: Proceedings of 17th International Conference on Soft Computing – MENDEL 2011. Mendel series vol. 2011, pp. 560–566, Brno (2011), ISSN: 1803- 3814

Part III
Intelligent Image Processing, Bio-Inspired Robotics

Labyrinth Arrangement Analysis Based on Image Processing

Pavel Škrabánek

Abstract The approach described in this paper has been developed as a supporting tool for practical exercises based on a teaching aid; however, its application is far from being limited to be used only in the education process. The main task of the approach is extraction of information about structure of a working environment from color images and its transformation to an appropriate form suitable for path-planning. The working environment considered in this paper is a labyrinth. Since the information is to be used further by path-planning, its expression in the form of a graph is required, or, more precisely, the adjacency matrix is required as the output. The transformation of the real world into a discreet representation is based on the exact cell decomposition in this approach. The approach employs the well-known image processing algorithms; however, the procedure of the labyrinth layout analysis as well as the transformation of the acquired information into the adjacency matrix has been developed and tested in this work. The approach has been realized and verified in MATLAB using the Image Processing Toolbox.

Keywords Image processing · Teaching aid · Mobile robot · Path-planning · Graph · MATLAB · Labyrinth building kit

1 Introduction

The artificial intelligence (AI) is growing in importance in the last decade which has been reflected in education of the undergraduate and graduate students, too. Since theoretical oriented lectures do not suffice to ensure the required high quality of education, they are usually complemented by practical exercises.

The practical exercises of AI courses can have many different forms; however, the use of mobile robots has become to be very trendy, as confirms the number of

P. Škrabánek (✉)
Faculty of Electrical Engineering and Informatics, University of Pardubice,
Studentská 95, 532 10 Pardubice, Czech Republic
e-mail: pavel.skrabanek@upce.cz

publications. Commercial products are often used for this purpose. In this context, the very popular LEGO Mindstorms should be mentioned [1, 2, 6]. Nevertheless, design of solutions according to the curriculum requirements are not rare [8, 10]. The development of a proper teaching aid is usually time-consuming; however, the final solution exactly matches with the needs of teaching.

In our case, the teaching requirements were also the main motivation to develop a proper multi-objective teaching aid for AI courses. The aid consists of differential wheeled mobile robots, a camera system and a labyrinth building kit, among others. The aid allows creating of a variety of different practical exercises focused on different topics of AI, such as path-planning, heuristic optimization or decision making. However, the teaching aid cannot be used by students without an appropriate software pack.

In the case of path-planning exercises, the software has to provide the information about the working area of the mobile robots, among others. The source of the information is the camera system in this case, thus the analysis of captured images followed by an appropriate transformation has to be performed. Since the mentioned tasks are not simple and their solving is time consuming, the developed approach has been introduced in this paper to be an inspiration for others.

The approach has been developed according to path-planning requirements therefore a brief reminder of the path-planning is given in Sect. 1.1. Since the approach is linked up to the technical solution of the labyrinth building kit and the camera system, their introduction is outlined in Sect. 1.2. The description of the approach is in Sect. 2.

1.1 Basis of the Path-Planning

The aim of the path-planning process in mobile robotics is to find a path between an initial state and a goal state; however, the path should avoid any collision with any forward known obstacle in the operating area of the robot.

Thus, the path-planning methods require at least a partial knowledge about the robot's working space where the knowledge has to be in an appropriate form. Most of the path-planning algorithms used in mobile robotics are based on the graph theory, thus the knowledge is usually required to be in the form of the graph.

The transformation of the real world to the graph is realized using some of the decomposition technics. The technics can be split up to three general groups labeled as roadmap [11], cell decomposition [11] and potential field [9]. An appropriate decomposition technic is usually selected according to a particular problem.

1.2 The Labyrinth Building Kit and Camera System

The design of the labyrinth components, as well as the camera system construction, play significant role in the described approach; however, the methodology of the labyrinth assembly and the installation of the camera system are equally important.

Since the topics are quite extensive, they have been split into two parts. The first part deals with the labyrinth building kit and related topics while the second one is dedicated to the camera system and its installation.

The Labyrinth Building Kit In the term of path-planning exercises, a labyrinth is a working space of mobile robots. Each labyrinth inherently contains many obstacles created by partitions. In the case of the teaching aid, the size of partitions are standardized, i.e. the length a_{pr} is 17 cm, the width b_{pr} is 1 cm and the height c_{pr} is 10 cm.

The color of the partitions is white except one edge (17 cm × 1 cm). This edge is of black color for most of the partitions except eight pieces of red color. The choice of the colors was made according to the image processing requirements.

A labyrinth is assembled using the partitions and posts of height 10 cm and of square ground plan of size 1 cm × 1 cm. The basis of each labyrinth is a floor of white color. The partitions are placed on the floor using the posts. Each vertical side of a post contains one track where a partition can be inserted thus a rectangular layout of the labyrinth is naturally created.

A large number of various labyrinths can be assembled using the kit; however, several condition are given to be kept, i.e. the outer shape of the labyrinth has to be rectangular, the labyrinth has to be a closed system (isolated from the surrounding world), and dimensions of the labyrinth should respect the technical limitations of the camera system. The labyrinth's inner layout is not limited but not each possible configuration makes sense.

The partitions are required to be inserted to posts so that the colored edge is visible from the top view. The red colored partitions are essential by analysis thus their placement has to match up to special requirements.

According to dimensions of the labyrinth, one or two red colored partitions are placed in each outer wall of the labyrinth. At least four red partitions should be used in the labyrinth to provide sufficient data for a correct analysis. Their placement has to satisfy two requirements; no two red partitions can be placed side by side, and no red partition can be placed next to a labyrinth corner. Their placement should be also irregular but balanced to minimize errors caused by an inaccurate camera placement and its lens distortion.

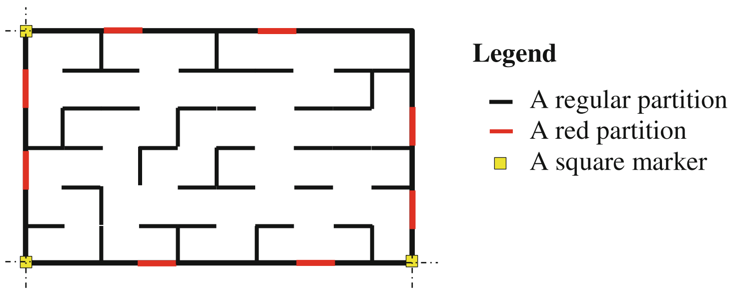


Fig. 1 An example of a correctly assembled labyrinth

Once the labyrinth has been assembled, three yellow square markers of the side length 3 cm are placed to the labyrinth's corners: two to the bottom corners and one to the left top corner. An example of a labyrinth meeting the requirements is in Fig. 1.

The Camera System The camera system consists of a stand arm and camera Logitech Webcam C930e. The camera has three bands (RGB) and it allows setting of several resolutions; however, the highest resolution (1080 × 1920 pixels) is used with in the exercises.

The camera is fixed on the stand arm and the whole camera system is placed such that camera is above the labyrinth and the focus of the lens is placed approximately in its middle. The arm should be parallel with the floor and the alignment of image edges should be parallel with the outer walls of the labyrinth. The camera's stand allows variable adjustment of the height. The altitude of the camera can vary from 1 m to 2 m. The altitude of the camera should be set so that the whole labyrinth is in the swath; however, minimum of its surrounding should be there. An example of a correct installation of the camera system is shown in Fig. 2.

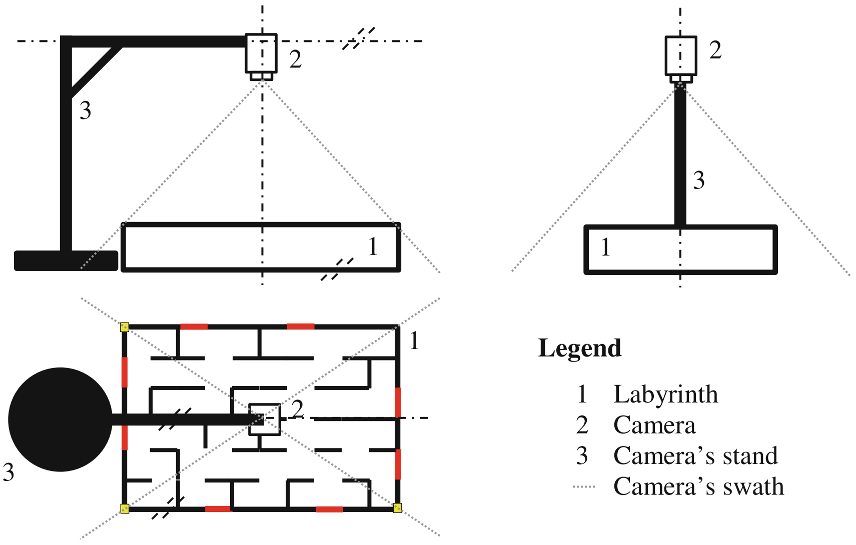


Fig. 2 An example of a correctly installed camera's system

2 The Analysis of a Labyrinth Layout

The required output of the analysis is the graph representation of the inner layout of an assembled labyrinth. Since a MATLAB version of the A* algorithm [12] is used within the exercises, the output of the analysis is the adjacency matrix [3]. Its input is an image of a completely assembled labyrinth captured by the camera; however, no other object can be placed in the view.

As was mentioned in Sect. 1.1, the decomposition of the real world to an appropriate discreet representation has to be performed. Considering the standardized dimensions of the partitions and the rectangular layout of labyrinths, the exact cell decomposition [11] seems to be the best choice. The size of each cell is than given by the dimensions of the partitions and borders of cells are placed on sites of the partitions. Thus, the result of the decomposition is a grid of equally sized square cells. The described decomposition is evident in Fig. 3 where the decomposition of the top left corner of the exemplary labyrinth is shown.

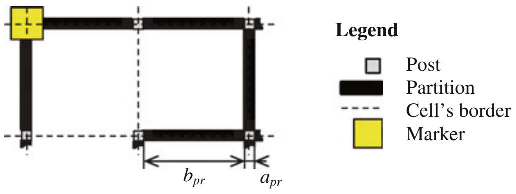


Fig. 3 The exact cell decomposition of the top left corner of the exemplary labyrinth

Naturally, the dimensions of the partitions on an image differ from the real one, thus they have to be determined than ever the decomposition can be performed. However, it is not the only thing to be solved before the analysis of the labyrinth layout can be realized.

Firstly, a captured image has to be transformed into a standardized form. This process is described in Sect. 2.2. The following step is the extraction of auxiliary information from the transformed image. This task is described in Sect. 2.3. The last step is the analysis of the labyrinth layout resulting in the adjacency matrix. This step is described in Sect. 2.4. Since the same image preprocessing has to be executed in the first and second steps, it is generally described in Sect. 2.1. The solution introduced in this section has been created using MATLAB and Image Processing Toolbox.

2.1 Image Preprocessing

Since the used camera has three bands, a captured image is saved in a three dimensional matrix, say **A**, of the size $m \times n \times 3$, where each layer contains data from one band, and n, m are given by the set resolution of the camera. The elements a of the matrix **A** are the numerical representation of the intensity where the intensity is expressed by an integer from $[0, 255]$.

The choice of the colors used in the labyrinth building kit was realized according to the image processing requirements, thus the image preprocessing is very simple. Its first step is the calculation of a 'difference' between two bands, say v and w, where $v, w \in \{1, 2, 3\}$, and $v \neq w$. Strictly speaking, the data in the layer w in the matrix $\mathbf{A}$ are subtracted from the data in the layer v; however, the elements of the resulting matrix must not be less than zero. This operation can be simply realized in MATLAB using the function `imapplymatrix` where the result is a two dimensional matrix, say $\mathbf{B^{vw}}$, of the size $m \times n$.

The second step is conversion of the intensity image $\mathbf{B^{vw}}$ to a binary image, say $\mathbf{C^{vw}}$, where the size of the resulting matrix $\mathbf{C^{vw}}$ is $m \times n$. It is realized using a threshold function which is represented in MATLAB by the function `im2bw` where its input parameter T is the threshold value, and $T \in [0, 1]$ (more in [4, 7]). This step of the image preprocessing is required by the function `regionprops` used in Sects. 2.2 and 2.3; however, the right setting of T prevents the resulting image $\mathbf{C^{vw}}$ from undesirable 'objects', too.

2.2 The Image Transformation

Despite all efforts, the installation of the camera system according to all the conditions given in Sect. 1.2 is usually not possible therefore the image transformation has to be performed before the next steps can be carried out.

The construction of the stand guarantees almost parallel alignment of the arm with the floor. Also the requirement to place the focus of the camera approximately into the middle of the labyrinth is usually fulfil. Thus, the stumbling-block is the parallelism of the labyrinth border with the image border.

The parallelism problem can be solved by rotation of an image; however, the picture angle θ has to be known. In the case of the labyrinth's images, the θ can be determined using the yellow square markers placed in the corners of the labyrinth, thus their positions in the image have to be identify.

This can be easily done in MATLAB using the function `regionprops` which can, among others, return coordinates of the centers of mass for each recognized object (more in [4, 7]). Since the function requires a binary image as the input, the image preprocessing has to be performed (see Sect. 2.1). Given that the color of the markers, the 'difference' between second and third band seems to be appropriate, i.e. $v = 2$ and $w = 3$, then, the suitable setting of the threshold value has proven to be $T = 0.5$.

The only highlighted objects in the resulting binary image $\mathbf{C^{23}}$ should be the markers, thus the function `regionprops` set to the mode 'Area' returns for $\mathbf{C^{23}}$ their coordinates of the centers of gravity $\left(x_{m_r}, y_{m_r}\right)$ where r is the index of a marker, and $r \in \{1, 2, 3\}$ (see Fig. 4).

Once the coordinates $\left(x_{m_r}, y_{m_r}\right)$ are known, the picture angle θ can be determined as

$$\theta = \arctan \frac{d_{my}}{d_{mx}}, \tag{1}$$

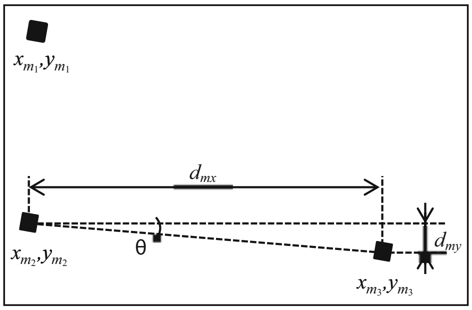


Fig. 4 An example of a resulting binary image with highlighted markers including the designation of the related variables

where $d_{my} = (y_{m_3} - y_{m_2})$, and $d_{mx} = (x_{m_3} - x_{m_2})$. The picture angle issue might be more evident in Fig. 4. The figure shows an example of a resulting binary image where the white squares are the highlighted markers.

Using the picture angle θ, the transformation of the original image $\mathbf{A}$ can be realized. In this work, the affine transformation with the inverse mapping approach has been used [5]. To complete the process of transformation, the bilinear interpolation [5] is applied to the rotated image. This process can be realized in MATLAB using one function called `imrotate` [7]. Let us denote the transformed image as $\tilde{\mathbf{A}}$ than the size of the matrix $\tilde{\mathbf{A}}$ is $m \times n \times 3$.

2.3 Extraction of Auxiliary Information

Before the analysis of the labyrinth layout can be performed, information about its localization and about its basic proportions has to be known. As the source of the information the transformed image $\tilde{\mathbf{A}}$ is used. The information about the labyrinth position and its proportions can be determined using the yellow square markers.

Thus, the image preprocessing is realized the same way as is described in Sect. 2; however, the transformed image $\tilde{\mathbf{A}}$ is used this time. The resulting binary image is saved in the matrix $\tilde{\mathbf{C}}^{\mathbf{23}}$. The coordinates of the centers of gravity of the markers in the transformed image $(x_{\tilde{m}_r}, y_{\tilde{m}_r})$ are obtained using the function `regionprops` set to the mode ‘Area’.

Let us suppose that the origin of the labyrinth is given by its left top corner, than its coordinates in the transformed image are $(x_{\tilde{m}_1}, y_{\tilde{m}_1})$. The length of the labyrinth in the image in the sense of x axis, l_x, and its length in the sense of y axis, l_y, are

$$l_x = x_{\tilde{m}_3} - x_{\tilde{m}_2}, \qquad l_y = y_{\tilde{m}_2} - y_{\tilde{m}_1}. \tag{2}$$

The lengths and the position of the labyrinth origin are not sufficient for the analysis execution. Also the proportions of the partitions in the image have to be known. For that purpose, the red partitions are included into the labyrinth building kit.

The data necessary for determining of the partition's proportions can be obtained using the function `regionprops` set to the mode 'BoundingBox' (more in [4, 7]). In this mode, the function finds the smallest rectangle containing an object in a binary image. Thus, the transformed image $\tilde{\mathbf{A}}$ has to be converted into the binary image again.

The conversion of $\tilde{\mathbf{A}}$ is realized according to Sect. 2.1; however, the 'difference' between first and third band is performed in this case, i.e. $v = 1$, $w = 3$. The optimal setting of the threshold T is 0.38. In the resulting binary image $\tilde{\mathbf{C}}^{\mathbf{13}}$, the red partitions and the yellow markers are highlighted; however, the markers are undesirable for the following process. Their elimination can be easily done by a simple matrix operation $\tilde{\mathbf{C}}^{\mathbf{w}} = \tilde{\mathbf{C}}^{\mathbf{13}} - \tilde{\mathbf{C}}^{\mathbf{23}}$.

Thus, the resulting binary image $\tilde{\mathbf{C}}^{\mathbf{w}}$ is used as the input of the function `regionprops` set to the mode 'BoundingBox'. The output of the function consist of the partition's dimensions where the greater value is the projection of a_{pr} to the image, say a_{pi}, while the smaller one is the projection of b_{pr}, say b_{pi}. Since several red partitions have been used by the labyrinth construction, the length of a cell in the image w_c is determined using their average values $\bar{a}_{pi}, \bar{b}_{pi}$, i.e.

$$w_c = \|\bar{a}_{pi} + \bar{b}_{pi}\|, \tag{3}$$

where $\|\|$ denotes rounding to the nearest integer. This notation is used also further.

2.4 Analysis of the Labyrinth Layout

The analysis of the labyrinth layout is realized in a loop when a window is stepwise moved from the left to the right side of an image, line by line, as is in a simplified form shown in Fig. 5. The step shift of the window corresponds to the length of the cell w_c. Since the initial position of the window is set to the left top corner of the labyrinth, the k-th cell is processed in the k-th step of the analysis. In each step of the analysis, the presence of partitions is examined. The acquired information is then used by the adjacency matrix construction. This description of the analysis process is, of course, very simplified. Its full description follows below.

Before the process of the analysis can be started, the number of rows, say n_y, as well as the number of columns, say n_x, of the grid created by the decomposition (see Fig. 7) has to be known. Since the dimensions of the labyrinth (2) as well as the size length of cells (3) are known, the parameters can be determined as

$$n_x = \left\| \frac{l_x}{w_c} \right\|, \qquad n_y = \left\| \frac{l_y}{w_c} \right\|. \tag{4}$$

As is illustrated in Fig. 5, the analysis is realized using a sliding window. The window is further represented by a square matrix $\mathbf{W}$ of the size $w_w \times w_w$ where

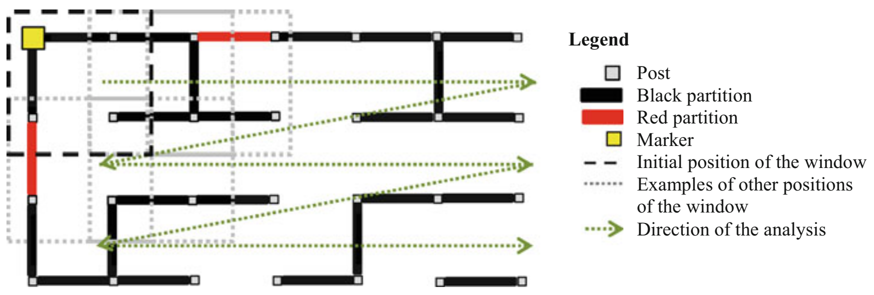

Legend
Post
Black partition
Red partition
Marker
Initial position of the window
Examples of other positions of the window
Direction of the analysis

Fig. 5 The mechanism of the analysis illustrated on the exemplary labyrinth

$$w_w = \|\lambda w_c\|, \tag{5}$$

and where λ is a number from $(1, 2]$, in our case $\lambda = 1.33$.

The analysis is not performed directly on the transformed image $\tilde{\mathbf{A}}$; however, a binary image is used. Thus, the transformed image $\tilde{\mathbf{A}}$ has to be processed by the function `im2bw` where the optimum threshold value is $T = 0.8$. The red and black partitions should be the only objects remaining in the resulting binary image $\tilde{\mathbf{C}}^{\mathbf{a}}$. Both type of partitions are represented by 0 in the binary image $\tilde{\mathbf{C}}^{\mathbf{a}}$; however, the rest of the labyrinth is represented by 1.

Since the analysis is performed on the binary image, the sliding window $\mathbf{W}$ should be defined using the matrix $\tilde{\mathbf{C}}^{\mathbf{a}}$, i.e. the content of the window in the k-th step of the analysis is

$$\mathbf{W}^{(k)} = \begin{pmatrix} \tilde{c}^a_{x_w^{(k)} y_w^{(k)}} & \tilde{c}^a_{\left(x_w^{(k)}+1\right) y_w^{(k)}} & \cdots & \tilde{c}^a_{\left(x_w^{(k)}+w_w\right) y_w^{(k)}} \\ \tilde{c}^a_{x_w^{(k)}\left(y_w^{(k)}+1\right)} & \tilde{c}^a_{\left(x_w^{(k)}+1\right)\left(y_w^{(k)}+1\right)} & \cdots & \tilde{c}^a_{\left(x_w^{(k)}+w_w\right)\left(y_w^{(k)}+1\right)} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{c}^a_{x_w^{(k)}\left(y_w^{(k)}+w_w\right)} & \tilde{c}^a_{\left(x_w^{(k)}+1\right)\left(y_w^{(k)}+w_w\right)} & \cdots & \tilde{c}^a_{\left(x_w^{(k)}+w_w\right)\left(y_w^{(k)}+w_w\right)} \end{pmatrix}, \tag{6}$$

where $\tilde{c}^a$ is an element of the matrix $\tilde{\mathbf{C}}^{\mathbf{a}}$, $k \in \{1, 2, \dots, (n_x n_y - 1)\}$. The origin of the source of data $\left(x_w^{(k)}, y_w^{(k)}\right)$ is for $k \in \left\{2, 3, \dots, (n_x n_y - 1)\right\}$ defined as

$$\left(x_w^{(k)}, y_w^{(k)}\right) = \begin{cases} \left(x_w^{(k-1)}, y_w^{(k-1)}\right) + \left(w_c, 0\right) & k \notin X, \\ \left(x_w^{(k-1)}, y_w^{(k-1)}\right) + \left(-n_x w_c, w_c\right) & k \in X, \end{cases} \tag{7}$$

where $X = \left\{n_x, 2n_x, \dots, (n_y - 1)n_x\right\}$; however, for $k = 1$ holds that

$$\left(x_w^{(1)}, y_w^{(1)}\right) = \left(\|x_{\tilde{m}_1}\|, \|y_{\tilde{m}_1}\|\right) + \left(x_o, y_o\right), \tag{8}$$

where $\left(x_o, y_o\right)$ is an offset of the window, where $x_o, y_o \in \mathbb{Z}$. In our case, the offset $\left(x_o, y_o\right) = (-5, -5)$ has been used.

