Advances in Intelligent Systems and Computing 576

Radek Matoušek *Editor*

Recent Advances in Soft Computing

Proceedings of the 22nd International Conference on Soft Computing (MENDEL 2016) held in Brno, Czech Republic, at June 8–10, 2016

Springer

Advances in Intelligent Systems and Computing

Volume 576

Series editor

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

About this Series

The series "Advances in Intelligent Systems and Computing" contains publications on theory, applications, and design methods of Intelligent Systems and Intelligent Computing. Virtually all disciplines such as engineering, natural sciences, computer and information science, ICT, economics, business, e-commerce, environment, healthcare, life science are covered. The list of topics spans all the areas of modern intelligent systems and computing.

The publications within "Advances in Intelligent Systems and Computing" are primarily textbooks and proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

More information about this series at http://www.springer.com/series/11156

Radek Matoušek
Editor

Recent Advances in Soft Computing

Proceedings of the 22nd International Conference on Soft Computing (MENDEL 2016) held in Brno, Czech Republic, at June 8–10, 2016

Springer

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

ISSN 2194-5357 ISSN 2194-5365 (electronic)
Advances in Intelligent Systems and Computing
ISBN 978-3-319-58087-6 ISBN 978-3-319-58088-3 (eBook)
DOI 10.1007/978-3-319-58088-3

Library of Congress Control Number: 2017939344

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

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 2016. 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.

The main aim of the Mendel conference was 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 a distinct way to address a certain problem in its specific domain. In 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.

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 chapters: Chapter 1: *Evolutionary Computing, Swarm intelligence, and Metaheuristics,* Chapter 2: *Neural Networks, Self-organization and Machine Learning*, and Chapter 3: *Intelligent Image Processing*.

We would like to thank the members of the International Program Committees and Reviewers for their hard work. We believe that Mendel conference represents a high standard conference in the domain of soft computing. Mendel 2016 enjoyed outstanding keynote lectures by distinguished guest speakers: René Lozi (France), Roman Senkerik (Czech Republic), Wolfram Wiesemann (UK), and Janez Brest (Slovenia).

Particular thanks go to the conference organizers and main sponsors as well. In 2016, the conference is organized under the auspices of the rector of Brno University of Technology with support of WU Vienna University of Economics and Business, and University of Vaasa. The conference sponsors are Humusoft Ltd. (international reseller and developer for MathWorks, Inc., USA.), B&R automation Ltd. (multi-national company, specialized 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 Lars Nolle for their hard and highly valuable work. Their work has definitely contributed to the success of the Mendel 2016 conference.

Radek Matoušek

Organization

International Program Committee and Reviewers Board

Riccardo Poli	University of Essex, UK
Roman Senkerik	Tomas Bata University in Zlín, Czech Republic
Ronald Hochreiter	WU Vienna, Austria
Salem Abdel-Badeh	Ain Shams University, Egypt
Tomoharu Nakashima	Osaka Prefecture University, Japan
Urszula Boryczka	University of Silesia, Poland
William Langdon	University College London, UK
Xin-She Yang	National Physical Laboratory, UK
Zuzana Oplatkova	Tomas Bata University in Zlín, Czech Republic

Executive Organizing Committee and Local Organizers

Chair and Volume Editor-in-Chief

Radek Matoušek, Czech Republic

Co-chairs and Associate Advisors

Jouni Lampinen, Finland
Ronald Hochreiter, Austria
Lars Nolle, Germany

Local Organizers

Peer Review Manager

Jiří Dvořák

Technical Review Manager

Ladislav Dobrovský

Senior Organizer

Jitka Pavlíková

Junior Organizers

Daniel Zuth
Petr Šoustek
Tomas Marada

Contents

Evolutionary Computing, Swarm Intelligence, Metaheuristics

A Multilevel Genetic Algorithm for the Maximum Constraint Satisfaction Problem

Noureddine Bouhmala(✉), Halvard Sanness Helgen, and Knut Morten Mathisen

Department of Maritime Technology and Innovation, University College of SouthEast, Postboks 235, 3603 Kongsberg, Norway
noureddine.bouhmala@usn.no

Abstract. The constraint satisfaction problem is a useful and well-studied framework for the modeling of many problems rising in Artificial Intelligence and other areas of Computer Science. As many real-world optimization problems become increasingly complex and hard to solve, better optimization algorithms are always needed. Genetic algorithms (GA) which belongs to the class of evolutionary algorithms is regarded as a highly successful algorithm when applied to a broad range of discrete as well continuous optimization problems. This paper introduces a hybrid approach combining a genetic algorithm with the multilevel paradigm for solving the maximum constraint satisfaction problem. The promising performances achieved by the proposed approach is demonstrated by comparisons made to solve conventional random benchmark problems.