The aim of the analysis is to determine all possibilities of mobile robot's movement within the labyrinth. Considering that the exact cell decomposition described on the beginning of Sect. 2 is used in this approach, the analysis problem can be reduced to detection of partitions.

Since the labyrinth is the closed word system and the initial position of the sliding window is placed to the left top corner of the labyrinth according to (8), only the presence of partitions on the right and the bottom sites in $\mathbf{W}^{(k)}$ has to be verified. For illustration, an example of a possible content of $\mathbf{W}^{(k)}$ is given in Fig. 6.

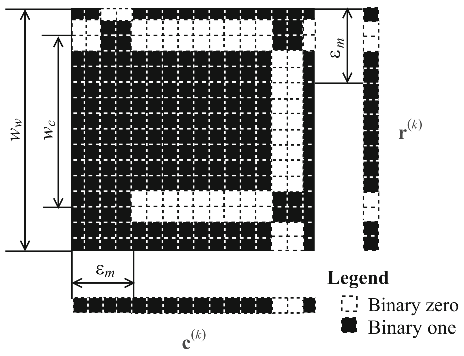


Fig. 6 Example of a possible content of $\mathbf{W}^{(k)}$ including auxiliary vectors $\mathbf{r}^{(k)}$ and $\mathbf{c}^{(k)}$

Presence of a partition on the bottom site is in the k-th step verified using an auxiliary vector $\mathbf{r}^{(k)}$ of the length w_w where its i-th element is

$$r_i^{(k)} = \left\| \frac{1}{w_w} \sum_{j=1}^{w_w} w_{ij}^{(k)} \right\|, \tag{9}$$

i.e. the i-th element of the vector $\mathbf{r}^{(k)}$ is the mean value of all elements of the matrix $\mathbf{W}^{(k)}$, $w^{(k)}$, placed on i-th row round towards the nearest integer.

The presence of a partition is confirmed by at least one zero value in the vector $\mathbf{r}^{(k)}$; however, only a part of the vector should be taken into account. The beginning of the area of interest is given by ε_m where its exact value can be chosen from $\varepsilon_m \in \{1, 2, \ldots, w_w\}$. In our case, $\varepsilon_m = \|0.2 w_w\|$ has been used.

Similarly, presence of a partition on the right site is judged using an auxiliary vector $\mathbf{c}^{(k)}$ of the length w_w in the k-th step of the analysis process. The i-th element of $\mathbf{c}^{(k)}$ is

$$c_i^{(k)} = \left\| \frac{1}{w_w} \sum_{j=1}^{w_w} w_{ji}^{(k)} \right\|. \tag{10}$$

Also in this case, only a part of the vector is used by the analysis. The beginning of the area of interest is given by ε_m, too.

As was stated above, the results of the analysis are required to be saved into the adjacent matrix, say **M**. The matrix **M** is created according to the results of the cell decomposition (see Fig. 7), where each cell represents a vertex of a graph and the possibility of transition between two adjacent cells is an edge of the graph; consequently, the dimension of **M** is given by the number of created cells, i.e. its size is $n_x n_y \times n_x n_y$.

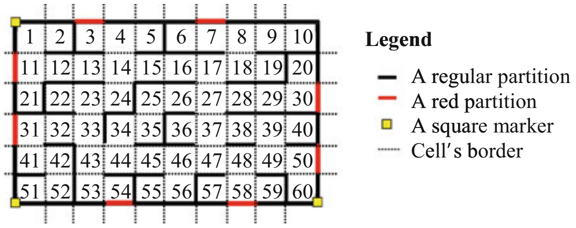


Fig. 7 The cell decomposition of the exemplary labyrinth including indices

The content of the matrix **M** is created according to the results of the analysis, i.e. using the auxiliary vectors $\mathbf{c}^{(k)}$ and $\mathbf{r}^{(k)}$. In the case, a partition is presented in the right or in the downwards direction, the transition in this and the reveres direction is not possible which is expressed by setting of the adequate elements m of the adjacency matrix **M** to 0, otherwise they are set to 1, i.e.

$$m_{k(k+1)} = m_{(k+1),k} = \begin{cases} 0 & \text{if } \exists c_i^{(k)} = 0, \\ 1 & \text{otherwise}, \end{cases} \tag{11}$$

and

$$m_{k(k+n_x)} = m_{(k+n_x)k} = \begin{cases} 0 & \text{if } \exists r_i^{(k)} = 0, \\ 1 & \text{otherwise}, \end{cases} \tag{12}$$

where $i \in \left\{\varepsilon_m, \varepsilon_m + 1, \ldots, w_w\right\}$.

Since the transition is possible only between two adjacent cells, it is obvious that the rest of elements of the matrix **M** are zeros.

3 Conclusion

The described approach has been successfully verified on several path-planning tasks using the A* algorithm [12]. Strictly speaking, the introduced approach was used as the source of information necessary for the solving of path-planning tasks. The transformation of the information from the image into the adjacency matrix was successful for all five labyrinths which differed in their layout as well as in their dimensions.

Even though that the introduced approach has been designed for the specific purpose, the technical solutions related to the image processing might be inspiration by development of similar teaching aids; however, the ideas can be used in many other applications exceeding the teaching issue.

Acknowledgments I would like to offer my special thanks to the government of India as well as to the ambassador of India in Czech Republic, mr. Shri Ashok Venkatesan and to the first secretary of the embassy, mr. Somnath Chatterjee, for the support of my studies about image processing at the Indian Institute of Remote Sensing in Dehradun via the ITEC scholarship. The knowledge gained within the studies was very helpful for the solving of the problem described in the paper.

References

1. Cruz-Martín, A., Fernández-Madrigal, J.A., Galindo, C., González-Jiménez, J., Stockmans-Daou, C., Blanco, J.L.: A LEGO Mindstorms NXT approach for teaching at data acquisition, control systems engineering and real-time systems undergraduate courses. Comput. Educ. **59**(3), 974–988 (2012)
2. Cuéllar, M.P., Pegalajar, M.C.: Design and implementation of intelligent systems with LEGO Mindstorms for undergraduate computer engineers. Comput. Appl. Eng. Educ. **22**(1), 153–166 (2014)
3. Diestel, R.: Graph Theory, Graduate texts in mathematics, vol. 173, 4th edn. Springer (2012)
4. Gonzalez, R., Woods, R.E., Eddins, S.: Digital Image Processing Using MATLAB. Prentice-Hall, Inc. (2003)
5. Gonzalez, R., Woods, R.: Digital Image Processing, 3rd edn. Pearson Prentice Hall (2009)
6. Inanc, T., Dinh, H.: A low-cost autonomous mobile robotics experiment: control, vision, sonar, and handy board. Comput. Appl. Eng. Educ. **20**(2), 203–213 (2012)
7. Inc., T.M.: Matlab 2013b, Image Processing Toolbox. 3 Apple Hill Drive, Natick (2013). http://www.mathworks.com/help/images/
8. Jara, C.A., Candelas, F.A., Puente, S.T., Torres, F.: Hands-on experiences of undergraduate students in automatics and robotics using a virtual and remote laboratory. Comput. Educ. **57**(4), 2451–2461 (2011)
9. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**(1), 90–98 (1986)
10. Kulich, M., Chudoba, J., Košnar, K., Krajník, T., Faigl, J., Přeučil, L.: SyRoTek - distance teaching of mobile robotics. IEEE Trans. Educ. **56**(1), 18–23 (2013)
11. Siegwart, R., Nourbakhsh, I.: Introduction to Autonomous Mobile Robots. Bradford Book (2004)
12. Škrabánek, P.: Time efficient graph version of the A* algorithm optimized for matlab. In: Proceedings of Mendel 2014: 20th International Conference on Soft Computing, pp. 299–304 (2014)

Neural Network Approach to Image Steganography Techniques

Robert Jarušek, Eva Volna and Martin Kotyrba

Abstract Steganography is one of the methods used for the hidden exchange of information and it can be defined as the study of invisible communication that usually deals with the ways of hiding the existence of the communicated message. In this way, if successfully it is achieved, the message does not attract attention from eavesdroppers and attackers. Using steganography, information can be hidden in different embedding mediums, known as carriers. These carriers can be images, audio files, video files, and text files. The focus in this paper is on the use of an image file as a carrier. The proposed approach is based on backpropagation neural networks. The essential part of this article aims to verify the proposed approach in an experimental study. Further, contemporary method of application and results are presented in this paper as an example.

Keywords Steganography · Steganalysis · Blum blum shub generator · Discrete fourier transform (DFT) · Fast fourier transform (FFT) · Backpropagation neural networks

R. Jarušek · E. Volna (✉) · M. Kotyrba
Department of Informatics and Computers, University of Ostrava, 30 Dubna 22, 70103 Ostrava, Czech Republic
e-mail: eva.volna@osu.cz

R. Jarušek
e-mail: robert.jarusek@osu.cz

M. Kotyrba
e-mail: martin.kotyrba@osu.cz

1 Neural Network Classification Techniques Used on Steganography

Cryptography method is used to hide secret data by scrambling so that it is unreadable, but this method does not assure security and robustness as the hacker can easily guess that there is a confidential message passing on from the source to the destination [11]. Steganography is known as "covered writing" and the idea of steganography is hiding the existence of a message [2]. The cover carriers includes (images, audio, video, text, or other digitally representative code) which will hold the hidden information. A message is the information which is being hidden and it be can be a plaintext, cipher text, images, or anything that can be embedded into a bit stream. Cover image is called as carrier image and is the original image in which the secret data i.e., the payload has embedded. The image that is obtained after embedding the payload into the cover image is called as stego image. The skill of observing the invisible embedded messages in images, audio, video, text as multimedia and protocols called steganalysis. An efficient steganalysis method should determine the existence of implanted messages and stego digital image and present some results about the used steganographic algorithm [7]. Steganalysis can be categorized into two groups: (a) static and (b) dynamic. Guesstimate some parameter(s) of the embedded algorithm or the secret message is the target of dynamic steganalysis, identifies the existence/non-existence of a secret message is the aim of static steganalysis. Static steganalysis: discovering the existence/non-existence of a concealed message in a stego file and recognizing the stego embedded algorithm. Dynamic steganalysis: guesstimating the implanted message length, position(s) of the concealed message, the secret key used in implanting, some parameters of the stego implanting algorithm and take out the concealed message.

Neural Network classification (NN): One kind of classification techniques is called a neural network classifier. The important problem in a neural network is that convergence is not fast. Practically, this is the most important restriction of neural network applications, because data hiding method is not a linear method, if we only employ linear classification technique to categorize images. The neural network has an admirable facility to simulate any nonlinear correlation. Therefore, it has been used to categorize images. Neural network draws on three levels: input level, hidden level and output level. In [5], the authors presented actual features through means of quality analysis from clear images and stego images, after that, applying neural network approach as a separator to differentiate non-stego-images and stego-images. Liul and his colleagues utilized backpropagation neural network to approve their approach. The first phase is to learn and test neural network to acquire network parameters. In spite these parameters, they could simulate the outcomes. In neural network, they adjust a number of characteristics of the input layer. In [10], the authors presented a novel technique based on neural network to get numerical features of images to detect the essential concealed data. Shaohui and his colleagues utilize backpropagation neural network to simulate and train images. This technique discovers statistically indicates after original images has been concealed message, then utilizing the ability of estimation of neural network for demonstrating either an image is non-stego or stego image. A blind image steganalysis scheme is proposed, in which feature is consists of the numerical moments of characteristic functions of the test image, the prediction-error image and their wavelet sub bands. The approach of categorizer is another main factor in steganalysis. In [3], the authors suggested a blind steganalysis method which depends on a universal neural network (NN) approach and matches it to Stegdetect – a kind of tool that uses a linear classification device. More neural network classification techniques used on steganalysis is shown in [6].

2 Theoretical Background

In order to show the propose approach dealing with steganography based on neural networks, it is necessary to explain basic concepts, which are used in the following text.

2.1 Fourier Transforms for Image Processing

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image. The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction [12] and image compression [8].

The Discrete Fourier Transform (DFT) is the sampled Fourier Transform and therefore does not contain all frequencies forming an image, but only a set of samples which is large enough to fully describe the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image, i.e. the image in the spatial and Fourier domain are of the same size. For a square image of size $N \times N$, the two-dimensional DFT is given by (1):

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-i2\pi\left(\frac{k \cdot i}{N} + \frac{l \cdot j}{N}\right)} \quad (1)$$

where *f(a,b)* is the image in the spatial domain and the exponential term is the basis function corresponding to each point *F(k,l)* in the Fourier space. The equation can be interpreted as: the value of each point *F(k,l)* is obtained by multiplying the spatial image with the corresponding base function and summing the result. The basis functions are sine and cosine waves with increasing frequencies, *i.e. F(0,0)* represents the DC-component of the image which corresponds to the average brightness and *F(N − 1,N − 1)* represents the highest frequency.

In a similar way, the Fourier image can be re-transformed to the spatial domain. The inverse Fourier transform is given by (2):

$$f(a,b) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l) e^{-i2\pi\left(\frac{k \cdot a}{N} + \frac{l \cdot b}{N}\right)} \tag{2}$$

To obtain the result for the above equations, a double sum has to be calculated for each image point. However, because the Fourier Transform is *separable*, it can be written as (3):

$$F(k,l) = \frac{1}{N} \sum_{b=0}^{N-1} P(k,b) e^{-i2\pi\frac{l \cdot b}{N}} \text{ where } P(k,b) = \frac{1}{N} \sum_{a=0}^{N-1} f(a,b) e^{-i2\pi\frac{k \cdot a}{N}} \tag{3}$$

Using these two formulas, the spatial domain image is first transformed into an intermediate image using N one-dimensional Fourier Transforms. This intermediate image is then transformed into the final image, again using N one-dimensional Fourier Transforms. Expressing the two-dimensional Fourier Transform in terms of a series of 2N one-dimensional transforms decreases the number of required computations. Even with these computational savings, the ordinary one-dimensional DFT has N^2 complexity. This can be reduced to $N \log_2 N$, if we employ the Fast Fourier Transform (FFT) to compute the one-dimensional DFTs. This is a significant improvement, in particular for large images. There are various forms of the FFT and most of them restrict the size of the input image that may be transformed, often to $N = 2^n$ where n is an integer. The mathematical details are well described in [8].

The Fourier Transform produces a complex number valued output image which can be displayed with two images, either with the *real* and *imaginary* part or with *magnitude* and *phase*. In image processing, often only the magnitude of the Fourier Transform is displayed, as it contains most of the information of the geometric structure of the spatial domain image. However, if we want to re-transform the Fourier image into the correct spatial domain after some processing in the frequency domain, we must make sure to preserve both magnitude and phase of the Fourier image.

2.2 Blum Blum Shub Generator

Blum Blum Shub generator (BBS) is a pseudorandom number generator [1].

DEFINITION [$x^2 \bmod N$ GENERATOR]: Let $\boldsymbol{N}$ = {integers N | $N = P*Q$, such that P, Q are equal length ($|P| = |Q|$) distinct primes 3 mod 4} be the set of parameter values. For $N \in \boldsymbol{N}$, let $X_N = \left\{x^2 \bmod N \middle| x \in Z_N^*\right\}$ be the quadratic residues mod N. Let $X = \text{disjoint} \bigcup_{N \in N} X_N$ be the seed domain.

An interesting characteristic of this generator is that we can directly calculate any of the x values, i.e. (4):

$$x_i = \left(x_0^{(2^i(\text{mod}((P-1)*(Q-1))))}\right) (\text{mod} N) \tag{4}$$

This means that in applications where many keys are generated in this fashion, it is not necessary to save them all. Each key can be effectively indexed and recovered from that small index and the initial x and N.

2.3 Backpropagation Neural Network

Backpropagation algorithm belongs to a group called "gradient descent methods". An intuitive definition is that such an algorithm searches for the global minimum of the weight landscape by descending downhill in the most precipitous direction. The initial position is set at random selecting the weights of the network from some range (typically from −1 to 1 or from 0 to 1). Considering the different points, it is clear, that backpropagation using a fully connected neural network is not a deterministic algorithm. The basic backpropagation algorithm can be summed up in the following equation (the *delta rule*) for the change to the weight w_{ji} from node i to node j (5):

$$\begin{array}{ccccccc} \text{weight} & & \text{learning} & & \text{local} & & \text{input signal} \\ \text{change} & & \text{rate} & & \text{gradient} & & \text{to node } j \\ \Delta w_{ji} & = & \eta & \times & \delta_j & \times & y_j \end{array} \tag{5}$$

where the local gradient δ_j is defined as follows [9]:

1. If node j is an output node, then δ_j is the product of $\varphi'(v_j)$ and the error signal e_j, where $\varphi(\_)$ is the logistic function and v_j is the total input to node j (i.e. $\Sigma_i\, w_{ji}y_i$), and e_j is the error signal for node j (i.e. the difference between the desired output and the actual output);
2. If node j is a hidden node, then δ_j is the product of $\varphi'(v_j)$ and the weighted sum of the δ's computed for the nodes in the next hidden or output layer that are connected to node j.

The actual formula is $\delta_j = \varphi'(v_j)$ & Sigma$_k$ $\delta_k w_{kj}$ where k ranges over those nodes for which w_{kj} is non-zero (i.e. nodes k that actually have connections from node j. The δ_k values have already been computed as they are in the output layer (or a layer closer to the output layer than node j).

3 The Proposed Steganographic System

The proposed steganographic system is shown in Fig. 1. It works in the following steps:

1. Let us have an image whose size corresponds to the squares of 2 in pixels (i.e. $2^n \times 2^n$, where n is an integer greater than zero). The size of this is important so that we could use 2D Fast Fourier transform (FFT) without further necessary adjustments.
2. If we have such an image, we convert it into grey scale, i.e. we extract the brightness components. We receive the corresponding values of the range 0 to 255 and transform them to <0,1> scale.
3. Then we analyse the image with Fast Fourier Transform (FFT). On the FFT image, the low frequency area is in the center of the image and the high frequency areas are at the corners of the image. The result of a Fourier transform is complex - it has both real and imaginary parts. We calculate the magnitude of the complex number $z = a + bi$, which equals a complex number's absolute value $|z| = \sqrt{a^2 + b^2}$.
4. We can see such an image as a result of applying FFT (Fig. 2), where the second quadrant is centrally symmetric with the fourth quadrant and the first quadrant is centrally symmetric with the third quadrant. Therefore, we will process data only from the first and the second quadrant.

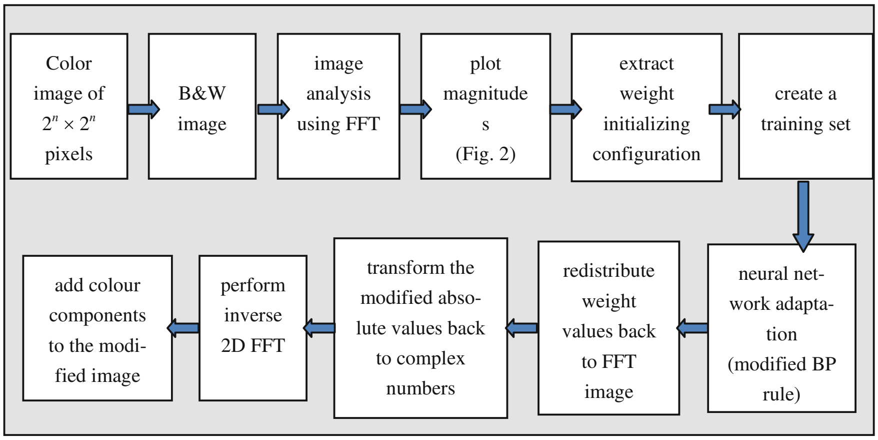

Color image of $2^n \times 2^n$ pixels
B&W image
image analysis using FFT
plot magnitude s (Fig. 2)
extract weight initializing configuration
create a training set
neural net-work adapta-tion (modified BP rule)
redistribute weight values back to FFT image
transform the modified abso-lute values back to complex numbers
perform inverse 2D FFT
add colour components to the modi-fied image

Fig. 1 The proposed steganographic system

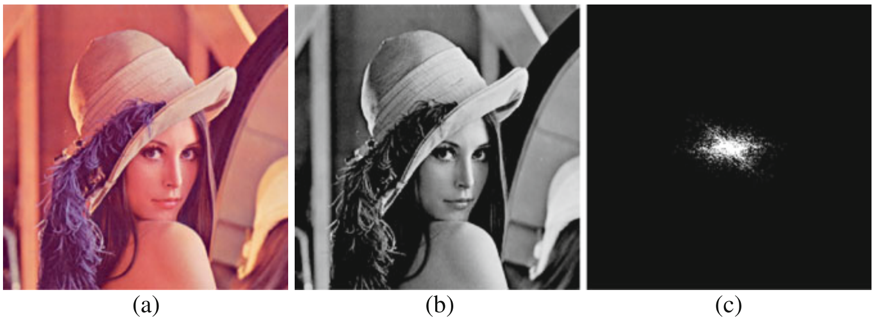


Fig. 2 (a) Colour image. (b) B&W image. (c) FFT application to image (Colour figure online)

5. We extract a weight initialising configuration of (unlearned) neural network using *the proposed algorithm* from the resulting data.
6. We adapt the resulting neural network according to the training set using a modified backpropagation rule.
7. After the adaptation, weight values are redistributed back into the FFT image using *the proposed algorithm*.
8. Modified absolute values are transformed back into the real and imaginary components of the complex numbers.
9. Then we perform the inverse 2D FFT.
10. We add color components into a modified image.

3.1 The Training Set

The proposed steganographic system based on neural networks has used the training set, which is created in the following way [4].

- BBS generator has worked with the following parameters, see Eq. (4): $P = 1017$, $Q = 1907$, and $N = 1939419$ (i.e. $N = P * Q$).
- The seed $x_0 = 123$.
- The “embedded message” was “SECRET”.

Based on the key which is represented by the chain corresponding with the seed x_0, series of random numbers x_i is generated according to the formula (4). We have used the random generator BBS with specific initialization of variables (6).

$$x_i = \left(123^{(2^i(\mathrm{mod}((1017-1)*(1907-1))))}\right)(\mathrm{mod}(1017*1097)) \tag{6}$$

Next, six consecutive values (e.g. six input vectors) were normalized to the interval $\langle 0, 1 \rangle$. A series of these values was used as input vectors of a given training set representing single character of a given embedded message "SECRET". The corresponding output vectors are created by the 7-bit ASCII representation of each character. Therefore the training set of each neural network includes only one pattern, e.g. each neural network is learned only on one character of an embedded message. In other words, we have used so many neural networks how many characters are included in the message. In our experimental study, we used six backpropagatin neural networks with topology 1–4–7 (e.g. one input unit, 4 hidden units and 7 output units). Each single line in Table 1 represents one training pattern for a given neural network.