Keywords: Genetic algorithms · Multilevel paradigm · Constraint satisfaction problem

1 Introduction

Many problems in the field of artificial intelligence can be modeled as constraint satisfaction problems (CSP). A constraint satisfaction problem (or CSP) is a tuple $\langle X, D, C\rangle$ where, $X = \{x_1, x_2, ...x_n\}$ is a finite set of variables, $D = \{D_{x_1}, D_{x_2}, ...D_{x_n}\}$ is a finite set of domains. Thus each variable $x \in X$ has a corresponding discrete domain D_x from which it can be instantiated, and $C = \{C_{i1}, C_{i2}, ...C_{ik}\}$ is a finite set of constraints. Each k-ary constraint restricts a k-tuple of variables, $(x_{i1}, x_{i2}, ...x_{ik})$ and specifies a subset of $D_{i1} \times ... \times D_{ik}$, each element of which are values that the variables can not take simultaneously. A solution to a CSP requires the assignment of values to each of the variables from their domains such that all the constraints on the variables are satisfied. The maximum constraint satisfaction problem (Max-CSP) aims at finding an assignment so as to maximize the number of satisfied constraints. Max-CSP can be regarded as the generalization of CSP; the solution maximizes the number of satisfied constraints. In this paper, attention is focused on binary CSPs, where all constraints are binary, i.e., they are based on the cartesian product of the domains of two variables. However, any non-binary CSP can theoretically be converted to a binary CSP [6]. Algorithms for solving CSPs apply the so-called 1-exchange neighborhood under which two solutions are direct neighbors if, and only if, they differ at most in the value assigned to one variable. Examples include the minimum conflict heuristic MCH [14], the break method for escaping from local minima [13], various enhanced MCH (e.g., randomized iterative improvement of MCH called WMCH [18], MCH with tabu search [17]), evolutionary algorithms [1,2,4,20]. Weights-based algorithms are techniques that work by introducing weights on variables or constraints in order to avoid local minima. Methods belonging to this category include genet [5], the exponentiated sub-gradient [16], the scaling and probabilistic smoothing [10], evolutionary algorithms combined with stepwise adaptation of weights [11], methods based on dynamically adapting weights on variables [15], or both (i.e., variables and constraints) [8]. Other methods rely on a larger neighborhood [3,12] where a sequence of moves is to be performed and the algorithm will select the move that returns the best solution.

2 Multilevel Genetic Algorithm

Genetic Algorithms (GAs) [9] are stochastic methods for global search and optimization and belong to the group of nature inspired meta-heuristics leading to the so-called natural computing. GAs operate on a population of artificial chromosomes. Each chromosome can be thought of as a point in the search space of candidate solution. Each individual is assigned a score (fitness) value that allows assessing its quality. Starting with a randomly generated population of chromosomes, GA carries out a process of fitness-based selection and recombination to produce a successor population, the next generation. During recombination, parent chromosomes are selected and their genetic material is recombined to produce child chromosomes. As this process is iterated, individuals from the set of solutions which is called population will evolve from generation to generation by repeated applications of an evaluation procedure that is based on genetic operators. Over many generations, the population becomes increasingly uniform until it ultimately converges to optimal or near-optimal solutions. Figure 1 shows graphically the multilevel paradigm using 6 variables. The four phases of the multilevel paradigm are coarsening, initial assignment, uncoarsening and refinement. The multilevel genetic algorithm works as follows:

– **Construction of levels (coarsening):** Let $G_0 = (V_0, E_0)$ be an undirected graph of vertices V and edges E. The set V denotes variables and each edge $(x_i, x_j) \in E$ implies a constraint joining the variables x_i and x_j. Given the initial graph G_0, the graph is repeatedly transformed into smaller and smaller graphs $G_1, G_2, ..., G_m$ such that $|V_0| > |V_1| > ... > |V_m|$. To coarsen a graph

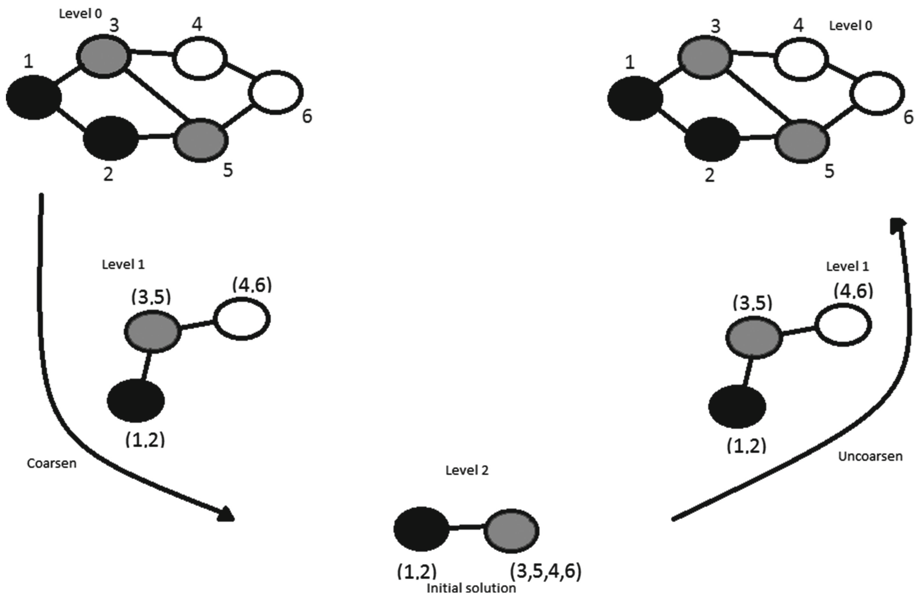


Fig. 1. The various phases of the multilevel Paradigm.

from G_j to G_{j+1}, a number of different techniques may be used. In this paper, when combining a set of variables into clusters, the variables are visited in a random order. If a variable x_i has not been matched yet, then the algorithm randomly selects one of its neighboring unmatched variable x_j, and a new cluster consisting of these two variables is created. Its neighbors are the combined neighbors of the merged variables x_i and x_j.

- **Initial assignment:** The process of constructing a hierarchy of graphs ceases as soon as the size of the coarsest graph reaches some desired threshold. A random initial population is generated at the lowest level $(G_k = (V_k, E_k)$. The chromosomes which are assignments of values to the variables are encoded as strings of bits, the length of which is the number of variables. At the lowest level, the length of the chromosome is equal to the number of clusters. The initial solution is simply constructed by assigning to each variable x_i in a cluster, a random value v_i from D_{x_i} In this work it is assumed that all variables have the same domain (i.e., same set of values), otherwise different random values should be assigned to each variable in the cluster. All the individuals of the initial population are evaluated and assigned a fitness expressed in Eq. 1 which counts the sum of weights of constraint violations where $<(x_i, s_i), (x_j, s_j)>$ denotes the constraint between the variables x_i and x_j where x_i is assigned the value s_i from D_{x_i} and x_j is assigned the value s_j from D_{x_j}, and $W_{i,j}$ is the weight between variable x_i and x_j.

$$Fitness = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Violation(W_{i,j}<(x_i, s_i), (x_j, s_j)> \qquad (1)$$