Table 1 Training patterns

Training patterns	Input $x_0 = 123$	Embedded message	OUTPUT ASCII
1	0.007801	S	(1,0,1,0,0,1,1)
2	0.018149	E	(1,0,0,0,1,0,1)
3	0.835446	C	(1,0,0,0,0,1,1)
4	0.902752	R	(1,0,1,0,0,1,0)
5	0.301604	E	(1,0,0,0,1,0,1)
6	0.481282	T	(1,0,1,0,1,0,0)

3.2 The Modified Backpropagation Rule

The backpropagation network has worked with the following parameters: activation function was binary sigmoid with slope parameter $\lambda = 1$, learning rate $\alpha = 0.05$–0.1, a parameter "momentum" was set at 0.

The modified backpropagation rule works in the following steps. The weight initializing configuration of each used neural network is extracted from FFT so that the weight value is extracted from each row of a 2D matrix according to the formula (7).

$$w_i = \sum_{j=0}^{2^n} z_j, \tag{7}$$

where n is an integer greater than zero corresponding to the size of an image z_j is the magnitude of the j complex number. These pixels are chosen sequentially (other approaches will be a subject of our further research). In this way, we receive weight initializing configuration of all neural networks. It means that we extract 43 weight values (e.g. 4 weight values between the input and the hidden layer, 28 weight values between the hidden and the output layer, and 11 biases associated with hidden and output units) for each neural network, because the used topology is 1–4–7. Since we need 6 neural networks (for each character of the embedded message one network), the whole number of initial weights is 258 values.

We adapt all neural networks with the received weight configurations according to the training set using a modified backpropagation rule. The own adaptation of each neural network runs by backpropagation rule, which was modified in following way. The proposed algorithm works in two phases. The control phase follows after each adaptation step. In the course of the adaptation, there is monitored whether output neuron values corresponding to desired outputs. If the actual value of the output neuron is larger (smaller) than the threshold b, then the value of the corresponding output neuron holds 1 (0). In our case, the threshold is set to $b = 0.5$. The partial adaptation is performed so long as the term is satisfied for all neurons in the output layer. After that the adaptation is completed. We are compelled to deal with an optimization problem, where it is both necessary to adapt the whole training set and to change weights minimally. Here, weight values represent the absolute values of complex numbers in 2D FFT. Such a obtained configuration of neural network is decomposed back into the FFT image according to the formula (8).

$$z_{ji}(new) = z_{ji}(old)\frac{w_i(new)}{w_i(old)}, \tag{8}$$

where image z_{ji} is the magnitude of the j complex number ($j = 1, \ldots, 2^n$), $w_i(new)$ is the *i-ht* weight value after adaptation and $w_i(old)$ is the initial *i-ht* weight value. In our experimental study $i = 1, \ldots, 258$.

3.3 Experimental Outcomes

Figures 3 and 4 show the same picture with and without an embedded message and their histograms. How you can see in both figures, there are minimal differences in their histograms. Figure 5 shows the difference between Figs. 3(a) and 4(a).

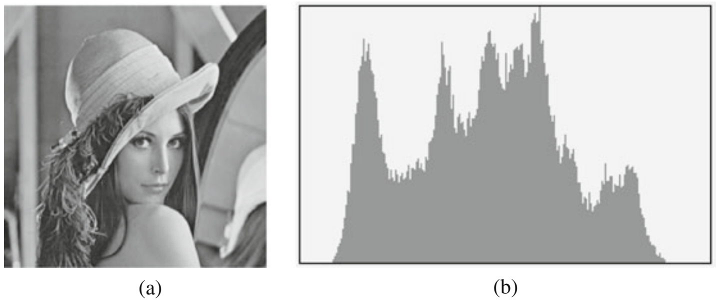


Fig. 3 The picture without an embedded message (a), a histogram (b)

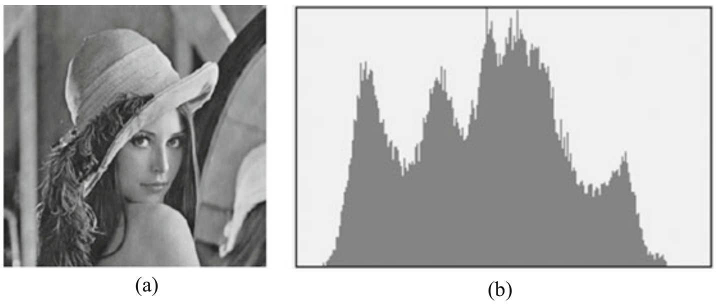


Fig. 4 The picture with an embedded message (a), a histogram (b)

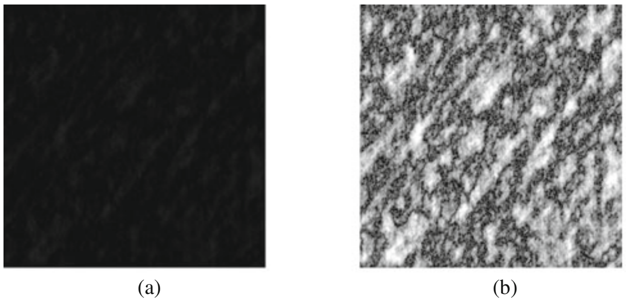


Fig. 5 The difference (a), a normalized difference (b)

4 Conclusions

The focus in this paper is on the use of an image file as a carrier. The proposed approach is based on backpropagation neural networks. The essential part of this article aims to verify the proposed approach in an experimental study. In contrast with standard steganography methods, which try to hide information into the image regardless of the data representation, the utilization of artificial neural networks for hiding information is promising especially because the neural network intentionally uses features of the image (a cover medium) for its adaptation. The stego images that are produced by using such data hiding techniques are inherently robust against main geometrical attacks.

Our further work will include the proposal of another way of selecting pixels in images, which will be applicable for the representation of weight values assigned to the neural network. These pixels can be generated as the continuing use of Blum Blum Shub generators operating according to a known secret key or for example to engage evolutionary techniques for their distribution. Another approach for our future work could be some proposal of a solution, when an adaptation of neural networks deals with an optimization problem, where it is both necessary to adapt the whole training set and to change weights minimally.

Acknowledgments The research described here has been financially supported by University of Ostrava grant SGS17/PřF/2015. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the sponsors.

References

1. Blum, L., Blum, M., Shub, M.A.: Simple unpredictable pseudo-random number generator. SIAM J. Comput. **15**(2), 364–383 (1986)
2. Dasgupta, K., Mandal, J.K., Dutta, P.: Hash based least significant bit technique for video steganography (HLSB). Int. J. Secur. Priv. Trust Manage. **2**(2) (2012)
3. Holoska, J., Oplatkova, Z., Zelinka, I., Senkerik, R.: Comparison between neural network steganalysis and linear classification method stegdetect. In: 2010 Second International Conference on Computational Intelligence, Modeling and Simulation (CIMSiM), vol. 10, pp. 15–20 (2010)
4. Jarušek, R., Volná, E., Kotyrba, M.: Steganography based on neural networks (a preliminary study). In: Proceedings of the 20th International Conference on Soft Computing, Mendel 2014, Brno, Czech Republic, pp. 223–228 (2014)
5. Liu, S., Yao, H., Gao, W.: Steganalysis based on wavelet texture analysis and neural network. In: Fifth World Congress on Intelligent Control and Automation, 2004. WCICA 2004, vol. 5, pp. 4066–4069 (2004)
6. Mohammadi, F.G., Abadeh, M.S.: A survey of data mining techniques for steganalysis. Recent Advances in Steganography, pp. 1–25 (2012)
7. Nissar, A., Mir, A.H.: Classification of steganalysis techniques study. Digit. Signal Process. **20** (6), 1758–1770 (2010)
8. Schatzman, J.C.: Accuracy of the discrete Fourier transform and the fast Fourier transform. SIAM J. Sci. Comput. **17**, 1150–1166 (1996). doi:10.1137/s1064827593247023
9. Seung, S.: Multilayer perceptrons and backpropagation learning. 9.641 Lecture 4. 1–6. http://hebb.mit.edu/courses/9.641/2002/lectures/lecture04.pdf (2002)
10. Shaohui, L., Hongxun, Y., Wen, G.: Neural network based steganalysis in still images. In: Proceedings of the 2003 International Conference on Multimedia and Expo, ICME'03, vol. 20, pp. 509–512 (2003)
11. Vani, G.B., Prasad, E.V.: Scalable and highly secured image steganography based on Hopfield chaotic neural network and wavelet transforms. Int. J. Comput. Sci. Issues **10**(3), 1 (2013)
12. Starha, P., Martisek, D., Matousek, R.: Numerical methods of object reconstruction using the method of moments. In: Proceedings of 20th International Conference on Soft Computing—Mendel 2014. Mendel series vol. 2014, pp. 241–248, Brno, ISSN: 1803-3814 (2014)

Rough-Fuzzy Collaborative Multi-level Image Thresholding: A Differential Evolution Approach

Sujoy Paul, Shounak Datta and Swagatam Das

Abstract In this article, a granular computing based multi-level gray image thresholding algorithm is presented. An image is divided into spatial blocks called granules, and the classes of gray levels are represented using a fuzzy-rough collaborative approach, where the measure of roughness of a rough set is also modified from the classical definition of rough sets. This measure for each rough set is minimized simultaneously to obtain the optimal thresholds. Tchebycheff decomposition approach is employed to transform this multi-objective optimization problem to a single objective optimization problem. Differential Evolution (DE), one of the most efficient evolutionary optimizers of current interest, is used to optimize this single objective function, thus reducing the execution time. Superiority of the proposed method is presented by comparing it with some popular image thresholding techniques. MSSIM index and Probabilistic Rand Index (PRI) are used for quantitative comparison on the Berkley Image Segmentation Data Set (BSDS300).

Keywords Rough sets · Fuzzy sets · Multi-level image thresholding · Fuzzy image thresholding · Differential evolution · Tchebycheff approach

1 Introduction

The goal of segmentation is to simplify or change the representation of an image into something that is more meaningful and easier to analyze for several computer vision and pattern recognition applications. Over past years popular segmentation approaches included edge-based methods [13], region-based methods [6], local [4],

S. Paul
Department of Electronics and Telecommunication Engineering,
Jadavpur University, Kolkata 700032, India

S. Datta · S. Das (✉)
Electronics and Communication Sciences Unit, Indian Statistical Institute,
Kolkata 700108, India
e-mail: swagatamdas19@yahoo.co.in

global threshold techniques, and connectivity-preserving relaxation methods [10]. Among global thresholding techniques, entropy based methods like Shannon entropy [3, 25], Renyi entropy [22], Tsalli entropy [23].

Otsu [18] developed a non-parametric multi-level image segmentation algorithm, in which the intra class variance of gray levels was minimized to obtain the thresholds. Tizhoosh et al. [29] proposed a fuzzy type II based image segmentation method, in which they obtained the optimal thresholds by maximization of a measure named ultra-fuzziness, associated with type II fuzzy sets. Global multi-level thresholding by unsupervised clustering techniques like K-means or Fuzzy C-means (FCM) [34] also needs mention. Selection of threshold based on restricted equivalence function and maximization of measures of similarity has been proposed in [5]. Some recently proposed image thresholding techniques, which need mention are [2, 33].

Rough set theory [20] has become a popular mathematical framework for granular computing [1]. The concepts of rough sets coupled with granulation of an image have been used for image thresholding by [19]. In their work, crisp values are used for rough set approximations and the rough entropy of the rough sets is maximized to obtain the optimal thresholds. However, their algorithm lacked an automated process of choosing the granule size and they provided segmentation results only on a few images. Also, the algorithm was not generalized for multi-level thresholding.

In this paper, a global thresholding technique is proposed, based on rough sets. Instead of crisp representation, fuzzy representation is used for lower approximations of a rough set. Thereafter, the fuzzy roughness measures formulated for these sets are simultaneously minimized for optimal selection of thresholds, which is a Multi-objective Optimization Problem (MOP). Unlike [19], we have avoided maximization of rough entropy to solve this problem due to its drawback, which is explained in Sect. 4.3. Instead, we have used Tchebycheff approach [16, 17] to convert this MOP to a Single-objective Optimization Problem (SOP), which is solved using a global optimizer, Differential Evolution (DE) [7, 26]. In our algorithm, by incorporation of fuzziness in the representation of lower approximation of the rough sets, the granule size may be kept constant as discussed in Sect. 4.1. By using DE as an optimizer, the proposed method becomes computationally efficient and this is illustrated through comparisons with two other global optimization methods in Sect. 4.4.

2 Image as Rough Sets with Fuzzy Membership Values: Proposed Method

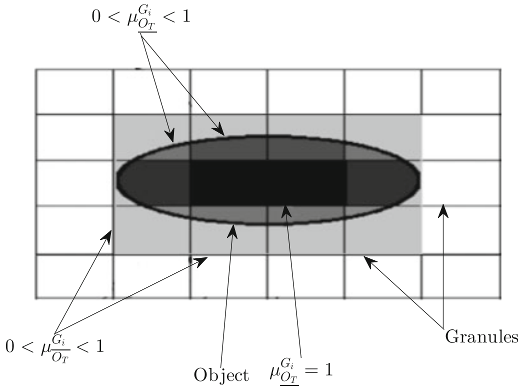
A Rough set [20, 21] provides the formal approximation of a crisp set in terms of a pair of sets called the lower and upper approximations of the original set also known as the B-lower and B-upper approximations and represented as a tuple $<\underline{B}X, \overline{B}X>$. Extensive description and survey on rough sets as well as fuzzy-rough sets may be found in [11, 14, 30].

In simple terms, in crisp representation of a rough set, if the features of an element are a subset of the properties of the object (or target set) which is to be represented, then it is included in the lower approximation set, else excluded. In such a crisp scheme, there is a high probability of potentially feasible elements to be discarded from the lower approximation set. To overcome this problem, a fuzzy membership value (between 0 and 1) may be used to denote the degree by which this element belongs to the lower approximation. On the other hand, the upper approximation set contains all those elements, whose fuzzy membership values are above zero, i.e. even if a single property of the object which is to be approximated matches with a property of the element. Figure 1 illustrates this concept using nomenclatures as discussed next. This concept is applied to extract objects from the background in an image.

Consider an image I consisting of a collection of pixels having levels in $[0, L-1]$. Let I be partitioned into non-overlapping windows of size $m \times n$. Each window can be considered as a granule. Information within these granules is the basic element that approximates the object or background of an image. These granules help to represent the image as rough sets. Let G_i be the set of unique gray levels of pixels present in the i^{th} granule. If T be the threshold that divides the gray scale into two sets of object (O_T) and background (B_T), then,

$$O_T = \{0, 1, 2, T-1, T\}, B_T = \{T+1, T+2, ., L-2, L-1\}$$

where L is the number of levels present in the image. These two sets form the target set that should be approximated using granules to form the lower and upper approximation sets.



$0 < \mu^{G_i}_{\underline{O_T}} < 1$
$0 < \mu^{G_i}_{\underline{O_T}} < 1$
Object
$\mu^{G_i}_{\underline{O_T}} = 1$
Granules

Fig. 1 Rough set representation of an object having fuzzy membership values

The lower and upper approximation sets may be defined as follows:
Lower approximation for object: $\mu_{\underline{O_T}} = \{\mu_i(d) : d = |O_T \cap G_i|\}$
Lower approximation for background: $\mu_{\underline{B_T}} = \{\mu_i(d) : d = |B_T \cap G_i|\}$

Upper approximation for object: $\mu_{\overline{O_T}} = \{x_i : x_i = 1, if\, O_T \cap G_i \neq \emptyset, x_i = 0, otherwise\}$
Upper approximation for background: $\mu_{\overline{B_T}} = \{x_i : x_i = 1, if\, B_T \cap G_i \neq \emptyset, x_i = 0, otherwise\}$
where $\mu_i(d)$ denote a linear fuzzy membership function as shown in Fig. 2 and i=1 to N_G, N_G being the number of granules. Let us denote the fuzzy-roughness of the object by R_{O_T} and that of the background by R_{B_T}. They may be defined as:

$$R_{O_T} = 1 - \frac{\sum \mu_{\underline{O_T}}}{\sum \mu_{\overline{O_T}}} \quad \text{and} \quad R_{B_T} = 1 - \frac{\sum \mu_{\underline{B_T}}}{\sum \mu_{\overline{B_T}}} \tag{1}$$

It should be noted that the threshold (T) may be s.t. $\sum \mu_{\underline{O_T}} = \sum \mu_{\overline{O_T}} = 0$, leading to a $0/0$ form of (1). In such cases, R_{O_T} is considered to be 1. This is applicable for R_{B_T} as well. Now, to obtain a good approximation of the image as object and background, the fuzzy membership values of the granules present in the lower approximation, should approach unity. Thus, the roughness for each rough set should be minimized simultaneously by varying the thresholds. Hence, in other words, the optimal threshold T^* may be expressed as,

$$T^* = \min_T [R_{O_T}, R_{B_T}] \tag{2}$$

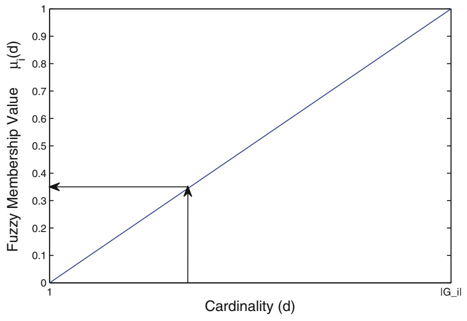


Fig. 2 Linear fuzzy membership function for lower approximation

The idea for bi-level thresholding presented can be extended to multi-level thresholding. For n-thresholds $T = \{T_1, T_2 \ldots T_n\}$, the gray scale will be divided into n + 1 regions, given by the set $S = \{S_1, S_2 \ldots S_{n+1}\}$, where $S_1 = \{0, 1, 2 \ldots T_1\}$, $S_2 = \{T_1 + 1, T_1 + 2, , T_2\} \ldots S_{n+1} = \{T_n + 1, T_n + 2, , L - 1\}$. These sets form the different target sets to be approximated by using the granules of the image to constitute the lower and upper approximation sets. Let us define the fuzzy membership values to the granules for lower approximations of set S_k as:

$$\mu_{\underline{S_k}} = \{\mu_i(d) : d = |S_k \cap G_i|\}$$

Membership values assigned to the granules for upper approximation for set S_k are defined as:

$$\mu_{\overline{S_k}} = \{x_i : x_i = 1, if\ S_k \cap G_i \neq \emptyset, x_i = 0, otherwise\}$$

where $k \in [1, n+1]$, $\mu_i(d)$ is a fuzzy membership function as shown in Fig. 2, i = 1 to N_G, N_G denoting the number of granules. Let us define the fuzzy-roughness for the set S as:

$$R_{S_k} = 1 - \frac{\sum \mu_{\underline{S_k}}}{\sum \mu_{\overline{S_k}}} \tag{3}$$

(3) where $k \in [1, n+1]$. Now in order to get a good approximation of the image as rough sets, the roughnesses (R_{S_k}) should be minimized simultaneously. This may be achieved by varying the threshold vector T. Hence, the optimal threshold vector T^* may be expressed as,

$$T^* = \min_T [R_{S_1}, R_{S_2} \ldots R_{S_{n+1}}] \tag{4}$$

It may be noted that the granules, which approximate the target sets, helps involving the information about the spatial variations of gray levels of pixels in an image. Equation 4 is a Multi-objective Optimization Problem (MOP). It is converted to a single objective by Tchebycheff Approach as discussed next.

3 Optimization

3.1 Tchebycheff Approach

Tchebycheff introduced a decomposition method [16, 17] that converts the problem of approximating the Pareto front of an MOP into a set of SOPs formed by applying varying weights to the objective functions. In the present context, equal weights are applied to all the roughness. The following eqn., known as the augmented weighted Tchebycheff approach [16], converts the MOP to a SOP.

$$g(T) = \max\{W_k * |R_{S_k}(T) - z_k^*|\} + \rho \sum_{i=1}^{m} (R_{S_i}(T) - z_i^*)$$
$$T^* = \min_T g(T) \tag{5}$$

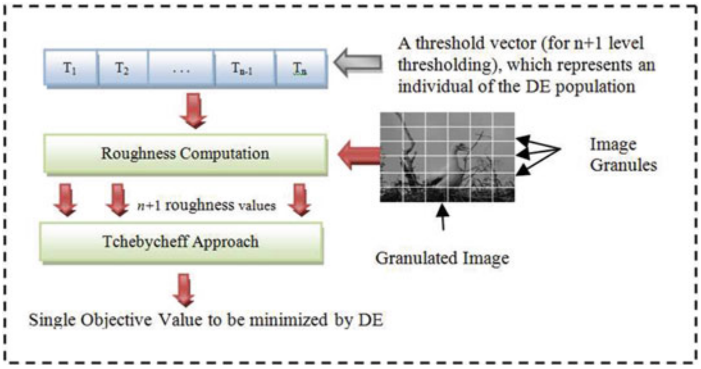
where $k \in 1, 2, m$, m being the number of objectives. $z^* = [z_1^*, z_2^* \ldots, z_m^*]$ is the ideal point s.t. $z_k^* = min\{R_{S_k}(T)\}$ for all possible decision values and W_k is the weight value which is considered to be $W_k = 1/m$ in the present context. $g(T)$ is required to be minimized in order to get an optimal set of thresholds. ρ is generally considered to have a low value such that it do not overshadow the first term of the equation [27]. $\rho = 0.01$ have been found to produce good appropriate results in the current context, and this value have been used for all results presented in this article.

3.2 *Differential Evolution*

DE [7, 26] is a population-based evolutionary algorithm for continuous parameter spaces. DE starts with a set of candidate solutions sampled from a uniform probability distribution over the feasible search volume. These candidate solutions then undergoes Mutation and Crossover on an iterative basis, which evolves them to attain optima. The iterative process comes to an end on reaching the termination criterion, which in this article has been considered to be the number of objective function ($g(T)$) evaluation. More details regarding DE may be found at [7]. Figure 3 depicts the computation process of the objective value which is optimized by DE.

4 Experimental Results

The images used in this article are from Berkeley Segmentation Dataset [36] (apart from a few). The proposed algorithm is compared with the following:



T1
T2
...
Tn-1
Tn
A threshold vector (for n+1 level thresholding), which represents an individual of the DE population
Roughness Computation
Image Granules
n+1 roughness values
Tchebycheff Approach
Granulated Image
Single Objective Value to be minimized by DE

Fig. 3 A flowchart of the objective value computation procedure

– Algorithm 1 (for bi-level) and 4 (multi-level) of [5] (will be referred as A14)
– Otsu's method ([18], [35])
– Algorithm 3 (for bi-level) and 5 (multi-level) of [5] (will be referred as A35)
– Shannon Entropy (SE) ([3], [28]).

4.1 Choice of Granule Size

Choice of granule size is an important factor in the proposed algorithm. In case of very large granule, the variation of the pixel gray levels within these granules may be such that, a high percentage of the granule gray levels may lie in the boundary regions between the two threshold levels. So the purpose of granulation and representation of an image, comprising of granules having pixels of almost same gray values would fail leading to unsuccessful thresholding. Segmentation of the images with varying granule size led to an observation that the segmented images produced best results and remained almost insensitive to the change in granule size roughly from 5×5 to less than 15×15.