- **Initial weights:** The next step of the algorithm assigns a fixed amount of weight equal to 1 across all the constraints. The distribution of weights to constraints aims at forcing hard constraints with large weights to be satisfied thereby preventing the algorithm at a later stage from getting stuck at a local optimum.
- **Parent selection:** During the refinement phase, new solutions are created by combining pairs of individuals in the population and then applying a crossover operator to each chosen pair. Combining pairs of individuals can be viewed as a matching process. In the version of GA used in this work, the individuals are visited in random order. An unmatched individual I_k is matched randomly with an unmatched individual I_l.
- **Genetic operators:** The task of the crossover operator is to reach regions of the search space with higher average quality. The two-point crossover operator is applied to each matched pair of individuals. The two-point crossover selects two randomly points within a chromosome and then interchanges the two parent chromosomes between these points to generate two new offspring. The mutation operator is then applied in order to introduce new features in the population. By mutation, the alleles of the produced child have a chance to be modified, which enables further exploration of the search space. The mutation operator takes a single parameter p_m, which specifies the probability of performing a possible mutation. Let $C = vl_1, vl_2, ...vl_m$ be a chromosome represented by a chain where each of whose gene vl_i is a value from the feasible domain (i.e., all the variable have the same domain). In our mutation operator, a chosen gene vl_i is mutated through modifying this gene's allele current value to a new random value from the feasible domain, if the probability test is passed. For coarser graphs ($G_k = (V_k, E_k)$) when ($k > 1$), the mutation is applied to a cluster of variables (i.e., all the variables within the chosen cluster undergo the same mutation).
- **Survivor selection:** The selection acts on individuals in the current population. Based on each individual quality (fitness), it determines the next population. In the roulette method, the selection is stochastic and biased toward the best individuals. The first step is to calculate the cumulative fitness of the whole population through the sum of the fitness of all individuals. After that, the probability of selection is calculated for each individual as being $P_{Selection_{I_k}} = f_{I_k} / \sum_1^N f_{I_k}$, where f_{I_k} is the fitness of individual I_k.
- **Updating weights:** The weights of each current violated constraint is then increased by one, whereas the newly satisfied constraints will have their weights decreased by one before the start of new generation.
- **Termination condition:** The convergence of GA is supposed to be reached if the best individual remains uncharged during 5 consecutive generations.
- **Uncoarsening:** Once GA has reached the convergence criterion with respect to a child level graph $G_k = (V_k, E_k)$, the assignment reached on that level must be projected on its parent graph $G_{k-1} = (V_{k-1}, E_{k-1})$. The projection algorithm is simple; if a cluster belongs to $G_k = (V_k, E_k)$ is assigned the value vl_i, the merged pair of clusters that it represents belonging to $G_{k-1} = (V_{k-1}, E_{k-1})$ are also assigned the value vl_i,

– **Refinement:** Having computed an initial solution at the coarsest graph, GA starts the search process from the coarsest level $G_k = (V_k, E_k)$ and continues to move towards smaller levels. The motivation behind this strategy is that the order in which the levels are traversed offers a better mechanism for performing diversification and intensification. The coarsest level allows GA to view any cluster of variables as a single entity leading the search to become guided in far away regions of the solution space and restricted to only those configurations in the solution space in which the variables grouped within a cluster are assigned the same value. As the switch from one level to another implies a decrease in the size of the neighborhood, the search is intensified around solutions from previous levels in order to reach better ones.

3 Experimental Results

3.1 Experimental Setup

The benchmark instances were generated using model A [19] as follows: Each instance is defined by the 4-tuple $\langle n, m, p_d, p_t \rangle$, where n is the number of variables; m is the size of each variable's domain; p_d, the constraint density, is the proportion of pairs of variables which have a constraint between them; and p_t,