4.2 Visual and Quantitative Comparisons

Visual comparison for bi-level thresholding is presented in Figs. 4, 5 and 6. It may be observed that the proposed algorithm detects the letter T of Fig. 4a correctly, which is almost a failure for the other methods. In Fig. 5b, the mushroom is well segmented from the background, which is not achieved by the other algorithms. Similar explanations may be given for other images. Quantitative comparison is performed by using the Mean Structural Similarity (MSSIM) index [32]. Comparison is done for 300 images of the BSDS300 database and the success percentage plots for 1, 2, and 3 thresholds is in Fig. 7, which indicate the proposed Rough-Fuzzy scheme to produce good results. A “success” of a segmented image of a particular original image for a particular level of thresholding means that the MSSIM value for that segmentation is best than the other algorithms.

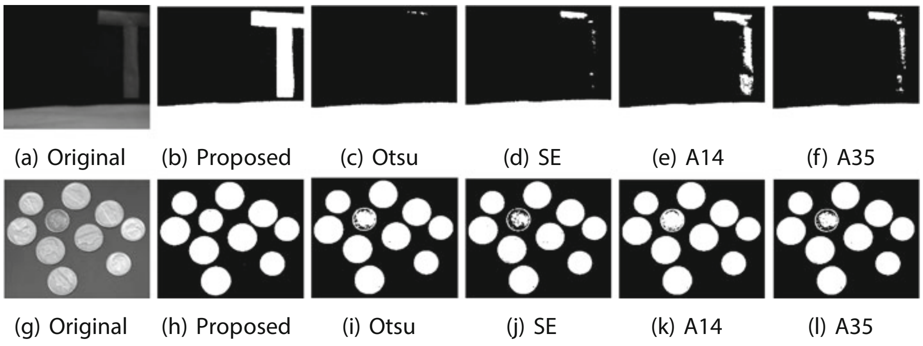


Fig. 4 Visual Comparison of thresholded images for bi-level thresholding

Probabilistic Rand Index (PRI) [31] is used to quantitatively compare the algorithms with respect to the images segmented by human subjects. It counts the fraction

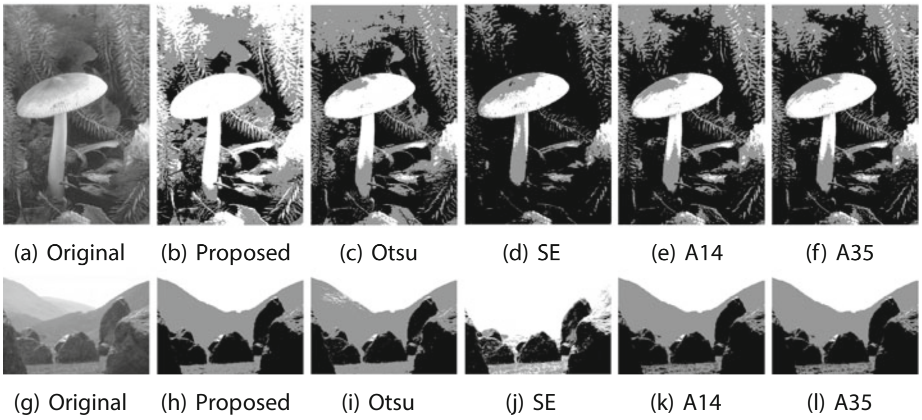


Fig. 5 Visual Comparison of thresholded images for tri-level thresholding

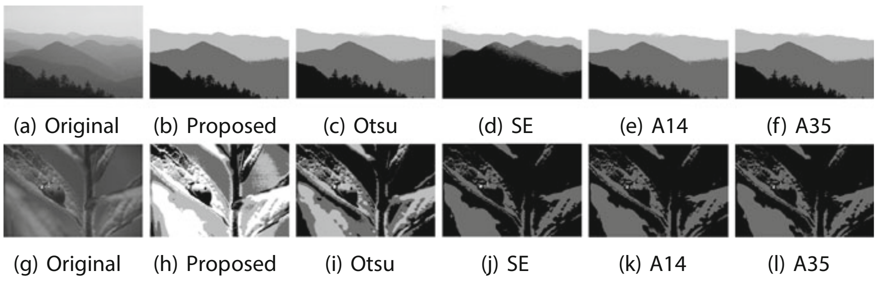


Fig. 6 Visual Comparison of thresholded images for 4-level thresholding

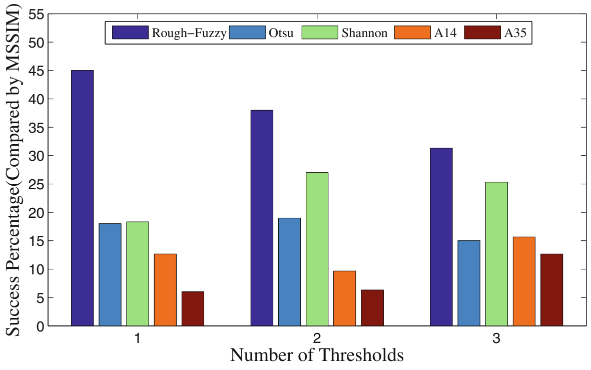


Fig. 7 Comparison using MSSIM values of 300 images in the dataset (in %)

of pairs of pixels whose labels are consistent between the human segmentation and the computed segmentation. An average over the five PRI values corresponding to 5 human segmented images for each test image is taken for comparison. Figure 11 presents an example of human segmentations along with images segmented by other algorithms. The success percentage with respect to the PRI measure over 300 images of BSDS300 is shown in Fig. 8.

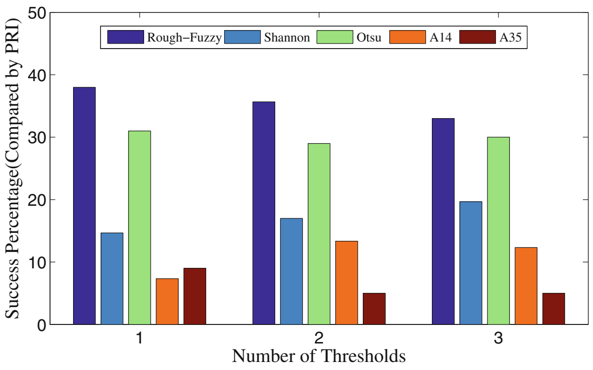


Fig. 8 Comparison of PRI values using 300 images of the dataset (in %)

4.3 Advantage of Tchebycheff Approach

Pal et al. [19] defined a measure called rough entropy (for bi-level), which if maximized, minimizes the roughness, in the following way:

$$RE_T = -\frac{e}{2}[R_{O_T} ln(R_{O_T}) + R_{E_T} ln(R_{E_T})] \tag{6}$$

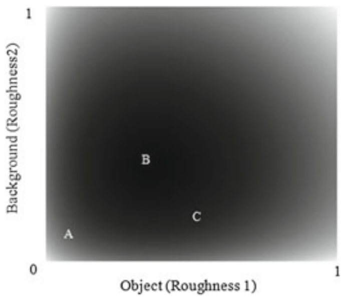


Fig. 9 Rough entropy using gray levels (Darker means Higher Entropy)

However, the process of optimizing the rough entropy may fail in some cases. For example, the ideal requirement is to have $R_{O_T} = R_{B_T} = 0$, but RE_T reaches a maximum of 1 when $R_{O_T} = R_{B_T} = 1/e$. This is illustrated in Fig. 9 where point A has lesser entropy and roughness measures than point B and C. The weighted

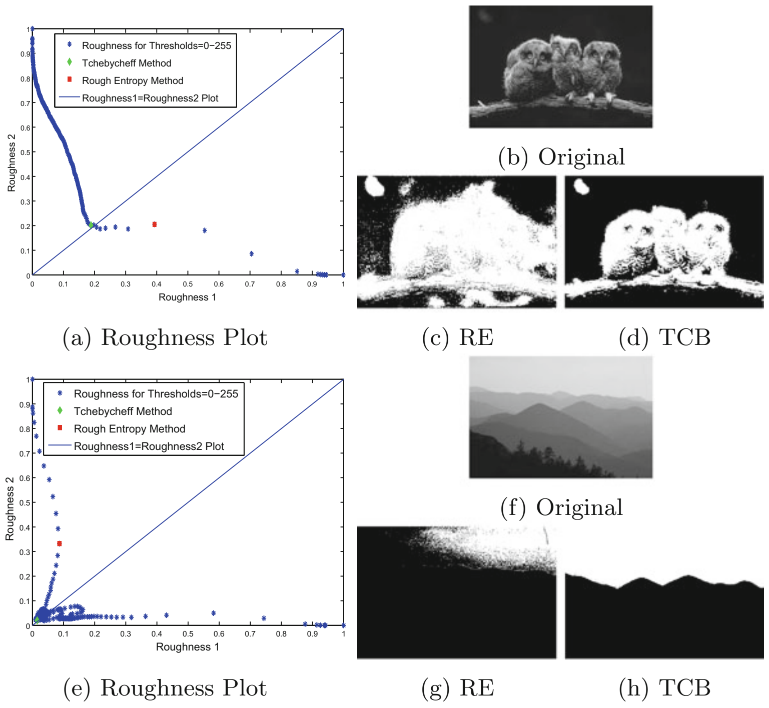


Fig. 10 Comparison of Rough Entropy (RE) and Tchebycheff (TCB) approach (i.e. the proposed method) for bi-level thresholding

Tchebycheff method overcomes this problem. Some examples for illustration of this fact are given in Fig. 10. In Fig. 10e it can be seen that optimal point determined by Rough-Entropy maximization lies far away from the origin than the point optimized by modified Tchebycheff method. In Fig. 10a, it can be seen that the Tchebycheff method minimizes the object background to a greater extent than the Rough-Entropy method.

4.4 Advantage of Using Differential Evolution

DE is a powerful evolutionary optimizer. Recently a convergence proof of this algorithm [12] has been reported under minor regularity assumptions. As the dimension of the search space (i.e. number of thresholds) increases linearly, the number of objective function evaluations (and hence the execution time) taken increases almost exponentially for exhaustive search, but almost linearly in case of DE. Some other well-known global optimization techniques like Particle Swarm Optimization (PSO) [15] and a real coded Genetic Algorithm (GA) [8] may also be used to optimize the

Table 1 Average computational time (in second)

Number of thresholds	DE	PSO	GA	Exhaustive search
1	0.1727	0.1880	0.4864	**0.0688**
2	**0.3055**	0.3992	1.1870	0.827
3	**0.5018**	0.6445	2.1472	3.0435
4	**0.7535**	0.9530	3.4752	647.64

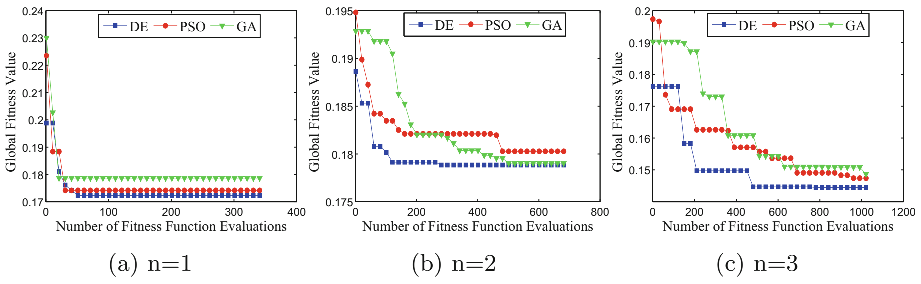


Fig. 11 Examples of convergence plots for 1,2,3 thresholds

proposed objective function of Eq. (5), under homogeneous experimental conditions. Table 1 presents a comparison w.r.t. computational time (averaged over 300 images of [36]) with other optimizers. Examples of the convergence plots of these algorithms are provided in Fig. 11 for a single image. It may be observed that DE not only converges faster, but also attains a better optimal value.

5 Conclusion

On the basis of image granules, a rough-fuzzy scheme is introduced for image thresholding, which uses fuzzy membership values for lower approximation of the rough sets. Tchebycheff approach is used to transform the problem of simultaneous roughness minimization from a multi-objective to a single objective problem. The advantage of this method over rough entropy is shown by mathematical reasoning and also with some examples. Along with bi-level segmentation, we have extended the concept to multi-level image thresholding. The superiority of this algorithm is shown with respect to some popular image segmentation algorithms. Usage of Differential Evolution, for solving the single objective problem, keeps the computational time low to a considerable extent.

References

1. Bargiela, A., and Pedrycz, W.: Granular Computing - An Introduction. Kluwer Academic Publishers (2003)
2. Beauchemin, M.: Image thresholding based on semivariance. Pattern Recogn. Lett. **34**(5), 456–462 (2013)
3. Benzid, R., Arar, D., Bentoumi, M.: A fast technique for gray level image thresholding and quantization based on the entropy maximization. In: 5th International Multi-Conference on Systems, Signals and Devices, pp. 1–4 (2008)
4. Bourjandi, M.: Image segmentation using thresholding by local fuzzy entropy-based competitive fuzzy edge detection. In: 2nd International Conference on Computer and Electrical Engineering, vol 2, pp. 298–301 (2009)
5. Bustince, H., Barrenechea, E., Pagola, M.: Image thresholding using restricted equivalence functions and maximizing the measures of similarity. Fuzzy Sets Syst. Elsevier **158**, 496–516 (2007)
6. Chen, G., Hu, T., Guo, X., Meng, X.: A fast region-based image segmentation based on least square method. IEEE Intl. Conf. Syst. Man Cybern. 972–977 (2009)
7. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)
8. Deb, K., Anand, A., Joshi, D.: A computationally efficient evolutionary algorithm for real-parameter optimization. Evo. Comp. **10**(4), 371–395 (2002)
9. Dubois, D., Prade, H.: Putting rough sets and fuzzy sets together. Intell. Decis. Support Theor. Decis. Libr. **11**, 203–232 (1992)
10. Eriksson, A., Barr, O., Astrom, K.: Image segmentation using minimal graph cuts. Intl. J. Eng. Res. Technol. (IJERT) 1(6) (2012)
11. Feng, F.: Generalized rough fuzzy sets based on soft sets. In:. International Workshop on Intelligence Systems and Applications, pp. 1–4 (2009)
12. Ghosh, S., Das, S., Vasilakos, A.V., Suresh, K.: On convergence of differential evolution over a class of continuous functions with unique global optimum. IEEE Trans. SMC-B **42**(1), 107–124 (2012)
13. Hsiao, Y.T., Chuang, C.L., Jiang, J.A., Chien, C.C.: A contour based image segmentation algorithm using morphological edge detection. IEEE Int. Conf. Syst. Man Cybern. **3**, 2962–2967 (2005)
14. Hu, Q., Zhang, L., An, S., Zhang, D., Yu, D.: On robust fuzzy rough set models. IEEE Trans. Fuzzy Syst. **20**(4), 636–651 (2012)
15. Kennedy, J., Eberhat, R.: Particle swarm optimization. IEEE Int. Conf. Neural Netw. **4**, 1942–1948 (1995)
16. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Struct. Multidisc. Optim. **26**, 369–395 (2004)
17. Miettinen, K.: Nonlinear Multi-objective Optimization. International Series in Operations Research & Management Science, vol. 12. Springer, Norwell (1999)
18. Otsu, N.: A threshold selection method from gray level histograms. IEEE Trans. Syst. Man Cybern. **9**, 62–66 (1972)
19. Pal, S.K., Shankar, B.U., Mitra, P.: Granular computing, rough entropy and object extraction. Pattern Recogn. Lett. **26**, 2509–2517 (2005)
20. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Info. Sci. **177**, 3–27 (2007)
21. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data, Kluwer Academic, Dordrecht (1991)
22. Sahoo, P.K., Arora, G.: A thresholding method based on two dimensional Renyis entropy. Pattern Recogn. **37**, 1149–1161 (2004)
23. Sarkar, S., Das, S., Paul, S., Polley, S., Burman, R., Chaudhuri, S.S.: Multi-level image segmentation based on fuzzy - Tsallis entropy and differential evolution. IEEE Int. Conf. Fuzzy Syst. 1–8 (2013)

24. Sarkar, S., Das, S.: Multi-level image thresholding based on two-dimensional histogram and maximum Tsallis entropy - a differential evolution approach. IEEE Trans. Image Process. **22**(12), 4788–4797 (2013)
25. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379423 (1948)
26. Storn, R., Price, K.: Differential evolution A simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
27. Steuer, R.E.: Multiple Criteria Optimization: Theory, Computation, and Application. Robert E. Krieger Publishing, Malabar (1989)
28. Tadaki, K.T.: The Tsallis entropy and Shannon entropy of a universal probability. IEEE International Symposium on Information Theory, pp. 2111–2115 (2008)
29. Tizhoosh, H.R.: Image thresholding using type II fuzzy sets. Pattern Recogn. **38**, 2363–2372 (2008)
30. Tsang, E.C.C., Wang, C., Chen, C., Wu, C., Hu, Q.: Communication between information systems using fuzzy rough sets. IEEE Trans. Fuzzy Syst. **21**(3), 527–540 (2013)
31. Unnikrishnan, R., Pantofaru, C., Hebert, M.: Towards objective evaluation of image segmentation algorithms. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 929–944 (2007)
32. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
33. Xiao, Y., Cao, Z., Yuan, J.: Entropic image thresholding based on GLGM histogram. Pattern Recogn. Lett. **40**, 47–55 (2014)
34. Xu, Y.: Image decomposition based ultrasound image segmentation by using fuzzy clustering. IEEE Symp. Ind. Electron. Appl. **1**, 6–10 (2009)
35. Xue, J.H., Zhang, Y.J.: Ridler and Calvards, Kittler and Illingworth's and Otsu's methods for image thresholding. Pattern Recogn. Lett. **33**(6), 793–797 (2012)
36. The Berkeley Segmentation Dataset and Benchmark. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

Fusion of 3D Model and Uncalibrated Stereo Reconstruction

Jan Klecka and Karel Horak

Abstract Paper is focused on fusion of topological 3D model and color pictures of the same scene. It is describing method designed for such fusion, based on registration of uncalibrated stereo reconstruction to 3D model. This registration removes the reconstruction ambiguity, thereby makes possible acquiring projection matrices of source cameras. Projection matrices are then used for mapping 3D model into images and coloring its points, during this process has to be checked visibility to separate points covered from camera viewpoint, which should not be colored. Real data experiment has been realized and the results are presented at the end of the paper.

Keywords Data fusion · Stereo reconstruction · Data registration

1 Introduction

Fusion of an uncolored 3D data and color images is a process which results into creation of colored 3D model of the scene is interesting area for research today, because many different 3D sensors have become publicly available in past few years and data from this type of sensors may not contain enough information for further processing or comfortable observing. Our research in this area is specified on minimal a priory information solution. Examples of possible use of our method can be more robust localization in SLAM algorithms or just an easy virtualization of scene.

J. Klecka (✉) · K. Horak
Department of Control and Instrumentation, Brno University of Technology,
Brno 61600, Czech Republic
e-mail: klecka@feec.vutbr.cz

K. Horak
e-mail: horak@feec.vutbr.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems
and Computing 378, DOI 10.1007/978-3-319-19824-8_28

Standard way to approach this problem of this type assumes fixed set camera-3D sensor is available in order to calibration can be find before usage. The calibration is usually acquired using easy-to-detect object, with well-defined size. The object is placed, generally for a multiple times, in fields of view of both sensors from which 3D sensor coordinates system to camera coordinates system correspondences are find. These correspondences are then used to express mathematic formula which represent relations between their coordinates systems – the calibration. With this knowledge any point from 3D sensor scan can be mapped into camera image.

2 Algorithm Outline

As proposed before, we wanted to develop method which lowers the standard ways a priory information assumption, specifically the need of calibration. Our method consist of three steps: Firstly the scene is reconstructed up to projective ambiguity from picture pair. Secondly the uncalibrated reconstruction is registered to 3D scan of the scene, which removes the ambiguity and allows to obtain projection matrices of both pictures. Finally these matrices are used to projection points of 3D scan into input pictures in order to acquire information about their color.

An experiment on real data has been performed to verify correctness of proposed algorithm. The 3D model for the experiment consist of multiple scans of 2D laser scanner SICK LMS 111 with measurement plane oriented vertically and rotated around vertical axis in 65° range with 0.5° resolution. Pictures have been captured by hand held camera (Fig. 1).

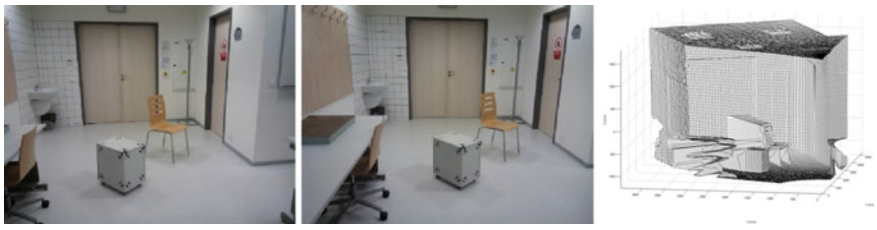


Fig. 1 Experiment input data: Hand-held camera images (left and center) 3D scan of scene (right)

3 Uncalibrated Stereo Reconstruction

Every picture taken by any camera is a projection of in general 3D scene into a plane, so it's obvious that due to this mechanism is one dimension (usually called depth) lost. Stereo reconstruction is process of recovering the information about lost dimension from two pictures of the same scene, taken from slightly different viewpoints. Principle of this method is based on premise that every point in image can be present as a ray in the 3D space. So if projection of a 3D point can be detected in two different images then its space position is found as an intersection of their respective rays. However for a proper determination of position and orientation of a ray in space from image coordinates it is necessary to have calibrated cameras. But, as shown in [1], even without knowledge of calibration, restraints of epipolar geometry allows to obtain some reconstruction, nevertheless only up to projective transformation ambiguity. This process consist of three steps: Firstly must be computed fundamental matrix. Then it is necessary to find as much correspondence points as possible – usually for this step is preferred to use disparity map. Finally fundamental matrix is used to figure out pair of projection matrices witch can be used to protectively ambiguous reconstruction of every point correspondence through triangulation.

3.1 Fundamental Matrix

Fundamental matrix is the algebraic representation of the epipolar geometry [1]. If $\mathbf{x} = \lambda(xy1)^T$ and $\mathbf{x}'$ represent positions of corresponding points in homogenous coordinates then fundamental matrix $\mathbf{F}$ for these two images will satisfy equation for every correspondence:

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = \mathbf{0} \tag{1}$$

Important properties of this matrix are: $\mathbf{F}$ is rank two matrix with seven degrees of freedom – is defined up to scale and $\det(\mathbf{F}) = 0$, any point $\mathbf{x}$ in first image defines on second image so-call epipolar line, on which correspondence to $\mathbf{x}$ can be found, as $l' = \mathbf{F}\mathbf{x}$, all epipolar lines cross each other in one point called epipole e (or e' in second image).

Computing of fundamental matrix can be done by several ways. Our experiment has been realized using following method: Firstly correspondence point has been searched using SIFT [2] feature detector and descriptor. Then outliers in found correspondence has been removed by RANSAC algorithm which periodically compute $\hat{\mathbf{F}}$ using minimal seven point algorithm. And at last final $\mathbf{F}$ has been computed by linear optimization algorithm using all inliers followed by zeroing minimal singular value of linear criterion optimal matrix.