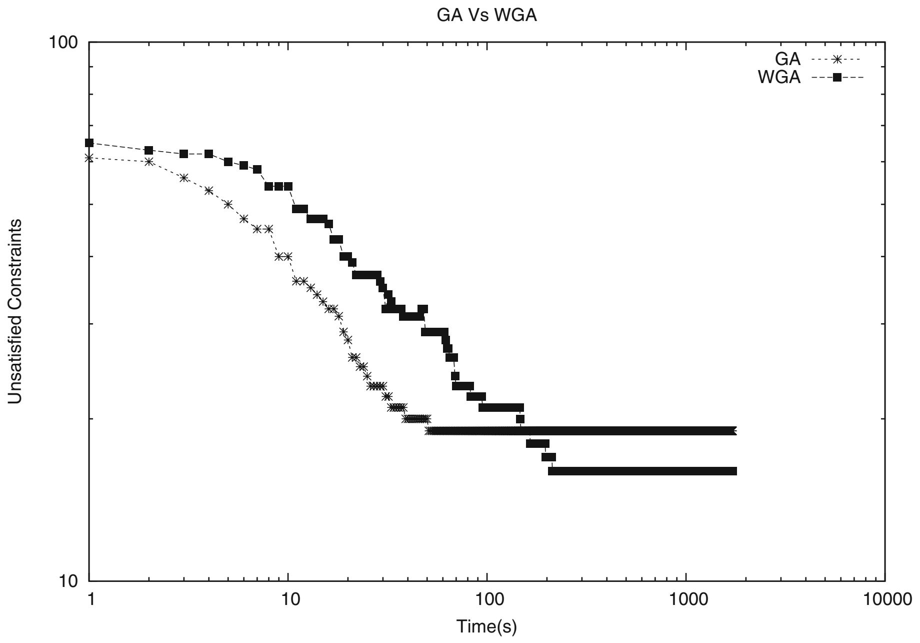


Fig. 2. GA vs WGA: Evolution of the mean unsatisfied constraints as a function of time for csp-N30-DS40-C125-cd026ct063.

the constraint tightness, is the probability that a pair of values is inconsistent. From the $(n \times (n-1)/2)$ possible constraints, each one is independently chosen to be added in the constraint graph with the probability p_d. Given a constraint, we select with the probability p_t which value pairs become no-goods. The model A will on average have $p_d \times (n-1)/2$ constraints, each of which having on average $p_t \times m^2$ inconsistent pairs of values. For each pair of density tightness, we generate one soluble instance (i.e., at least one solution exists). Because of the stochastic nature of GA, we let each algorithm do 100 independent runs, each run with a different random seed. Many NP-complete or NP-hard problems show a phase transition point that marks the spot where we go from problems that are under-constrained and so relatively easy to solve, to problems that are over-constrained and so relatively easy to prove insoluble. Problems that are on average harder to solve occur between these two types of relatively easy problem. The values of p_d and p_t are chosen in such a way that the instances generated are within the phase transition. In order to predict the phase transition region, a formula for the constrainedness [7] of binary CSPs was defined by:

$$\kappa = \frac{n-1}{2} p_d log_m \left(\frac{1}{1-pt}\right). \tag{2}$$

The tests were carried out on a DELL machine with 800 MHz CPU and 2 GB of memory. The code was written in C and compiled with the GNU C

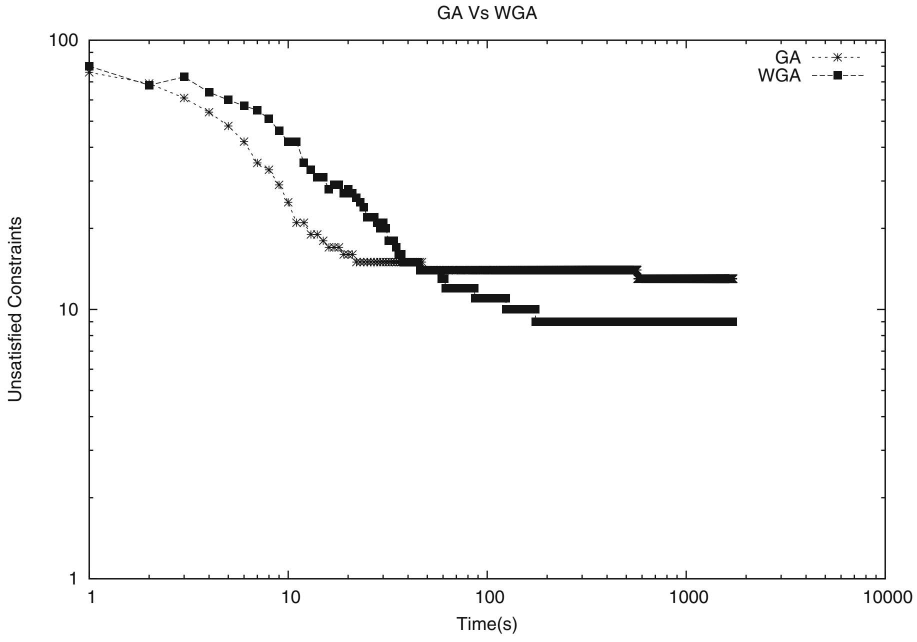


Fig. 3. GA vs WGA: Evolution of the mean unsatisfied constraints as a function of time. csp-N35-DS20-C562-cd094-ct017

compiler version 4.6. The following parameters have been fixed experimentally and are listed below:

- Crossover probability = 0.9
- Mutation probability = 0.1
- Population size = 50
- Stopping criteria for the coarsening phase: The reduction process stops as soon as the number of levels reaches 3. At this level, MLV-WGA generates an initial population.
- Convergence during the optimization phase: If there is no observable improvement of the fitness function of the best individual during 5 consecutive generations, MLV-WGA is assumed to have reached convergence and moves to a higher level.

3.2 Results

The plots in Figs. 2 and 3 compare the performance of GA with and without assignment of weights to constraints. The plots show the evolution of the mean unsatisfied number of constraints as a function of time. The difference of performances between GA and WGA (GA with weight) is most pronounced during the first stage of the search with the GA curve that lies below that of WGA.

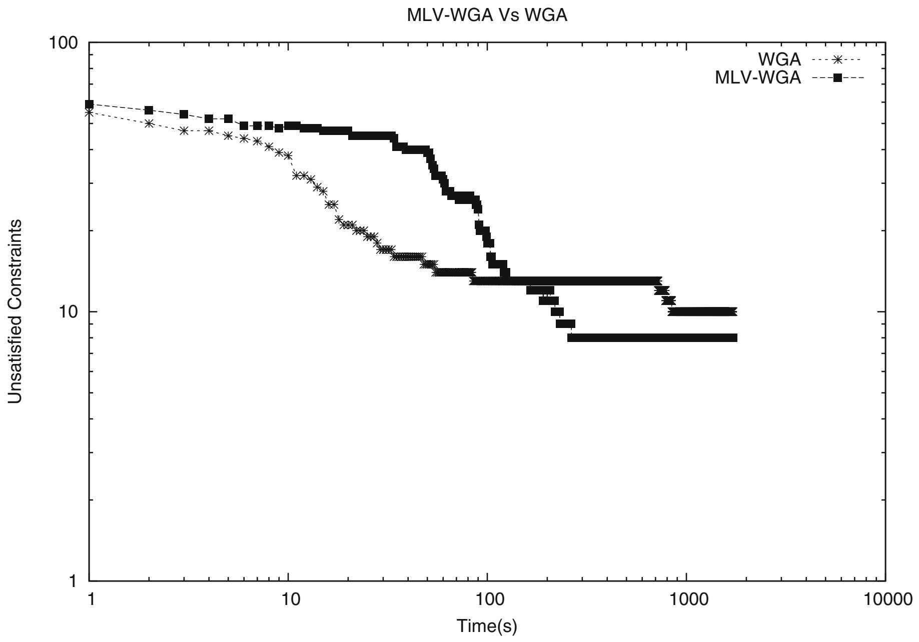


Fig. 4. MLV-WGA vs WGA: Evolution of the mean unsatisfied constraints as a function of time for csp-N25-40.