3.2 Disparity Map

A disparity map can be presented as result of very dense correspondence search, usually so dense that disparity map has the same resolution as source images. If $\mathbf{x}$

and $\mathbf{x}'$ again represent positions of corresponding points, then related point in disparity map can be described as $\mathbf{D}(\mathbf{x}) = \mathbf{x}' - \mathbf{x}$.

As mentioned before fundamental matrix constrains a correspondence for any $\mathbf{x}$ to be found on epipolar line l' and to make disparity map computation and representation more simple it is usual to rectify input images so their epipolar lines will become parallel to each other and to one of the axis (generally to the x axis). Then data in disparity map can be scalar because it represents difference only in one dimension.

To compute disparity map experiment data has been rectified and then semi block matching algorithm described in [3] has been used. Rectified images and the grayscale representation of resulting disparity map is presented in the Fig. 2.

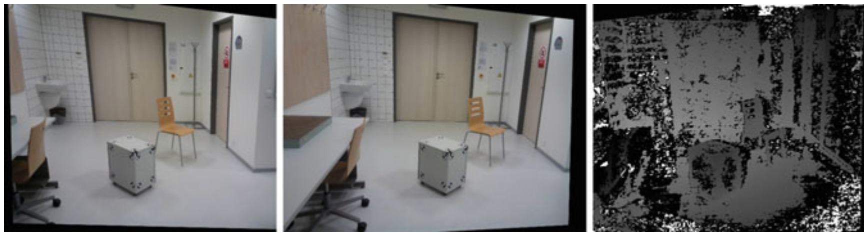


Fig. 2 Rectified images (left and middle), disparity map (right)

3.3 Triangulation

Triangulation in stereo reconstruction can be presented as a process of searching space point $\mathbf{X}$ from its projections in two different images. These projections can be found as correspondence pair of image points $\mathbf{x}$ and $\mathbf{x}'$.

$$\mathbf{x} = \mathbf{P}\mathbf{X} \quad \mathbf{x}' = \mathbf{P}'\mathbf{X} \tag{2}$$

Because true projection matrices are unknown in our case, we use any canonical pair instead which fits to fundamental matrix, namely:

$$\mathbf{P} = [\mathbf{I}|\mathbf{0}] \quad \mathbf{P}' = \left[[\mathbf{e}']_x\mathbf{F} + \mathbf{e}'\mathbf{v}^{\mathrm{T}}|\lambda\mathbf{e}'\right] \tag{3}$$

where $[\mathbf{e}']_x$ is skew symmetric matrix which satisfy $[\mathbf{e}']_x\mathbf{a} = \mathbf{e}' \times \mathbf{a}$, $\mathbf{v}$ is arbitrary vector and λ nonzero scalar. The experiment has been done with $\mathbf{v} = \mathbf{0}$ and $\lambda = 1$.

When projection matrices are defined, there is several way how to solve triangulation in this task. The experiment has been done using linear homogenous method:

$$\begin{bmatrix} x\mathbf{p}^{3\mathrm{T}} - \mathbf{p}^{1\mathrm{T}} \\ y\mathbf{p}^{3\mathrm{T}} - \mathbf{p}^{2\mathrm{T}} \\ x'\mathbf{p}'^{3\mathrm{T}} - \mathbf{p}'^{1\mathrm{T}} \\ y'\mathbf{p}'^{3\mathrm{T}} - \mathbf{p}'^{1\mathrm{T}} \end{bmatrix} \mathbf{X} = \mathbf{0} \tag{4}$$

where $\mathbf{p}^{i\mathrm{T}}$ is row of $\mathbf{P}$.

4 Data Registration

A purpose of data registration is to transform two data sets in such way that they will become spatially consistent. In described algorithm we using this concept to registration the uncalibrated stereo reconstruction to 3D model of reconstructed scene because of the projective ambiguity is removed from such reconstruction. For realization of this registration we decided to use idea of ICP (Iterative Closest Point) algorithm described in [4]. However due to the fact that the ICP is a numerical method a guess of the solution has to be done at first.

The initial guess $\mathbf{H}_0$ of searched projection transformation $\mathbf{H}$ has been got through several handpick reconstruction to 3D model correspondences. Transformation is then derived from these correspondences using homogenous method of least squares.

Then the found transformation is gradually getting precision by periodical appliance of following algorithm: Firstly the closest point in 3D model is find for every point of stereo reconstruction. Secondly some of the worst (the most distant) closest points correspondences are rejected as outliers. Then from remaining correspondences is randomly picked subset, because it turned out that full set of correspondences inliers is usually too large for effective processing. Finally transformation step $\mathbf{H}_i$ is calculated from correspondence subset and is added to so far found transformation.

In our implementation, six correspondences have been handpicked for initial guess calculation. For outliers removal 25 % of most distant correspondence has been rejected. And 5000 samples has been randomly picked for transformation step calculation from inliers (Fig. 3).

5 Coloring 3D Model

Registration stereo reconstruction to 3D model is informational equivalent of a priory camera pair calibration, so relations between images and 3D model coordinates systems are known now and the last problem, which we dealt in this section, is how this knowledge can be used to merge information in images and 3D model. However, because following approach is greatly dependent on format of 3D model

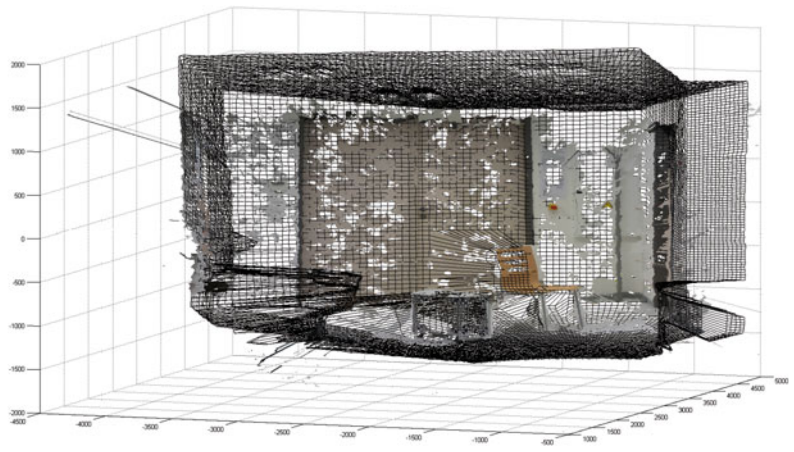


Fig. 3 Registration after 15 iterations

it is multiple ways to solve this problem. We describe method of our implementation used in our experiment. This part of algorithm is dividable to three parts: Firstly we acquire 'true' projection matrices. Secondly every point of 3D model is check if it's visible on images. And finally to every point which is visible at least in one image is assigned color.

5.1 Projection Matrices

For proper explaining process of recovering 'true' projection matrices let us define: canonical projection matrix pair $\mathbf{P}, \mathbf{P}'$ used to obtain uncalibrated reconstruction $\mathbf{X}$ from images $\mathbf{x}, \mathbf{x}'$, projection transformation $\mathbf{H}$ acquired by registration part, 'true' stereo reconstruction $\mathbf{X}_t = \mathbf{HX}$ and 'true' projection matrix pair $\mathbf{P}_t, \mathbf{P}'_t$.

Relation between 'true' and canonical projection matrices is then mathematically defined as:

$$\left.\begin{array}{c}\mathbf{x} = \mathbf{P}_t\mathbf{X}_t = \mathbf{PX} \\ \mathbf{X}_t = \mathbf{HX}\end{array}\right\} \mathbf{P}_t = \mathbf{PH}^{-1} \tag{5}$$

Similar equation can be derive for $\mathbf{P}'_t$.

We verified correctness of acquired matrices by projecting every point of 3D model into both images, thereby we can assigned color to this points by interpolation and after that visually check alignment. One of this way colored model can be seen in Fig. 4 – cause of illustration purposes presented model has been generated from 10 times denser 3D model (obtained by linear interpolation).

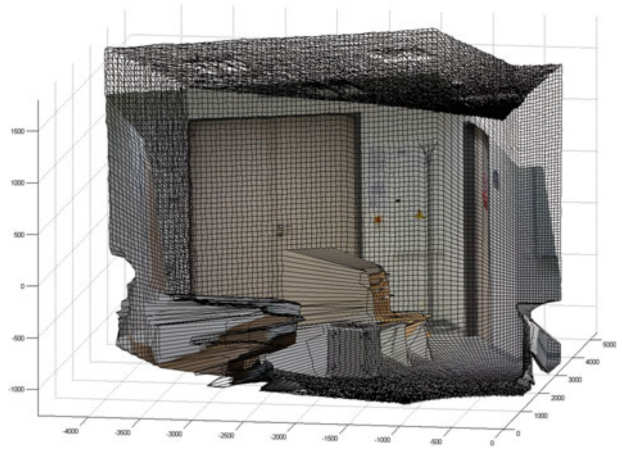


Fig. 4 Texturized 3D model using one image

5.2 Visibility Check

Because 3D sensor and source camera of input images have generally different fields of view, 3D model contains some points which should not be projected into image. This points can be categorized into two groups: points which are projected outside the borders of image and points which would be projected into image but from camera viewpoint are covered behind some closer surface.

Dealing with first group is very easy – it is sufficient to check if coordinates of projected point are inside image borders, but dealing with second group desire more complex approach: Firstly surfaces have to be defined, because we had no information which scanned points form surface we assumed that every neighboring points creating surface. Then we created depth map by interpolating distance from center of projection of projected 3D points into pixels of image. During this process we segmenting points by checking if respective pixels aren't occupied by closer surface (Fig. 5).

5.3 Color Fusion

After visibility check 3D points can be classified into three groups: Points which are not visible in any image, points which are visible only in one image and points which are visible in both images. Dealing with first two groups is very strait forward – to the first group is not assigned any color and to the second group is assigned color from respective image. However, to the points from the last group can be assigned color from both images. Generally it is possible to use this fact for noise reduction through the use of appropriate combination colors from both

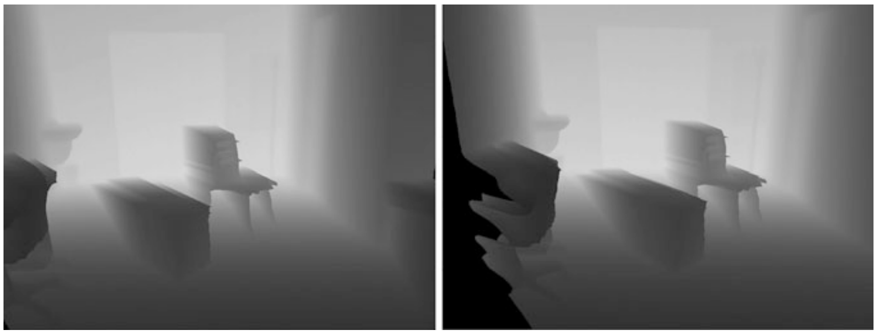
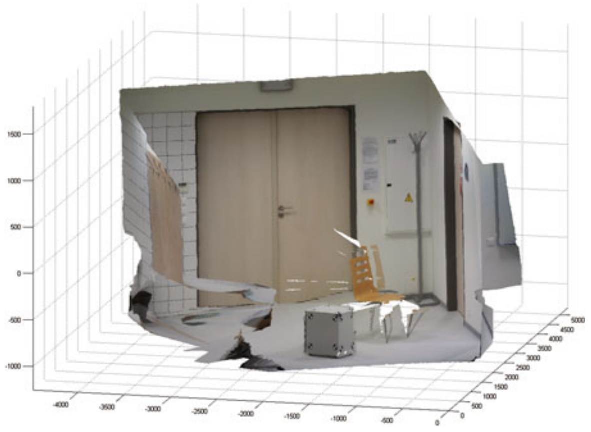


Fig. 5 Depth maps of input images

Fig. 6 The resulting fusion of input data



images. Because in our experiment we dealt with static scene we assigning to this group mean from two respective colors. Results of this process can be seen in Fig. 6.

6 Conclusion

In this paper we describe method for coloring 3D model without need of a priory calibration. Main idea of our method lies in registration stereo reconstruction to colored model. From experiments result we can see that method algorithm is in general correct but alignment of data 3D profile and assigned texture is not perfect (focus e.g. on the chair in Fig. 6). The misalignment is cause by imperfect registration and by nonlinear distortion of the image source camera, which is not compensated in any step. Subject of our further research will be improvement of registration part of algorithm, especially initial guess part, because for usefulness in applications like e.g. SLAM should this part be automated.

Acknowledgments The completion of this paper was made possible by grant No. FEKT-S-14-2429 - "The research of new control methods, measurement procedures and intelligent instruments in automation", and the related financial assistance was provided from the internal science fund of Brno University of Technology and also with assistance Competence Center realized by TACR (reg. number TE01020197).

References

1. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision, 2nd edn. Cambridge University Press, Cambridge (2003). ISBN:05-215-4051-8
2. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
3. Hirschmüller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 328–341 (2008)
4. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)

Visual SLAM Based on Phase Correlation and Particle Filters

Michal Růžička and Petr Mašek

Abstract This paper deals with design of visual SLAM method, which is based on phase correlation and particle filters. This method can be used for localization of autonomous mobile robots inside of buildings. The method contains two parts. The first one is mapping of environment, where the mobile robot operates. For this purpose was used phase correlation and images stitching method. The second one is localization problem, which was solved by particle filters, where a particles weights re-sampling was realized by phase correlation image processing method as well. Localization uses the map, which was created by phase correlation and stitching method.

Keywords Phase correlation · Particle filters · Visual SLAM · Localization · Mapping · Stitching · Computer vision · Image processing · Mobile robot · Discrete fourier transform · Inverse discrete fourier transform · Hanning window

1 Introduction

This paper deals with design of visual SLAM method, which combines phase correlation and particle filters methods to reach SLAM functionality. Method uses one cam-era only, which is located perpendicular to the ceiling. During the mobile robot movement is continuously creating ceiling map by phase correlation and stitching method, where phase correlation method is able to register affine transformations between two similar images. This means translation, rotation and scale. Created map of ceiling is used for localization. Particle filters were used for

M. Růžička (✉) · P. Mašek
Faculty of Mechanical Engineering: Institute of Automation and Computer Science, Brno University of Technology, Technická 2896/2616 69, Brno, Czech Republic
e-mail: y110384@stud.fme.vutbr.cz

P. Mašek
e-mail: y70232@stud.fme.vutbr.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_29

localization problem. Particle filters contain mobile robot motion model and particles re-sampling specific rules to compute weights of each particle, which are based on phase correlation as well.

This paper is organized as follows. The Sect. 2 contains environment mapping description, which is based on the phase correlation. The Sect. 3 deals with local-ization part, which uses particle filters with Ackerman's chassis motion model. Practical experiments are described in the Sect. 4. Section 5 is reserved for conclusions and future works.

2 Mapping

This section deals with environmental mapping problem. The visual SLAM was designed for environment inside of buildings, where the mobile robot operates. The ceiling represents mobile robot map in this case. Visual SLAM uses single camera only with low frame resolution 320×240. Frames are sequentially taken and processed by phase correlation method, during the mapping task.

The next subsection explains the concept of phase correlation. In the next step are images stitched together into a big single frame, which represents the mobile robot map. Previously mentioned step is described in Sect. 2.2.

2.1 Phase Correlation

The phase correlation is able to register changes between two similar signals. The camera frame is represented by two dimensional matrix of pixels, which can be considered as signal. This means the phase correlation can be used for image processing purpose. Image processing version of phase correlation is method for register of affine transformation between two similar images. Translation, rotation and scale are considered to image affine transformations. Phase correlation uses discrete Fourier transformation (1) [6, 7, 11], inverse discrete Fourier transformation (2) [6, 7, 11] and correlation formula (3) [1–4, 11, 12]. These three methods are base of phase correlation. It is worth mentioning constants in correlation formula (3). If variables p and q converge to zero, than we are talking about phase correlation. On the other side when variables p and q converge to one, we are talking about cross correlation.

The main advantage of phase correlation, against to cross correlation, is sharpen global peak. This means the global peak is easier to locate against to peak of cross correlation. Ideal case for phase correlation would be, than variable p and q equal zero, but convergence to zero arranges invariance against to division by zero. Table 1 contains simplified algorithm of phase correlation.

$$F(\xi,\eta) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(\mathrm{x},y)e^{-\frac{2\pi i}{N}(\mathrm{x}\xi+\mathrm{y}\eta)} \tag{1}$$

$$D^{-1}\{F\}(\mathrm{x},\mathrm{y}) = \frac{1}{N^2}\sum_{\xi=0}^{N-1}\sum_{\eta=0}^{N-1} F(\xi,\eta)e^{\frac{2\pi i}{N}(\mathrm{x}\xi+\mathrm{y}\eta)} \tag{2}$$

$$P_{f_1,f_1}^{p,q}(\mathrm{x},\mathrm{y}) = D^{-1}\left\{\frac{F_1(\xi,\eta)\cdot F_2^*(\xi,\eta)}{(|F_1(\xi,\eta)|+p)\cdot(|F_2(\xi,\eta)|+q)}\right\} \tag{3}$$

Table 1 Phase correlation algorithm [1–4, 11]

1.	Load two images in gray scale *f1* and *f2*
2.	Put each image to square image with size *N*, and centre this to obtain two images *fb1*, *fb2*
3.	Multiply *fb1* and *fb2* by Hanning window image to obtain images *fw1*, *fw2*
4.	Apply discrete Fourier transform to *fw1* and *fw2*. Compute the magnitude and switch to logarithmic scale to obtain specters *Al1* and *Al2*
5.	Convert *Al1* and *Al2* from Cartesian coordinate system to Logarithmic-polar coordinate system to obtain spectres *Alpl1* and *Alpl2*
6.	Compute correlation for *Alpl1* and *Alpl2* to obtain spectre C
7.	Apply inverse discrete Fourier transform to obtain *F* image
8.	Locate global peak in *F* image
9.	Apply weighted centroid to get global peak in sub-pixel phase correlation accuracy
10.	Get scale and rotation from global peak point. Scale: $\ln(\xi)$. Rotation: η
11.	Apply affine transformation (rotation and scale) to image *f2* to get image *f2rs*
12.	Put *f2rs* to square image with size *N*, and centre this to obtain two images *fb2rs*
13.	Multiply *fb2rs* by Hanning window image to obtain images *fw2rs*
14.	Apply discrete Fourier transform to *fw2rs*. Compute the magnitude and switch to logarithmic scale to obtain specters *Al2rs*
15.	Compute correlation for *Al1* and *Al2rs* to obtain spectre *C*
16.	Apply inverse discrete Fourier transform to obtain *F* image
17.	Locate global peak in *F* image
18.	Apply weighted centroid to get global peak in sub-pixel phase correlation accuracy [5]
19.	Get shift from global peak

This algorithm is able to get translation with sub-pixel accuracy in hundredths. The estimation of rotation accuracy is approximately in tenths. The estimation accuracy of scale is in hundredths. This is valid for high resolution frames. The accuracy for low resolution frames, which visual SLAM uses, is worse but it still acceptable. Couple of images, to be registered, have to be overlapped to each other. The higher the image resolution can thus be smaller overlap.

2.2 Map Stitching

In previous section was described phase correlation method, the output is rotated and scaled image into another, where the shift between two images is known.

Map stitching method works as follows. Sequentially taken output frames, from phase correlation method are stitched into one image by known shift between current and previous image frame, which was stitched to the map previously. The map dynamically rises during stitching process. You should be aware of one the problem. The shift is real number, due to phase correlation sub-pixel accuracy in hundredths, but image columns and rows are inherently natural numbers. It is evident, that images cannot be stitched with sub-pixel accuracy. Algorithm resizes map and image to be stitch by chosen constant. The shift is multiplied by this constant as well. After that images can be stitched with considerable accuracy. Map is resized by the same constant at the end. It is still important to mention that the image was not stitched cells but only the right half. This method is computational demanding, due to this fact currently used method will be replaced by more efficient method. Table 2 contains stitching algorithm overview.

Table 2 Stitching algorithm

1.	Get rotated and scaled image *f2RS* and shift s from phase correlation
2.	Get sub-mat of $f2RS_{O,T}$ To obtain $f2RSRH_{O,P}$, where $P \in <\frac{T}{2}, T>$
3.	Resize *Map* by constant r. $Map_{M,N} \rightarrow Map_{Y,Z}$, where $Y = r \cdot M, Z = r \cdot N$
4.	Resize *f2RSRH* by constant r. $f2RSRH_{O,P} \rightarrow f2RSRH_{O*r,P*r}$
5.	Multiply s *by* r
6.	Stitch *f2RSRH* to *Map*
7.	Resize *Map* by constant r. $Map_{Y,Z} \rightarrow Map_{M/r,N/r,}$ where $M = Y/r, N = Z/r$

3 Localization

Localization is the next part of visual SLAM method. Localization is based on particle filters, which contain Ackerman's chassis motion model (4) (5) [8–10].

$$x = d \cdot \cos(\varphi) \tag{4}$$

$$y = d \cdot \sin(\varphi) \tag{5}$$

The motion model variable φ represents mobile robot orientation, d is travelled distance and variables x, y represents mobile robot location. Particle generator is described by expressions (6) and (7) [10].

$$x_i = d_N \cdot \cos(\varphi_N) \quad (6)$$

$$y_i = d_N \cdot \sin(\varphi_N) \quad (7)$$

Where dN was randomly generated by distribution function with parameters: μ = d, σd = 10 and φN for μ = φ, σφ = 5. Index i represents particle index. Nowadays algorithm uses 25 particles on the algorithm start and after re-sampling, which uses weight for each particle. Weights evaluation is based on the phase correlation for shift registration only. Idea is as follows. What would be seen on camera frame if the robot was located at the place where this particle is located? We create image by this idea from the map. Basically cut of part the map image, where a new image centre represents particles location. This image is compared with real camera image to get shift be-tween both of them. The smaller the Euclidean distance between these images, the higher the weight of particles will be evaluated. After re-sampling are left just 25 particles. The robot location is evaluated as weighted mean of re-sampled particles locations. Table 3 contains more detailed, but even so simplified applied particle filters algorithm in pseudo code [10].

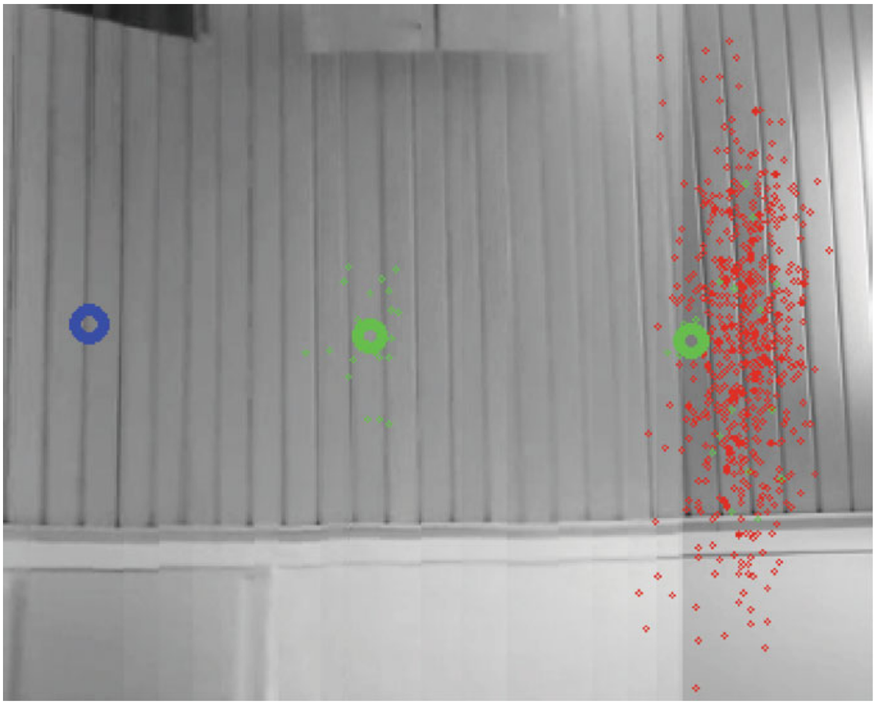


Fig. 1 Particle filters algorithm after two steps

Figure 1 shows two steps of presented particle filter algorithm implementation. Start mobile robot position is sign by blue circle. Algorithm starts just in this state. 25 particles were generated. Weighted mean of particles locations gets mobile robots position. This represents green circle. The next 25 particles are generated for each particle from previously generated particles in this step. Red particles were rejected from algorithm solution. Green particles will be used for next position estimation. The green circle represents mobile robot position as in previous step.

Table 3 Particle filters algorithm [10]

```
σd = 10;
σφ = 5 ;

for(1:ParticlesCount)
{
  dN = distributionFunction(d, σd);
  φN = distributionFunction(φ, σφ);

  particleLocationX = dN * cos * (φN);
  particleLocationY = dN * sin * (φN);
}
while(true)
{
  for(1:ActiveParticlesCount)
  {
    dN = distributionFunction(d, σd);
    φN = distributionFunction(φ, σφ);
    particleLocationX = dN * cos * (φN);
    particleLocationY = dN * sin * (φN);
    particleView = getParticleView(particleLocationX,
particleLocationY)
    cameraView = getCameraView();
                          shift = PhaseCorre-
late(cameraView, particleView);
               particleWeight = shift;
  }
              invertWeights(); // wi = 1/wi
  normalizeWeights(); //weights are defined in closed
interval from 0 to 1
  getTopParticles_80PercentOfParticleCount();
  getRandomParticles_20PercentOfParticleCount();
  getParticlesLocationWeightedMean();
}
```

4 Practical Experiments

This section contains three practical experiments. The first one shows result of two image stitching. The second one shows result of map stitching and the last one shows visual SLAM sample image.

Source video was captured by front camera of LG G2 device, which was attached to mobile robot. There is evident high noise level. The source video was compressed. This means the source video has a poor quality and resolution is 320 × 240 per frame. It is worth mentioning that the camera was not calibrated. Despite these facts we can demonstrate good functionality of mapping method.

4.1 Two Frames Stitching Experiment

This practical experiment demonstrates result of stitching two images from source video. The result is shown in Fig. 2. Boundary between these images shows white line. Boundary is not evident by human eye. When you analyze the image closely then can be found inaccuracy in the lower part of image. This inaccuracy was caused by un-calibrated camera mainly.

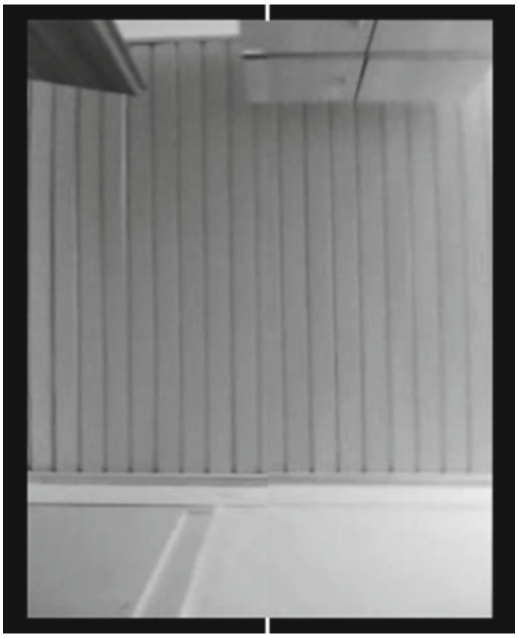


Fig. 2 Two stitched images. White line shows boundary between these images

4.2 Map Stitching Experiment

This practical experiment demonstrates result the stitched map from many frames. Result is shown in the Fig. 3. There are evident a very significant boundary between stitched images in the map. This was caused by non-homogenous light condition by fluorescent bulb during mobile robots movement.

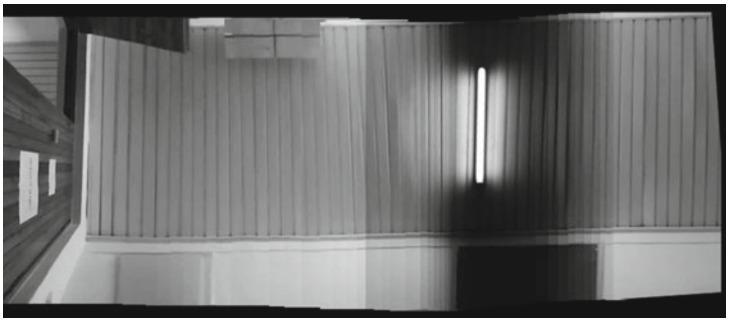


Fig. 3 Sample of stitched map

4.3 Localization Experiment

The last practical experiment demonstrates full visual SLAM functionality. Result is shown in Fig. 4. Blue circle represents mobile robots start position. Green circles equal estimated mobile robots locations. Red points mean particles, which were rejected from calculation during re-sampling process. Finally green points serve particles, which were selected for next computation during re-sampling process.

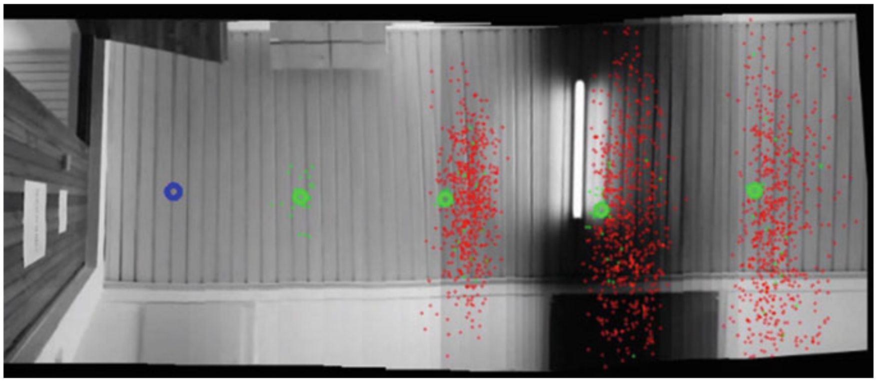


Fig. 4 Visual SLAM sample

5 Conclusions

This paper deals with design of visual SLAM method, which is based on phase correlation and particle filters. The mapping algorithm was described in the Sect. 2. Localization method was described in Sect. 3. Practical experiments, which confirm function of proposal visual SLAM was described in Sect. 4.

This is a new method for mobile robot localization and development will continue. Proposal method was implemented by C++ programming language, but is still computer demanding. Particle filters will be modified for CUDA technology [13] and stitching method will be replaced by more efficient method. The final visual SLAM method should be run on low cost power devices in the real time.

Acknowledgments This research was supported by grant of BUT IGA No. FSI-S-14-2533: "Applied Computer Science and Control".

References

1. Druckmüller, M.: Phase correlation method for the alignment of total solar eclipse images. Astrophys. J. **2**(706), 1605–1608 (2009) ISSN:0004- 637X
2. Berjak, J., Druckmüller, M.: Automatic analysis and recognition of moving objects in the picture by method of phase correlation, pp. 35 (2004)
3. Konecny, Z., Druckmüllerová, H.: Improvement of time-periodical production schedule of the group of products in the group of workplaces through the lot sizes alteration. In: Matousek, R. (ed.) Proceedings of 19th International Conference on Soft Computing—MENDEL 2013, Mendel Journal series vol. 2013, pp. 331–336, Brno (2013)
4. Druckmüllerová, H.: Registration of real images by means of phase correlation. In: Matousek, R. (ed.) Proceedings of Mendel 16th International Conference on Soft Computing—MENDEL 2010, Mendel Journal series vol. 2010, pp. 578–583, Brno (2010). ISSN:1803- 3814
5. Argyriou, V., Vlachos, T.: A study of sub-pixel motion estimation using phase correlation. In: BMVC, pp. 387–396 (2006)
6. Bracewell, R.N.: Affine theorem for two-dimensional fourier transform. Electron. Lett. **29**(3), 304 (1993)
7. De Castro, E., Morandi, C.: Registration of translated and rotated images using finite fourier transforms. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-9**(5), 700–703 (1987)
8. Vechet, S., Ondrousek, V.: Motion planning of autonomous mobile robot in highly populated dynamic environment. In: 9th International Conference on Mechatronics, no. 9, pp. 453–461, Warsaw (2011)
9. Vechet, S.: The rule based path planner for autonomous mobile robot. In: Matousek, R. (ed.) Proceedings of 17th International Conference on Soft Computing—MENDEL 2011, Mendel Journal series vol. 2011, pp. 546–551, Brno University of Technology, Brno (2011)
10. Krejsa, J., Vechet, S.: Infrared beacons based localization of mobile robot. In: Elektronika ir Elektrotechnika, pp. 17–22, Kaunas (2012). doi:10.5755/j01.eee.117.1.1046
11. Druckmüllerová, H.: Phase-correlation based image registration. Supervisor Mgr. Jana Procházková, PhD, Faculty of Mechanical Engineering, Brno University of Technology, Brno (100 pages) (2010)

12. Starha, P., Martisek, D., Matousek, R.: Numerical methods of object reconstruction using the method of moments. In: Proceedings of 20th International Conference on Soft Computing—MENDEL 2014. Mendel series vol. 2014, pp. 241–248, Brno (2014). ISSN:1803- 3814
13. Matousek, R.: HC12: The principle of CUDA implementation. In: Proceedings of 16th International Conference on Soft Computing—MENDEL 2010, Mendel series vol. 2010, pp. 303–308, Brno (2010). ISSN:1803- 3814

Notes on Differential Kinematics in Conformal Geometric Algebra Approach

Jaroslav Hrdina and Petr Vašík

Abstract We consider different elements of a 5D conformal geometric algebra (CGA) as moving geometric objects whose final position is given by a specific kinematic chain. We show the form of the differential kinematics equations for different CGA elements, in particular point pairs, spheres and their centres.

1 Introduction

In robotics, kinematics is the task of determining the robot's precise position and orientation in relation to the controlling parameters given by the robot's construction. If we consider not only the position but also the velocity at which the robot moves, we have to solve so–called differential kinematics, see e.g. [5]. Note that generally the velocity concerned can be both linear and angular or both at the same time. In this paper we show the way to translate the classical differential kinematics notions into the language of conformal geometric algebra. The contribution of the conformal geometric algebra to the kinematics solution is quite straightforward and well described, see e.g. [2] with the application for a robotic snake. The benefit of this is twofold: simplification of the kinematic equations and reduction of the number of operations, which decreases the computational time and thus the complexity of the robot control. We provide the preliminaries about general Clifford (geometric) algebras and conformal geometric algebras (referred to as CGA) particularly in order to apply the algebra operations on the differential kinematics calculation. We show that the form of the differential kinematics equations for different CGA elements, in particular point pairs, spheres and their centres, respectively, is quite analogous.

J. Hrdina (✉) · P. Vašík
Department of Algebra and Discrete Mathematics, Brno University of Technology, Faculty of Mechanical Engineering, Technick á 2896/2, 616 69 Brno, Czech Republic
e-mail: hrdina@fme.vutbr.cz

P. Vašík
e-mail: vasik@fme.vutbr.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_30

2 Conformal Geometric Algebra

Note that the term geometric algebra is used for the real Clifford algebra if the geometric representation of all objects is needed. Thus, in the sequel, we do not distinguish these two notions yet the basic definitions are valid for the Clifford algebras in general.

The pair $(\mathbb{V}, Q)$, where $\mathbb{V}$ is a vector space of dimension n and Q is a quadratic form, is called a quadratic vector space. To define Clifford algebras in coordinates, we start by choosing a basis $e_i,\ i = 1, \dots, n$ of $\mathbb{V}$ and by $I_i,\ i = 1, \dots, n$ we denote the image of e_i under the inclusion $\mathbb{V} \hookrightarrow Cl(\mathbb{V}, Q)$. Then the elements I_i satisfy the relation

$$I_j I_k + I_k I_j = 2B_{jk}1,$$

where 1 is the unit element in the Clifford algebra and B is a bilinear form obtained from Q by polarization. In a quadratic finite dimensional real vector space it is always possible to choose a basis e_i for which the matrix of the bilinear form B has the form

$$\begin{pmatrix} O_r & & \\ & E_s & \\ & & -E_t \end{pmatrix}, \ r + s + t = n,$$

where E_k denotes the $k \times k$ identity matrix and O_k the $k \times k$ zero matrix. Let us restrict to the case $r = 0$, whence B is non-degenerate. Then B defines the inner product of signature (s, t) and we denote the corresponding Clifford algebra by $Cl(s, t)$. For example, $Cl(0, 2)$ is generated by I_1, I_2, satisfying $I_1^2 = I_2^2 = -E$ with $I_1 I_2 = -I_2 I_1$, i.e. $Cl(0, 2)$ is isomorphic to the quaternions $\mathbb{H}$. Now, let $\mathbb{R}^{4,1}$ be a vector space $\mathbb{R}^5$ equipped with scalar product of signature (4, 1), i.e. the bilinear form B is of the form

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \tag{1}$$

and let us consider the corresponding Clifford algebra $Cl(4, 1)$ with the basis $\{e_1, e_2, e_3, e_+, e_-\}$. Note that this space satisfies the definition of the conformal geometric algebra, which in general is a projective embedding of the Euclidean space $\mathbb{R}^{p,q}$ into $\mathbb{R}^{p+1,q+1}$ and in our particular case it is a space widely studied in robotics, precisely a projective embedding of $\mathbb{R}^3$ into $\mathbb{R}^{4,1}$.

To describe the elements of $\mathcal{O} := Cl(4, 1)$ we have to compute free, associative, distributive algebra over the set $\{e_1, e_2, e_3, e_+, e_-\}$ such that the following identities are satisfied:

$$e_1^2 = e_2^2 = e_3^2 = e_+^2 = 1, \ e_-^2 = -1,$$

Table 1 Basis of $Cl(4,1)$

scalars	1
vectors	$e_1,\ e_2,\ e_3,\ e_+,\ e_-$
bivectors	$e_1e_2,\ e_1e_3,\ e_1e_+,\ e_1e_-,$
	$e_2e_3,\ e_2e_+,\ e_2e_-,\ e_3e_+,$
	e_3e_-, e_+e_-
trivectors	$e_1e_2e_3,\ e_1e_2e_+,\ e_1e_2e_-,\ e_1e_3e_+,\ e_1e_3e_-,\ e_1e_+e_-,$
	$e_2e_3e_+,\ e_2e_3e_-,\ e_2e_+e_-,\ e_3e_+e_-$
4-vectors	$e_1e_2e_3e_+,\ e_1e_2e_3e_-,\ e_1e_2e_+e_-,\ e_1e_3e_+e_-,\ e_2e_3e_+e_-$
pseudoscalar	$e_1e_2e_3e_+e_-$

$$e_ie_j = -e_je_i,\ i \neq j,\ i,j \in \{1,2,3,+,-\}.$$

In this case we get $2^5 = 32$ dimensional vector space, see Table 1.

Let us note that the *geometric product* on $\mathbb{R}^{4,1}$ coincides with the scalar product:

$$\begin{aligned}&(x_1e_1 + x_2e_2 + x_3e_3 + x_+e_+ + x_-e_-)(y_1e_1 + y_2e_2 + y_3e_3 + y_+e_+ + y_-e_-)\\&= x_1y_1e_1^2 + x_2y_2e_2^2 + x_3y_3e_3^2 + x_+y_+e_+^2 + x_-y_-e_-^2\\&= x_1y_1 + x_2y_2 + x_3y_3 + x_+y_+ - x_-y_-\end{aligned}$$

and the norm on $\mathbb{R}^{4,1}$ can be understood as the vector square: $x^2 = ||x||^2,\ x \in \mathbb{R}^{4,1}$. Now, we define two additional products based on the geometric one for any $u, v, \in \mathbb{R}^{4,1}$, *dot product* and *wedge product*, respectively:

$$\begin{aligned}u \cdot v &= \frac{1}{2}(uv + vu),\\ u \wedge v &= \frac{1}{2}(uv - vu)\end{aligned}$$

and consequently the geometric product formula:

$$uv = u \cdot v + u \wedge v.$$

To work with the CGA effectively, we have to define a new basis of $\mathbb{R}^{4,1}$ as the set $\{e_1,\ e_2,\ e_3,\ e_0,\ e_\infty\}$ such that $e_0 = \frac{1}{2}(e_- + e_+)$ and $e_\infty = (e_- - e_+)$. Note that the matrix of the appropriate bilinear form in this basis changes from (1) to

$$\begin{pmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&0&-1\\0&0&0&-1&0\end{pmatrix}.$$

Then the following properties hold:

$$\begin{aligned}
e_0^2 &= \frac{1}{4}(e_-^2 - e_-e_+ - e_+e_- + e_+^2) = 1 - 0 - 1 = 0,\\
e_\infty^2 &= \frac{1}{4}(e_-^2 + e_-e_+ + e_+e_- + e_+^2) = 1 + 0 - 1 = 0,\\
e_\infty e_0 &= \frac{1}{2}(e_- + e_+)(e_- - e_+) = -\frac{1}{2} - \frac{1}{2}e_-e_+ + \frac{1}{2}e_+e_- - \frac{1}{2} = -1 - (e_- \wedge e_+),\\
e_0 e_\infty &= \frac{1}{2}(e_- - e_+)(e_- + e_+) = -\frac{1}{2} + \frac{1}{2}e_-e_+ - \frac{1}{2}e_+e_- - \frac{1}{2} = -1 + (e_- \wedge e_+),
\end{aligned}$$

i.e. finally

$$e_\infty e_0 = -e_0 e_\infty - 2.$$

In CGA, the fundamental geometric objects are represented as follows. The point x with the coordinates (x_1, x_2, x_3) in $\mathbb{R}^3$ is represented as an element P of $Cl(4,1)$ by

$$P = x + \frac{1}{2}x^2 e_\infty + e_0.$$

The sphere S with the centre $x = x_1e_1 + x_2e_2 + x_3e_3$ and the radius r is given by

$$S = x_1e_1 + x_2e_2 + x_3e_3 - \frac{1}{2}r^2 e_\infty.$$

A new geometric object appears referred to as a point pair P given by the wedge of two points P_1, P_2:

$$P = P_1 \wedge P_2.$$

Note that for some of the robots description, e.g. for the robotic snakes, this particular object is of high importance, see [2].

3 Point Pair Computations

We present the CGA operations of translation and rotation applied on a pair point in order to show the important properties and the compliance with the classical geometric understanding.

Let $P_1 = [0, 1, 0]$ and $P_2 = [0, -1, 0]$ be two points in $\mathbb{R}^3$, then $P_1 = e_2 + \frac{1}{2}e_\infty + e_0$ and $P_2 = -e_2 + \frac{1}{2}e_\infty + e_0$ are the corresponding vectors in CGA. The point pair $P = P_1 \wedge P_2$ is given by the following computations:

$$P = P_1 \wedge P_2 = \frac{1}{2}e_2 \wedge e_\infty + e_2 \wedge e_0 - \frac{1}{2}e_\infty \wedge e_2 + \frac{1}{2}e_\infty \wedge e_0 - e_0 \wedge e_2 \qquad (2)$$

$$+ \frac{1}{2}e_0 \wedge e_2 e_\infty = e_2 \wedge e_\infty + 2e_2 \wedge e_0$$

$$= e_2 e_\infty + 2e_2 e_0. \qquad (3)$$

On the other hand, geometrically the points P_1 and P_2 are the intersections of the following three spheres in $\mathbb{R}^3$:

$$\begin{aligned} S_1 :& \quad (x-1)^2 + y^2 + z^2 = 2, \\ S_2 :& \quad (x+1)^2 + y^2 + z^2 = 2, \\ S_3 :& \quad x^2 + y^2 + (z-1)^2 = 2 \end{aligned}$$

and consequently

$$\begin{aligned} S_1 &= e_1 + \frac{1}{2}e_\infty + e_0 - e_\infty = e_1 - \frac{1}{2}e_\infty + e_0, \\ S_2 &= -e_1 + \frac{1}{2}e_\infty + e_0 - e_\infty = -e_1 - \frac{1}{2}e_\infty + e_0 \text{ and} \\ S_3 &= e_3 + \frac{1}{2}e_\infty + e_0 - e_\infty = e_3 - \frac{1}{2}e_\infty + e_0 \end{aligned}$$

are the corresponding vectors in CGA. The intersections of geometric objects A and B in CGA are given as the wedge product $A \wedge B$ (or dually by $A^* \cdot B$), where the operations $\wedge$ and $\cdot$ are extended on all elements of CGA. Consequently, the following property for wedge product of a vector P and an arbitrary r–vector Q holds:

$$P \wedge Q = \frac{1}{2}\left(PQ + (-1)^r QP\right)$$

and similarly for a dot product we have

$$P \cdot Q = \frac{1}{2}\left(PQ + (-1)^{r+1} QP\right).$$

Now we calculate:

$$\begin{aligned} S_1 \wedge S_2 \wedge S_3 &= (-\frac{1}{2}e_1 \wedge e_\infty + e_1 \wedge e_0 + \frac{1}{2}e_\infty \wedge e_1 - \frac{1}{2}e_\infty \wedge e_0 - e_0 \wedge e_1 \\ &\quad - e_0 \wedge \frac{1}{2}e_\infty) \wedge S_3 = (-e_1 \wedge e_\infty + 2e_1 \wedge e_0) \wedge (e_3 - \frac{1}{2}e_\infty + e_0) \\ &= -e_1 \wedge e_\infty \wedge e_3 + 2e_1 \wedge e_0 \wedge e_3 - 2e_1 \wedge e_0 \wedge \frac{1}{2}e_\infty - e_1 \wedge e_\infty \wedge e_0 \\ &= -e_1 e_\infty e_3 + 2e_1 e_0 e_3. \end{aligned}$$

To obtain the same expression as in (2), we have to compute $(S_1 \wedge S_2 \wedge S_3)^*$, where $A^* = -Ae_-e_+e_3e_2e_1$ denotes the dual to A in CGA and the operation $A \mapsto A^*$ is realized by the multiplication of the inverse unit pseudoscalar. Note that the duality is just an algebraic operation and does not affect the object geometrically, it is indeed just a transformation between the so–called IPNS and OPNS representation, see e.g. [9]. The result is the following:

$$\begin{aligned}(S_1 \wedge S_2 \wedge S_3)^* &= -(-e_1e_\infty e_3 + 2e_1e_0e_3)e_-e_+e_3e_2e_1\\ &= e_1e_\infty e_3e_-e_+e_3e_2e_1 - 2e_1e_0e_3e_-e_+e_3e_2e_1\\ &= e_\infty e_2e_-e_+ + 2e_0e_2e_-e_+\\ &= e_\infty e_2e_-e_+ + 2e_0e_2e_-e_+ = -e_\infty e_2 - 2e_0e_2 = P.\end{aligned}$$

In CGA (in GA generally), rotations and translations are realized by conjugation

$$O \mapsto TO\tilde{T},$$

where T is the appropriate multi-vector from $\mathcal{O}$ and $\tilde{T}$ denotes so–called reverse, i.e. it is an operation that reverses the order of the basis vectors in the coordinate expression of the elements of T. For instance, the translation in the direction $t = t_1e_1 + t_2e_2 + t_3e_3$ is realized by the multi-vector

$$T = 1 - \frac{1}{2}te_\infty.$$

To translate the point pair $P = e_\infty e_2 + 2e_0e_2$ in the direction of e_1 we have to calculate

$$\begin{aligned}TP\tilde{T} &= (1 - \frac{1}{2}e_1e_\infty)(e_2e_\infty + 2e_2e_0)(1 + \frac{1}{2}e_1e_\infty)\\ &= (e_2e_\infty + 2e_2e_0 - e_1e_\infty e_2e_0)(1 + \frac{1}{2}e_1e_\infty)\\ &= e_2e_\infty + 2e_2e_0 - e_1e_\infty e_2e_0 + e_2e_0e_1e_\infty - \frac{1}{2}e_1e_\infty e_2e_0e_1e_\infty\\ &= e_2e_\infty + 2e_2e_0 + e_1e_2e_\infty e_0 + e_1e_2e_0e_\infty + \frac{1}{2}e_\infty e_2e_0e_\infty\\ &= e_2e_\infty + 2e_2e_0 + e_1e_2(-e_0a_\infty - 2) + e_1e_2e_0e_\infty + \frac{1}{2}e_\infty e_2(-e_\infty e_0 - 2)\\ &= e_2e_\infty + 2e_2e_0 - e_1e_2e_0e_\infty + e_1e_2e_0e_\infty - 2e_1e_2 - e_\infty e_2\\ &= 2e_2e_\infty + 2e_2e_0 - 2e_1e_2.\end{aligned}$$

In fact, we move the points $[0, 1, 0]$ and $[0, -1, 0]$ to the points $[1, 1, 0]$ and $[1, -1, 0]$, i.e. we move the point pair P to the point pair

$$(e_1 + e_2 + e_\infty + e_0) \wedge (e_1 - e_2 + e_\infty + e_0)$$

but after the evaluation we get

$$\begin{aligned}&(e_1 + e_2 + e_\infty + e_0) \wedge (e_1 - e_2 + e_\infty + e_0)\\&= e_2 \wedge e_1 + e_\infty \wedge e_1 + e_0 \wedge e_1 - e_1 \wedge e_2 - e_\infty \wedge e_2 - e_0 \wedge e_2\\&\quad + e_1 \wedge e_\infty + e_2 \wedge e_\infty + e_0 \wedge e_\infty + e_1 \wedge e_0 + e_2 \wedge e_0 + e_\infty \wedge e_0\\&= -2e_1e_2 + 2e_2e_\infty + 2e_2e_0,\end{aligned}$$

which is the same multi-vector.