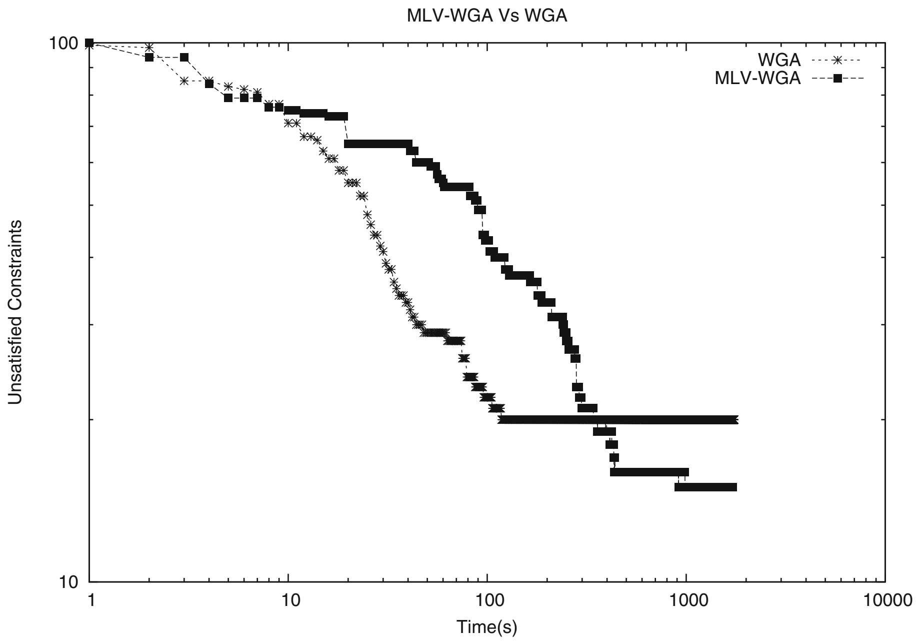


Fig. 5. MLV-WGA vs WGA: Evolution of the mean unsatisfied constraints as a function of time for csp-N35-40.

As the time increases, GA reaches a premature convergence while WGA continues to improve returning a solution of better quality than GA. The impact of introducing weights to constraints enables each constraint relating at most two variables to be assigned an ideal weight expressing its relative hardness taking into account the values assigned to its relating variables and the values of the variables defining the neighboring constraints. This ideal weight leads GA to enter a state of balance that is required for GA to satisfy hard constraints as early as possible before being able to reach solutions of better quality. The plots in Figs. 4 and 5 compare WGA with its multilevel variant MLV-WGA. The improvement in quality imparted by the multilevel context is immediately clear. Both WGA and MLV-WGA exhibits what is called a plateau region. A plateau region spans a region in the search space where cross-over and mutation operators leave the best solution or the mean solution unchanged. However, the length of this region is shorter with MLV-WGA compared to that of WGA. The multilevel context uses the projected solution obtained at $G_{m+1}(V_{m+1}, E_{m+1})$ as the initial solution for $G_m(V_m, E_m)$ for further refinement. Even though the solution at $G_{m+1}(V_{m+1}, E_{m+1})$ is at a local minimum, the projected solution may not be at a local optimum with respect to $G_m(V_m, E_m)$. The projected assignment is already a good solution leading WGA to converge quicker within few generations to a better solution. Tables 1, 2, 3 and 4 show a comparison of the two algorithms. For each algorithm, the best (Min) and the worst (Max) results are given, while

Table 1. MLV-WGA vs WGA- number of variables: 25

Instance	MLV-WGA				WGA			
	Min	Max	Mean	RE_{av}	Min	Max	Mean	RE_{av}
N25-DS20-C36-cd-014-ct083	3	7	4.58	**0.128**	3	8	5.41	0.151
N25-DS20-C44-cd012-ct087	6	10	8.04	**0.183**	8	14	9.91	0.226
N25-DS20-C54-cd018-ct075	3	7	5.37	**0.100**	4	9	6.91	0.128
N25-DS20-C78-cd026-ct061	2	8	4.33	**0.056**	2	10	5.79	0.073
N25-DS20-C225-cd078-ct027	3	8	4.16	**0.019**	3	9	5.66	0.026
N25-DS20-C229-cd072-ct029	4	9	6.04	**0.014**	4	11	8.16	0.036
N25-DS20-C242-cd086-ct025	1	6	3.5	**0.015**	3	10	5.70	0.024
N25-DS20-C269-cd086-ct025	4	10	5.66	**0.022**	4	10	7.54	0.029
N25-DS20-C279-cd094-ct023	2	7	4.75	**0.018**	4	9	6.75	0.025
N25-DS40-C53-cd016-ct085	6	11	8.91	**0.169**	8	13	10.70	0.202
N25-DS40-C70-cd026-ct069	2	6	4.25	**0.061**	3	8	5.75	0.083
N25-DS40-C72-cd022-ct075	6	12	9	**0.125**	6	15	10.45	0.146
N25-DS40-C102-cd032-ct061	5	12	8.12	**0.080**	7	14	10.33	0.102
N25-DS40-C103-cd034-ct059	5	9	6.83	**0.067**	4	12	8.79	0.086
N25-DS40-C237-cd082-ct031	3	8	5.66	**0.024**	5	10	7.87	0.034
N25-DS40-C253-cd088-ct029	3	7	4.95	**0.020**	5	12	8.04	0.032
N25-DS40-C264-cd088-ct029	5	10	6.91	**0.027**	6	16	8.91	0.034
N25-DS40-C281-cd096-ct027	3	9	5.62	**0.020**	4	12	8.54	0.031
N25-DS40-C290-cd096-ct027	4	10	7.08	**0.025**	6	14	9	0.032