Let us now focus on the rotation. As mentioned above, such transformation is again represented by conjugation of a particular element denoted by R. More precisely, if we consider $L = a_1e_2e_3 + a_2e_1e_3 + a_3e_1e_2$ to be an axis, then the multi-vector

$$R = \cos\frac{\varphi}{2} - L\sin\frac{\varphi}{2}$$

represents the rotation around the axis L by the angle φ. For instance, the multi-vector

$$R = \cos\frac{\varphi}{2} - \sin\frac{\varphi}{2}e_1e_2$$

represents the rotation around "e_3". Thus to rotate the point pair P around e_3 by the angle φ we have to compute

$$\begin{aligned}RP\tilde{R} &= (\cos\frac{\varphi}{2} - \sin\frac{\varphi}{2}e_1e_2)(e_2e_\infty + 2e_2e_0)(\cos\frac{\varphi}{2} + \sin\frac{\varphi}{2}e_1e_2)\\&= (\cos\frac{\varphi}{2}e_2e_\infty - \sin\frac{\varphi}{2}e_1e_\infty + 2\cos\frac{\varphi}{2}e_2e_0 - 2\sin\frac{\varphi}{2}e_1e_0)\\&\qquad\qquad\qquad\qquad\qquad\qquad(\cos\frac{\varphi}{2} + \sin\frac{\varphi}{2}e_1e_2)\\&= \cos^2\frac{\varphi}{2}e_2e_\infty - \sin\frac{\varphi}{2}\cos\frac{\varphi}{2}e_1e_\infty + 2\cos^2\frac{\varphi}{2}e_2e_0 - 2\sin\frac{\varphi}{2}\cos\frac{\varphi}{2}e_1e_0\\&\quad + \cos\frac{\varphi}{2}\sin\frac{\varphi}{2}e_\infty e_1 + \sin^2\frac{\varphi}{2}e_\infty e_2 + 2\cos\frac{\varphi}{2}\sin\frac{\varphi}{2}e_0e_1 + 2\sin^2\frac{\varphi}{2}e_0e_2\\&= (\cos^2\frac{\varphi}{2} - \sin^2\frac{\varphi}{2})e_2e_\infty + 2(\cos^2\frac{\varphi}{2} - \sin^2\frac{\varphi}{2})e_2e_0 - 2\sin\frac{\varphi}{2}\cos\frac{\varphi}{2}e_1e_\infty\\&\quad - 4\sin\frac{\varphi}{2}\cos\frac{\varphi}{2}e_1e_0\\&= (\cos\varphi)e_2e_\infty + 2(\cos\varphi)e_2e_0 - (\sin\varphi)e_1e_\infty - 2(\sin\varphi)e_1e_0.\end{aligned}$$

In the same way, we can rotate the points $[0, 1, 0]$ and $[0, -1, 0]$ and obtain new pair of points with coordinates $[-\sin\varphi, \cos\varphi, 0]$ and $[\sin\varphi, -\cos\varphi, 0]$, respectively, i.e. if the resulting couple is understood as the point pair in CGA, it is realized as:

$$(-(\sin\varphi)e_1 + (\cos\varphi)e_2 + \frac{1}{2}e_\infty + e_0) \wedge ((\sin\varphi)e_1 - (\cos\varphi)e_2 + \frac{1}{2}e_\infty + e_0).$$

Let us now verify this fact by evaluating the above wedge product:

$$
\begin{aligned}
&(-(\sin\varphi)e_1 + (\cos\varphi)e_2 + \frac{1}{2}e_\infty + e_0) \wedge ((\sin\varphi)e_1 - (\cos\varphi)e_2 + \frac{1}{2}e_\infty + e_0)\\
&= (\cos\varphi)(\sin\varphi)e_2e_1 + \frac{1}{2}(\sin\varphi)e_\infty e_1 + (\sin\varphi)e_0e_1 + (\sin\varphi)(\cos\varphi)e_1e_2\\
&\quad - \frac{1}{2}(\cos\varphi)e_\infty e_2 - (\cos\varphi)e_0e_2 - (\sin\varphi)\frac{1}{2}e_1e_\infty + (\cos\varphi)\frac{1}{2}e_2e_\infty + \frac{1}{2}e_0 \wedge e_\infty\\
&\quad - (\sin\varphi)e_1e_0 + (\cos\varphi)e_2e_0 + \frac{1}{2}e_\infty \wedge e_0 = \frac{1}{2}(\sin\varphi)e_\infty e_1 + (\sin\varphi)e_0e_1\\
&\quad - \frac{1}{2}(\cos\varphi)e_\infty e_2 - (\cos\varphi)e_0e_2 - (\sin\varphi)\frac{1}{2}e_1e_\infty + (\cos\varphi)\frac{1}{2}e_2e_\infty\\
&\quad - (\sin\varphi)e_1e_0 + (\cos\varphi)e_2e_0\\
&= (\sin\varphi)e_\infty e_1 + 2(\sin\varphi)e_0e_1 - (\cos\varphi)e_\infty e_2 - 2(\cos\varphi)e_0e_2.
\end{aligned}
$$

As we received the same multi-vector, this example shows the compliance of the CGA operations with the classical geometric objects manipulation.

4 Differential Kinematics

By differential kinematics we understand the description of a motion not by the coordinates in the final state but by means of the velocity, i.e. by the length and the direction of the velocity vector. As we consider both translations and rotations both linear and angular velocities are involved. If we consider a robot whose final position is given by a kinematic chain (i.e. by the system of kinematic equations), the formulae for the differential kinematics are obtained simply as its total derivative. Let us consider the following kinematic chain in CGA:

$$M_nM_{n-1}\cdots M_1P\tilde{M}_1\cdots\tilde{M}_{n-1}\tilde{M}_n, \tag{4}$$

where M_i denotes the so–called motor (the abbreviation of "moment and vector," i.e. the element performing a screw motion). Note that denoting the translation and rotation elements by T and R, respectively, the motor M can be expressed as

$$M = TR\tilde{T},$$

and consequently for a rotation by a particular angle θ we have

$$M_\theta = TR\tilde{T} = \cos\frac{\theta}{2} - \sin\frac{\theta}{2}L = \mathrm{e}^{-\frac{\theta}{2}L},$$

where L denotes the appropriate rotation axis, see [5]. Thus, in the sequel, we consider the motors in the kinematic chain to be of the form

$$M_i = \mathrm{e}^{-\frac{1}{2}q_i L_i}.$$

At this point, by differentiation of the kinematic chain (4) we obtain [5, 9]

$$\dot{P} = \sum_{j=1}^{n} [P \cdot L'_j] dq_j, \tag{5}$$

where

$$L'_j = \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i}$$

and

$$[P \cdot L'_j] = \frac{1}{2}(PL'_j - L'_j P).$$

Note that (5) represents the point P motion. In the rest of the paper we shall derive similar equations for other geometric objects.

Lemma 1 *Let P_1 and P_2 be two moving points and let their final position be determined the same kinematic chain* (4). *Then the differential kinematics of the point pair $P_p = P_1 \wedge P_2$ is given by*

$$\dot{P}_p = \sum_{j=1}^{n} [P_p \cdot L'_j] dq_j.$$

Proof Let us consider trajectory of a point pair $P_p = P_1 \wedge P_2$ motion, i.e. a couple of curves containing the points P_1 and P_2, respectively. Then the derivative of a point pair $P_p = P_1 \wedge P_2$ can be understood as a derivative with respect to the trajectory parameter, i.e. time, which is denoted by t. By direct differentiation of a multiplication we obtain:

$$\begin{aligned} \partial_t(P_p) = \partial_t(P_1 \wedge P_2) = \partial_t\Big(\frac{1}{2}(P_1P_2 - (-1)^1 P_2 P_1)\Big) = \frac{1}{2}(\dot{P}_1 P_2 + P_1 \dot{P}_2) \\ - \frac{1}{2}(-1)^1(\dot{P}_2 P_1 + P_2 \dot{P}_1) = \dot{P}_1 \wedge P_2 + P_1 \wedge \dot{P}_2. \end{aligned} \tag{6}$$

If we substitute (5) into (6) we get

$$\begin{aligned}\partial_t(P_p) &= \dot{P}_1 \wedge P_2 + P_1 \wedge \dot{P}_2 = \sum_{j=1}^{n}[P_1 \cdot L'_j]dq_j \wedge P_2 + P_1 \wedge \sum_{j=1}^{n}[P_2 \cdot L'_j]dq_j \\ &= \sum_{j=1}^{n}\frac{1}{2}(P_1L'_j - L'_jP_1) \wedge P_2dq_j + P_1 \wedge \sum_{j=1}^{n}\frac{1}{2}(P_2L'_j - L'_jP_2)dq_j \\ &= \frac{1}{4}\sum_{j=1}^{n}(P_1L'_jP_2 - L'_jP_1P_2 - P_2P_1L'_j + P_2L'_jP_1)dq_j \\ &\quad + \frac{1}{4}\sum_{j=1}^{n}(P_1P_2L'_j - P_1L'_jP_2 - P_2L'_jP_1 + L'_jP_2P_1)dq_j \\ &= \frac{1}{4}\sum_{j=1}^{n}(-L'_jP_1P_2 - P_2P_1L'_j + P_1P_2L'_j + L'_jP_2P_1)dq_j \\ &= \frac{1}{2}\sum_{j=1}^{n}((P_1 \wedge P_2)L'_j - L'_j(P_1 \wedge P_2))dq_j = \sum_{j=1}^{n}[(P_1 \wedge P_2) \cdot L'_j]dq_j,\end{aligned}$$

which completes the proof.

Theorem 1 *Let P_1, P_2, P_3 and P_4 be four moving points whose final position is obtained by means of the same kinematic chain* (4). *Then the differential kinematics of a 1D–sphere $S_1 = P_1 \wedge P_2 \wedge P_3$ and a 2D–sphere $S_2 = P_1 \wedge P_2 \wedge P_3 \wedge P_4$ are given by:*

$$\dot{S}_1 = \sum_{j=1}^{n}[S_1 \cdot L'_j]dq_j,$$

$$\dot{S}_2 = \sum_{j=1}^{n}[S_2 \cdot L'_j]dq_j.$$

Proof We use the same calculations as in the proof of Lemma 1 with $P_2 \wedge P_3$ and $P_2 \wedge P_3 \wedge P_4$ instead of P_2, respectively.

Theorem 2 *Let P be a moving CGA geometric object, particularly a point, line, plane, point pair or a sphere of dimension 1 or 2, respectively. If the final position of P is determined by the kinematic chain* (4), *then the differential kinematics of P is given by the equations* (5).

The proof is a direct consequence of Lemma 1 and Theorem 1. The key fact is that a line is just a 1D–sphere S_1, where $P_3 = e_\infty$ and a plane is a 2D–sphere S_2, where $P_4 = e_\infty$.

The last object whose differential kinematics equations are useful for a robotic snake motion description is a sphere centre.

Theorem 3 *Let c be a centre of a sphere S (including a point pair as a 0D–sphere) whose final position is given by the kinematic chain* (4)*. Then the differential kinematics of c is given by*

$$\dot{c} = \sum_{j=1}^{n} [c \cdot L'_j] dq_j.$$

Proof The centre c of a sphere S is in CGA expressed as $c = Se_\infty \tilde{S}$. If we move a particular axis

$$L_0 = ue_2e_3 + ve_1e_3 + we_1e_2$$

to a general point (x, y, z), the shifted line L will be of the form

$$\begin{aligned} L &= (1 - \frac{1}{2}(xe_1 + ye_2 + ze_3)e_\infty)L_0(1 + \frac{1}{2}(xe_1 + ye_2 + ze_3)e_\infty) \\ &= L_0 - \frac{1}{2}(xe_1 + ye_2 + ze_3)e_\infty L_0 + \frac{1}{2}L_0(xe_1 + ye_2 + ze_3)e_\infty. \end{aligned}$$

In addition, one can easily see that

$$Le_\infty = L_0e_\infty = e_\infty L_0 = e_\infty L,$$

and generally for a particular axis L'_j

$$L'_j e_\infty = e_\infty L'_j.$$

Finally, we conclude the proof by the following direct evaluation:

$$\begin{aligned} \dot{c} &= \partial_t(Se_\infty\tilde{S}) = \dot{S}e_\infty\tilde{S} + Se_\infty\dot{\tilde{S}} \\ &= \sum_{j=1}^{n}[S \cdot L'_j]e_\infty\tilde{S}dq_j + Se_\infty\sum_{j=1}^{n}[\tilde{S} \cdot L'_j]dq_j \\ &= \frac{1}{2}\sum_{j=1}^{n}[SL'_j - L'_jS]e_\infty\tilde{S}dq_j + Se_\infty\frac{1}{2}\sum_{j=1}^{n}[\tilde{S}L'_j - L'_j\tilde{S}]dq_j \\ &= \sum_{j=1}^{n}\frac{1}{2}[SL'_je_\infty\tilde{S} - L'_jSe_\infty\tilde{S}]dq_j + \sum_{j=1}^{n}\frac{1}{2}[Se_\infty\tilde{S}L'_j - Se_\infty L'_j\tilde{S}]dq_j \\ &= \sum_{j=1}^{n}\frac{1}{2}[SL'_je_\infty\tilde{S} - SL'_je_\infty\tilde{S}]dq_j + \sum_{j=1}^{n}\frac{1}{2}[Se_\infty\tilde{S}L'_j - L'_jSe_\infty\tilde{S}]dq_j \\ &= 0 + \sum_{j=1}^{n}\frac{1}{2}[cL'_j - L'_jc]dq_j = \sum_{j=1}^{n}[c \cdot L'_j]dq_j. \end{aligned}$$

Acknowledgments The research was supported by the project NETME CENTRE PLUS (LO1202). The results of the project NETME CENTRE PLUS (LO1202) were co-funded by the Ministry of Education, Youth and Sports within the support programme "National Sustainability Programme".

References

1. Hildenbrand, D.: Foundations of Geometric Algebra Computing. Geometry and Computing, vol. 8. Springer, Berlin (2013)
2. Hrdina, J., Návrat, A., Vašík, P.: 3-link robotic snake control based on CGA. In: Advances in Applied Clifford Algebras (2015)
3. Návrat, A., Matousek, R.: Trident snake control based on CGA. In: Matousek, R. (ed.) Mendel 2015: Recent Advance in Computer Science. AISC, vol. 378, pp. 375–385. Springer, Heidelberg (2015)
4. Perwass, C.: Geometric Algebra with Applications in Engineering. Geometry and Computing, vol. 4. Springer, Berlin (2009)
5. Zamora-Esquivel, J., Bayro-Corrochano, E.: Kinematics and diferential kinematics of binocular robot heads. In: Robotics and Automation, ICRA (2006)
6. Selig, J.M.: Geometric Fundamentals of Robotics. Monographs in Computer Science. Springer, New York (2004)
7. Murray, R.M., Zexiang, L., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)
8. Liljebäck, P., Pettersen, K.Y., Stavdahl, Ø., Gravdahl, J.T.: Snake Robots, Modelling, Mechatronics, and Control. Advances in Industrial Control. Springer, London (2013)
9. González-Jiménez, L., Carbajal-Espinosa, O., Loukianov, A., Bayro-Corrochano, E.: Robust pose control of robot manipulators using conformal geometric algebra. Adv. Appl. Clifford Al. **24**(2), 533–552 (2014)

Trident Snake Control Based on Conformal Geometric Algebra

Aleš Návrat and Radomil Matoušek

Abstract Local controllability of a trident snake robot is solved by means of 5D conformal geometric algebra. The non–holonomic kinematic equations are assembled, their property to be a Pfaff system is discussed and the solution is found. The functionality is demonstrated on a virtual model in CLUCalc programme.

Keywords Conformal geometric algebra · Clifford algebra · Mathematics robotic · Nonholonomic mechanics · Snake robots · Local controllability · Bionics

Mathematics Subject Classification (2010): 93C10 · 15A66

1 Introduction

Within this paper, we consider a trident snake robot, see [8–10], moving on a planar surface. More precisely, it is a robot with a triangular body with three legs when to each leg a pair of wheels is attached and thus the possible movement directions are determined uniquely. The aim is to find the complete local controllability solution in a new geometric form. In terms of generalized coordinates, the non–holonomic forward kinematics equations can be understood as a Pfaff system and its solution as a distribution in the configuration space. Rachevsky–Chow Theorem implies that the appropriate non–holonomic system is locally controllable if the corresponding distribution is not integrable and the span of the Lie algebra generated by the controlling distribution has to be of the same dimension as the configuration space. The

A. Návrat (✉)
Faculty of Mechanical Engineering, Department of Algebra and Discrete Mathematics,
Brno University of Technology,
Technická 2896/2, 616 69 Brno, Czech Republic
e-mail: navrat.a@fme.vutbr.cz

R. Matoušek
Faculty of Mechanical Engineering, Department of Applied Computer Science,
Brno University of Technology,
Technická 2896/2, 616 69 Brno, Czech Republic
e-mail: matousek@fme.vutbr.cz

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems and Computing 378, DOI 10.1007/978-3-319-19824-8_31

spanned Lie algebra is then naturally endowed by a filtration which shows the way to realize the movements by means of the vector field brackets [4, 5]. In our case, the system is locally controllable and the filtration is $(3,6)$.

The classical approach composes the kinematic chain of homogeneous matrices using the moving frame methods and Euler angles [6]. Instead of this, our aim is to use the notions of conformal geometric algebra (CGA), i.e. the subset of a Clifford algebra $Cl(4,1)$ where the Euclidean space $\mathbb{E}_3$ is included by a mapping $x \mapsto x + x^2 e_\infty + e_0$. In this geometric setting, we can easily handle both linear objects and spheres of dimensions 2, 1 and 0, see [1–3].

In particular, the 0–dimensional sphere, referred to as a point pair, is used to derive the kinematic equations and for the control of the non–holonomic robotic snake, consequently. More precisely, to any link of a snake a single point pair is assigned and the mechanism is transformed by rotations and translations. We introduce the forward kinematic equations (1), the differential kinematic equations (2), as well as the non–holonomic conditions (4). We demonstrate the functionality in the CLUCalc software designed for the computations in Clifford algebra, particularly in conformal geometric algebra.

2 Conformal Geometric Algebra – CGA

The classical approach composes the kinematic chain of homogeneous matrices using the moving frame methods and Euler angles, or in advance using the quaternion algebra $\mathbb{H}$ by conjugation $x \mapsto q^{-1}xq$, where we view an Euclidean point x as a quaternion

$$x = (x_1, x_2, x_3) \rightleftharpoons x_1 i + x_2 j + x_3 k.$$

and q is a quaternion given by

$$q = \cos\frac{\theta}{2} + u \sin\frac{\theta}{2},$$

where u is a axis of rotation $u_1 i + u_2 j + u_3 k$. Instead of this, we use the notions of conformal geometric algebra, i.e. the Clifford algebra $Cl(4,1)$ where the Euclidean space $\mathbb{E}_3$ is included by a mapping $x \mapsto x + x^2 e_\infty + e_0$. In this geometric setting, we can easily handle both linear objects and spheres of dimensions 2, 1 and 0. Namely, these objects are simply elements of the algebra and can be transformed and intersected with ease. In addition, rotations, translation, dilations and inversions all become rotations in our 5-dimensional space, see [1–3].

More precisely, let $\mathbb{R}^{4,1}$ denote a vector space $\mathbb{R}^5$ equipped with the scalar product of signature $(4,1)$ and let $\{e_1, e_2, e_3, e_+, e_-\}$ be a basis. The Clifford algebra $Cl(4,1)$ can be described as a free, associative and distributive algebra such that the following identities are satisfied:

$$\{e_1^2 = e_2^2 = e_3^2 = e_+^2 = 1,\ \ e_-^2 = -1\},$$

$$e_i e_j = -e_j e_i,\ i \neq j,\ i,j \in \{1,2,3,+,-\}.$$

Hence we get $2^5 = 32$–dimensional vector space. Let us note that the *geometric product* in the algebra restricted to $\mathbb{R}^{4,1}$ coincides with the scalar product and the norm in $\mathbb{R}^{4,1}$ can be understood as a vector square $x^2 = \|x\|^2$. Next to the geometric product, we define two additional products on $\mathbb{R}^{4,1}$ based on the geometric one for any $u, v, \in \mathcal{O}$, *dot product* and *wedge product*, respectively:

$$u \cdot v = \frac{1}{2}(uv + vu), \quad u \wedge v = \frac{1}{2}(uv - vu)$$

and thus the basis elements are derived as $uv = u \cdot v + u \wedge v$. The definition of these product extends to the whole algebra. Namely, given two basis blades $E_i = e_{a_1} \wedge \cdots \wedge e_{a_k}$ and $E_j = e_{a_1} \wedge \cdots \wedge e_{a_l}$ of grades k and l respectively the wedge (outer) product is defined as

$$E_i \wedge E_j := \langle E_i E_j \rangle_{k+l}$$

while the dot (inner) product is defined as

$$\begin{aligned} E_i \cdot E_j &:= \langle E_i E_j \rangle_{|k-l|}, \quad i,j, > 0 \\ &:= 0, \quad i = 0 \text{ or } j = 0, \end{aligned}$$

where $\langle\ \rangle_k$ is the grade projection into grade k. These products can be used effectively to compute an intersection of geometric objects and distances respectively.

To work with CGA effectively, one defines $e_0 = \frac{1}{2}(e_- + e_+)$ and $e_\infty = (e_- - e_+)$. Consequently, the following properties hold:

$$e_0^2 = 0,\ e_\infty^2 = 0,\ e_\infty e_0 + e_0 e_\infty = -2.$$

Then we can represent the basis geometric elements by the following multi–vectors from $Cl(4,1)$:

Fig. 1 Elements of $\mathcal{C}(4,1)$

Point	$Q = x + \frac{1}{2}x^2 e_\infty + e_0$
Sphere of radius r and center C	$S = C - \frac{1}{2}r^2 e_\infty$
Point pair Q_1, Q_2	$P = Q_1 \wedge Q_2.$

Each geometric transformation (rotation, translation, dilation, inversion) of a geometric object represented by an algebra element O is realized by conjugation $O \mapsto MO\tilde{M}$, where M is an appropriate multi–vector. For instance, the translation in the direction $t = t_1 e_1 + t_2 e_2 + t_3 e_3$ is realized by conjugation by the multi–vector

$$T = 1 - \frac{1}{2} t e_\infty,$$

which can be written as $e^{-\frac{1}{2} t e_\infty}$ and the rotation around the axis L by angle ϕ is realized by conjugation by the multi–vector

$$R = \cos\frac{\phi}{2} - L \sin\frac{\phi}{2}$$

where $L = a_1 e_2 e_3 + a_2 e_1 e_3 + a_3 e_1 e_2$. The rotation can be also written as $e^{-\frac{1}{2}\phi L}$.

3 Control Theory

The locomotion analysis, control and development of the trident snake robot has been described in [8–10]. The robot consists of a body in the shape of an equilateral triangle with circumscribed circle of radius r and three rigid links (also called legs) of constant length l connected to the vertices of the triangular body by three motorised joints. In this paper, we consider $r = 1$ and $l = 1$. To each free link end a pair of passive wheels is attached to provide an important snake-like property that the ground friction in the direction perpendicular to the link is considerably higher than the friction of a simple forward move. In particular, this prevents the slipping sideways. To describe the actual position of a trident snake robot we need the set of 6 generalized coordinates

$$q = (x, y, \theta, \phi_1, \phi_2, \phi_3)$$

as shown in Fig. 1. Hence the configuration space is (a subspace of) $\mathbb{R}^2 \times S^1 \times (S^1)^3$. Note that a fixed coordinate system (x, y) is attached.

To describe our robotic trident snake via CGA we use as a central object the set of three point pairs representing the robot's legs

$$(P_1, P_2, P_3).$$

These point pairs are computed in terms of the wedge product in CGA as $P_i = p_i^b \wedge p_i$ for $i = 1, 2, 3$, where p_i^b are the joints between body and legs and p_i are the ends of legs. On the other hand, the end points p_i and p_i^b respectively of P_i are extracted from the pair point by projections

$$\pm\frac{\sqrt{P_i \cdot P_i} + P_i}{e_\infty \cdot P_i}.$$

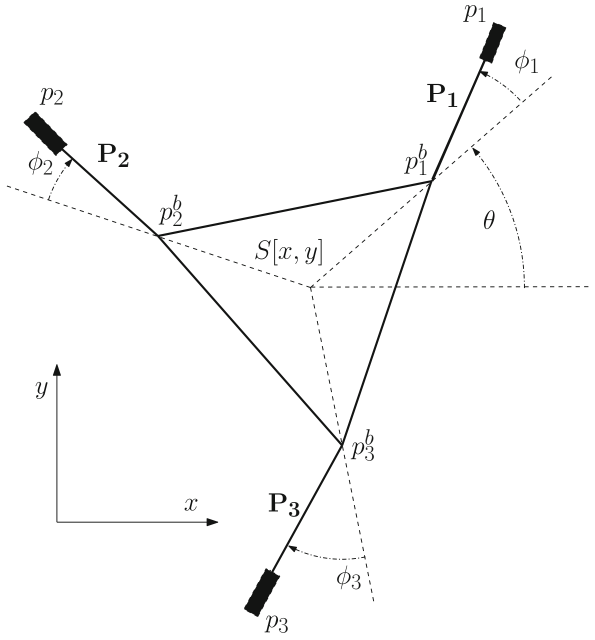


Fig. 2 Trident snake robot model

Consequently, we may freely switch between point pairs and points defining their ends. Of course, not all triples of pair points define a state of the robot. In terms of the CGA inner product, the consistency relations between the point pairs read $p_i^b \cdot p_j^b = -\frac{3}{2}$ for $i \neq j$ and $p_i \cdot p_i^b = -\frac{1}{2}$. These equations tell that the joints have constant distance $\sqrt{3}$ and the length of links is 1 (Fig. 2).

Now to each admissible state (P_1, P_2, P_3) we can assess the kinematic equations. For the trident zero position $q = 0$, three appropriate point pairs are established as

$$P_1(0) = (e_1 + \tfrac{1}{2}e_\infty + e_0) \wedge (2e_1 + 2e_\infty + e_0) = e_{1\infty} - e_{10} - \tfrac{3}{2}e_{\infty 0}$$

$$P_{2,3}(0) = (-\tfrac{1}{2}e_1 \pm \tfrac{\sqrt{3}}{2}e_2 + \tfrac{1}{2}e_\infty + e_0) \wedge (-e_1 \pm \sqrt{3}e_2 + 2e_\infty + e_0)$$

$$= -\tfrac{1}{2}e_{1\infty} + \tfrac{1}{2}e_{10} \pm \tfrac{\sqrt{3}}{2}e_{2\infty} \mp \tfrac{\sqrt{3}}{2}e_{20} - \tfrac{3}{2}e_{\infty 0},$$

where we have used a shortened notation $e_{1\infty} = e_1 \wedge e_\infty$ etc. The particular pair points in a general position $q = (x, y, \theta, \phi_1, \phi_2, \phi_3)$ are obtained by a translation to $[x, y]$ composed by a trident body rotation θ and a rotation of the corresponding leg ϕ_i. In CGA, it is expressed for each $i = 1, 2, 3$ as

$$P_i = R_{\phi_i} R_\theta T_{x,y} P_i(0) \tilde{T}_{x,y} \tilde{R}_\theta \tilde{R}_{\phi_i}, \tag{1}$$

where

$$T_{x,y} = 1 - \tfrac{1}{2}(xe_1 + ye_2)e_\infty,$$
$$R_\theta = \cos\tfrac{\theta}{2} - L_0 \sin\tfrac{\theta}{2},$$
$$R_{\phi_i} = \cos\tfrac{\phi_i}{2} - L_i \sin\tfrac{\phi_i}{2},$$

and where the axes of rotations are given by

$$L_0 = T_{x,y} e_{12} \tilde{T}_{x,y},$$
$$L_i = T_i e_{12} \tilde{T}_i, \text{ where } T_i = 1 - \tfrac{1}{2} p_i^b e_\infty.$$

Note that the same kinematic chain holds for the end points of point pairs.

The CGA approach is convenient also for solving problems of the inverse kinematics. In CGA, it can be done in a geometrically very intuitive way due to its easy handling of intersections of geometric objects like spheres, circles, planes. A basic problem is finding the generalized coordinates in terms of a robot position. In our case, having a state (P_1, P_2, P_3) obtained by (1), we first form the circumscribed circle $C = p_1^b \wedge p_2^b \wedge p_3^b$, its center $S = Ce_\infty \tilde{C}$, and for each $i = 1, 2, 3$ we form a line through the center and the corresponding joint $l_i^b = S \wedge p_i^b \wedge e_\infty$ and a line through the corresponding leg $l_i = p_i^b \wedge p_i \wedge e_\infty$. Then we compute the coordinates via the inner product as

$$x = S \cdot e_1$$
$$y = S \cdot e_2$$
$$\cos\theta = l_1^b \cdot e_{1\infty 0}$$
$$\cos\phi_i = l_i^b \cdot l_i$$

Let us now compute the velocity of the direct kinematics, which is obtained by differentiating (1). It is proved in [3] that the total differential of a general kinematic chain

$$P = R_1 \dots R_n P(0) \tilde{R}_n \dots R_1$$

containing rotations $R_1, \dots R_n$ is equal to

$$dP = \sum_{j=1}^{n} [P \cdot L_j] dq_j,$$

where $[P \cdot L_j]$ is the inner product of the geometric object (in the actual position) and the axis of the rotation R_j. This formula follows basically from the fact that each rotation can be expressed es an exponential. But the same is true for translations. We may view each translation as a degenerate rotation, with an ‘axis‘ containing e_∞.

Hence the formula above holds true also if we allow R_i to be a translation. In our case, the differentiation of kinematic chains (1) for point p_i, $i = 1, 2, 3$, then yields the following expressions:

$$\dot{p}_i = [p_i \cdot e_{1\infty}]\dot{x} + [p_i \cdot e_{2\infty}]\dot{y} + [p_i \cdot L_0]\dot{\theta} + [p_i \cdot L_i]\dot{\phi}_i \tag{2}$$

i.e. we have matrix system $\dot{p} = J\dot{q}$, where

$$J = \begin{pmatrix} p_1 \cdot e_{1\infty} & p_1 \cdot e_{2\infty} & p_1 \cdot L_0 & p_1 \cdot L_1 & 0 & 0 \\ p_2 \cdot e_{1\infty} & p_2 \cdot e_{2\infty} & p_2 \cdot L_0 & 0 & p_2 \cdot L_2 & 0 \\ p_3 \cdot e_{1\infty} & p_3 \cdot e_{2\infty} & p_3 \cdot L_0 & 0 & 0 & p_3 \cdot L_3 \end{pmatrix} \tag{3}$$

Note that we get exactly the same formulas for the differential kinematics of point pairs P_i but we rather compute the end points p_i where the wheels are located.

Then, as the wheels do not slip to the side direction, the velocity constraint condition is satisfied and can be written as

$$\dot{p}_i \wedge P_i \wedge e_\infty = 0. \tag{4}$$

Thus if we substitute (2) in (4), we obtain a system of linear ODEs, which can be written in a form $A\dot{q} = 0$, where the Pfaff matrix A is given by

$$A_{ij} = J_{ij} \wedge P_i \wedge e_\infty,$$

where J is the 'Jacobian' (3). It is easy to see that each A_{ij} is a multiple of $(e_3)^*$. Thus the Pfaff equation $A\dot{q} = 0$ can be solved for A considered as a matrix over the field of functions. The solution gives a control system $\dot{q} = G\mu$, where the control matrix G is a 6×3 matrix spanned by vector fields g_1, g_2, g_3, where

$$\begin{aligned} g_1 &= \cos\theta\partial_x + \sin\theta\partial_y + \sin\phi_1\partial_{\phi_1} + \sin(\phi_2 + \tfrac{2\pi}{3})\partial_{\phi_2} + \sin(\phi_3 + \tfrac{4\pi}{3})\partial_{\phi_3}, \\ g_2 &= \sin\theta\partial_x + \cos\theta\partial_y - \cos\phi_1\partial_{\phi_1} - \cos(\phi_2 + \tfrac{2\pi}{3})\partial_{\phi_2} - \cos(\phi_3 + \tfrac{4\pi}{3})\partial_{\phi_3}, \\ g_3 &= \partial_\theta - (1 + \cos\phi_1)\partial_{\phi_1} - (1 + \cos\phi_2)\partial_{\phi_2} - (1 + \cos\phi_3)\partial_{\phi_3}. \end{aligned}$$

It is easy to check that in regular points these vector fields define a (bracket generating) distribution with growth vector $(3, 6)$. It means that in each regular point the vectors g_1, g_2, g_3 together with their Lie brackets span the whole tangent space. Consequently, the system is controllable by Chow–Rashevsky theorem. For instance, in the initial position $q = 0$ the robot is controllable since we compute

$$\operatorname{span}\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & -2 & 1 & 0 & -2 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & -2 & 1 & \sqrt{3} & 1 \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & -2 & 1 & -\sqrt{3} & 1 \end{pmatrix} \cong \mathbb{R}^6.$$

By [8], the singular points for the trident snake robot, i.e. where the robot is not controllable, correspond to such positions that either the legs are parallel or the wheel axles intersect in an instantaneous center of rotation. Obviously, the former singular positions coincide with the latter if the center of rotation is in infinity. An example of a typical singular position is shown in Fig. 3. We have not found any formula for singular points in CGA that would be as simple as in the case of the 3–link snake [11].

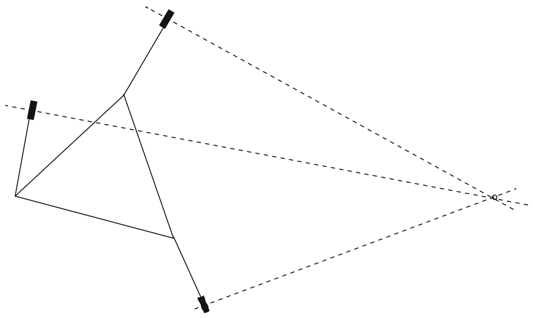


Fig. 3 A singular position of the robot

Let us remark at the end of this section that the results obtained above using CGA computation can directly be used to obtain the trident snake dynamics. Namely, the equations of motion are obtained as Euler–Lagrange equations, where the Lagrangian consists of the kinetic energy only since the robot is planar. Thus we only need to specify the translational and angular velocities. But the velocity of the body and angular velocities of body and wheels are obtained straight as derivatives of generalized coordinates and equation (2) gives the velocity of wheels. Note that although $\dot{p}_i$ is a vector in CGA, its square, which appears in the expression for the kinetic energy, is a number that gives the square of the real (physical) velocity.

4 CLUCalc Implementation

The proposed trident snake control was tested in CLUCalc software [1, 2], which is designed exactly for calculations in arbitrary predefined geometric algebra. The following code piece contains the definition of the initial position:

```
  // INITIAL POSITION
S0=VecN3(0,0,0);
LB0=VecN3(0,0,1);
R=RotorN3(0,0,1,2*Pi/3);
// Joints
pb10=VecN3(1,0,0);
pb20=R*pb10*~R;
pb30=R*pb20*~R;
// Axes
L10=TranslatorN3(pb10)*LB0*TranslatorN3(-pb10);
L20=R*L10*~R;
L30=R*L20*~R;
// Ends of legs
p10=VecN3(2,0,0);
p20=R*p10*~R;
p30=R*p20*~R;
```

The initial position is thus recalculated with respect to the controlling parameters change to get a current position. The code we demonstrate corresponds to the body and the first leg of the trident snake robot. The other legs are computed in the same way.

```
  T=TranslatorN3(x,y,0);
// BODY
// Center
S=T*S0*~T;
// Axis
LB=T*LB0*~T;
// Motor
MB=TranslatorN3(LB)*RotorN3(0,0,1,d)*~TranslatorN3(LB);

// FIRST LEG
// Joint
:Blue;
:pb1=MB*T*pb10*~T*~MB;
// Axis
L1=MB*T*L10*~T*~MB;
// Motor
M1=TranslatorN3(L1)*RotorN3(0,0,1,a)*TranslatorN3(-L1);
// End
:Black;
:p1=M1*MB*T*p10*~T*~MB*~M1;
```

The following three sets of pictures demonstrate the evolution from 0 in the direction of g_1, g_2 and g_3 vector fields.

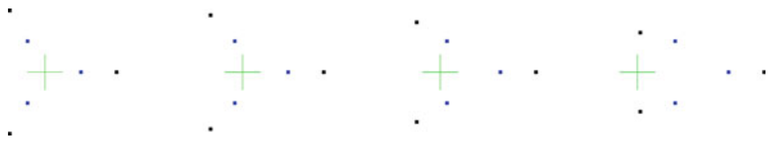


Fig. 4 g_1 direction (pictured by CLUCalc)

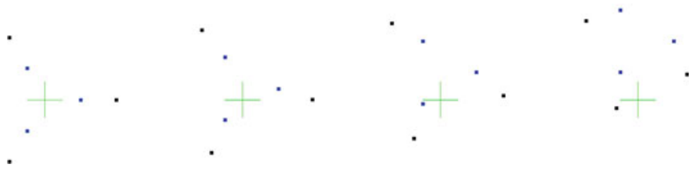


Fig. 5 g_2 direction (pictured by CLUCalc)

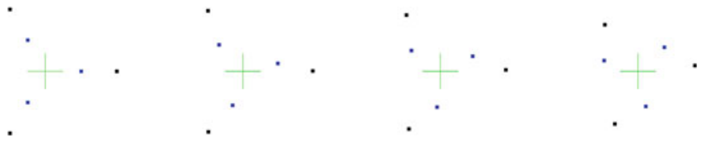


Fig. 6 g_3 direction (pictured by CLUCalc)

The last picture shows the motion corresponding to the bracket $[g_1, g_2]$ which is realized by means of a periodic transformation of the generators g_1 a g_2:

$$v(t) = -\epsilon\,\omega\sin(\omega t)g_1 + \epsilon\,\omega\cos(\omega t)g_2,$$

where $\epsilon = 0.3$, $\omega = 0.9$ and $t \in \langle 0, 2\pi/\omega \rangle$ (Figs. 4, 5, 6 and 7).

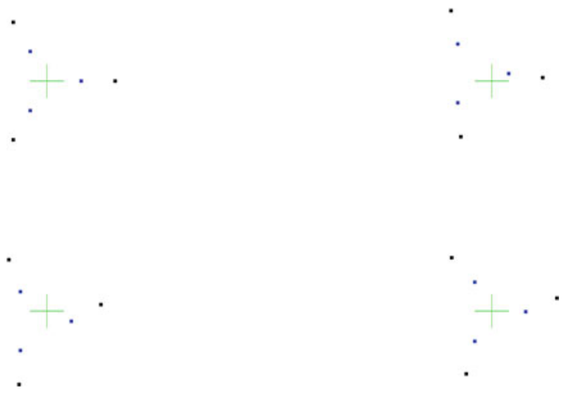


Fig. 7 $[g_1, g_2]$ direction (pictured by CLUCalc)

References

1. Hildenbrand, D.: Foundations of geometric algebra computing. Geometry and Computing, vol. 8. Springer, Berlin (2013)
2. Perwass, Ch.: Geometric algebra with applications in engineering. Geometry and Computing, vol. 4. Springer, Berlin (2009)
3. Zamora-Esquivel, J., Bayro-Corrochano, E.: Kinematics and diferential kinematics of binocular robot heads. In: Robotics and Automation, ICRA (2006)
4. Selig, J.M.: Geometric fundamentals of robotics. Monographs in Computer Science. Springer, Berlin (2004)
5. Murray, R.M., Zexiang, L., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Florida (1994)
6. Liljebäck, P., Pettersen, K.Y., Stavdahl, Ø., Gravdahl, J.T.: Snake Robots, Modelling, Mechatronics, and Control. Advances in Industrial Control. Springer, Berlin (2013)
7. Gonzlez-Jimnez, L., Carbajal-Espinosa, O., Loukianov, A., Bayro-Corrochano, E.: Robust pose control of robot manipulators using conformal geometric algebra. Adv. Appl. Clifford Algebras **24**(2), 533–552 (2014)
8. Ishikawa, M.: Trident snake robot: locomotion analysis and control, In: Proceedings IFAC NOLCOS, pp. 11691174. Stuttgart, Germany (2004)
9. Ishikawa, M., Minami, Y., Sugie, T.: Development and control experiment of the trident snake robot. In: Proceedings 45th IEEE CDC, pp. 64506455. San Diego, California (2006)
10. Ishikawa, M., Minami, Y., Sugie, T.: Development and control experiment of the trident snake robot. IEEE/ASME Trans. Mechatron. **15**, 915 (2010)
11. Hrdina, J., Návrat, A., Vašík, P.: Control of 3-link robotic snake based on Conformal geometric algebra preprint (2015)

Author Index

B
Brandejsky, Tomas, 53
Brest, Janez, 77
Burianek, Tomas, 63, 235

C
Canberk, Berk, 27

D
Dalecky, Stepan, 195
Das, Swagatam, 39, 103, 329
Datta, Shounak, 329
Davendra, Donald, 149

E
Etaner-Uyar, A. Sima, 27

F
Fister, Iztok, 77

G
Gazioglu, Emrullah, 27
George, Nithin V., 137
Ghosh, Arka, 103
Gopych, Ivan, 207
Gopych, Petro, 207

H
Hochreiter, Ronald, 181
Horak, Karel, 343
Howley, Enda, 3
Hrabec, Dušan, 17
Hrdina, Jaroslav, 363

J
Jafarian, Ahmad, 269
Jana, Nanda Dulal, 39
Janostik, Jakub, 149
Jariwala, Rushi, 137
Jarušek, Robert, 317

K
Klecka, Jan, 343
Kojecky, Lumir, 171
Kotyrba, Martin, 317

M
Mašek, Petr, 353
Mason, Karl, 3
Matoušek, Radomil, 375
Mazal, Jan, 17
Measoomy Nia, S., 269
Misak, Stanislav, 63, 235
Mullick, Sankha Subhra, 103

N
Návrat, Aleš, 375

O
Oplatkova, Zuzana Kominkova, 149, 291

P
Patidar, Rohan, 137
Paul, Sujoy, 329
Pluhacek, Michal, 127, 149
Poláková, Radka, 89
Popa, Călin-Adrian, 221, 245
Popela, Pavel, 17
Pospichal, Jiri, 281

R
Roupec, Jan, 17
Růžička, Michal, 353

S
Šeda, Miloš, 159
Šeda, Pavel, 159
Senkerik, Roman, 127, 149, 291
Sil, Jaya, 39
Singh, Elijah Blessing, 257
Škrabánek, Pavel, 305
Srinivasan, R., 257
Stodola, Petr, 17
Stuchly, Jindrich, 63, 235

T
Tepeh, Aleksandra, 77
Tvrdík, Josef, 89

V
Vantuch, Tomas, 63, 235
Vašík, Petr, 363
VinayKumar, Kotte, 257
Volna, Eva, 317

Z
Zboril, Frantisek V., 195
Zelinka, Ivan, 127, 149, 171