mean represents the average solution and RE_{av} represents the average relative error $(1-(n_c-n_s)/n_c$, where n_c is the number of constraints, n_c is the number of satisfied constraints). The algorithm with the lowest average relative error is indicated in bold. The MLV-WGA outperforms WGA in 53 cases out of 96, gives similar results in 20 cases, and was beaten in 23 cases. The performance of both algorithms differ significantly. The difference for the total performance is between 25% and 70% in the advantage of MLV-WGA. Comparing the worst performances of both algorithms, MLV-WGA produces bad results in 15 cases, both algorithms give similar results in 8 cases, and MLV-WGA was able to perform better than WGA in 73 cases. Looking at the average results, MLV-WGA does between 16% and 41% better than WGA in 84 cases, while the differences are very marginal in the remaining cases where WGA beats MLV-WGA. Looking at the average relative error for both algorithms, MLV-WGA gives a relative error ranging between 1% and 18% compared to 2% and 22% for N25, between 2% and 9% compared to 2% and 15% for N30, between 1% and 12% compared to 1% and 15% for N40, and finally between 1% and 14% compared to 1% and 15% for N50.

Table 2. MLV-WGA vs WGA: number of variables: 30

Instance	MLV-WGA				WGA			
	Min	Max	Mean	RE_{av}	Min	Max	Mean	RE_{av}
N30-DS20-C41-cd012-ct083	2	6	3.70	**0.026**	3	7	5.08	0.124
N30-DS20-C71-cd018-ct069	1	7	3.37	**0.048**	3	10	5.66	0.080
N30-DS20-C85-cd020-ct065	3	9	6	**0.071**	5	12	8.37	0.099
N30-DS20-C119-cd028-ct053	3	10	5.70	**0.048**	6	12	8.83	0.075
N30-DS20-C334-cd074-ct025	6	13	8.16	**0.025**	6	14	9.87	0.030
N30-DS20-C387-cd090-ct021	3	9	6.66	**0.018**	5	13	8.70	0.033
N30-DS20-C389-cd090-ct021	2	9	6.08	**0.016**	4	14	8.95	0.024
N30-DS20-C392-cd090-ct021	3	10	7.08	**0.019**	5	15	9.16	0.024
N30-DS20-C399-cd090-ct021	5	13	7.70	**0.020**	6	14	9.79	0.025
N30-DS40-C85-cd020-ct073	5	11	7.75	**0.092**	7	14	10.87	0.152
N30-DS40-C96-cd020-ct073	8	12	16	0.167	11	19	14.58	0.015
N30-DS40-C121-cd026-ct063	8	14	10.5	**0.087**	9	19	14.33	0.152
N30-DS40-C125-cd026-ct063	8	18	12.20	**0.098**	10	19	15.58	0.125
N30-DS40-C173-cd044-ct045	4	10	6.41	**0.038**	6	14	9.20	0.054
N30-DS40-C312-cd070-ct031	7	14	10.5	0.033	7	19	13.33	0.025
N30-DS40-C328-cd076-ct029	6	13	10.37	**0.032**	10	18	13.45	0.042
N30-DS40-C333-cd076-ct029	7	13	10.25	**0.031**	9	18	12.62	0.038
N30-DS40-C389-cd090-ct025	6	13	9.33	**0.024**	9	17	12.20	0.032
N30-DS40-C390-cd090-ct025	6	14	9.29	**0.024**	10	17	13	0.031

Table 3. MLV-WGA vs WGA- number of variables 40.

Instance	MLV-WGA				WGA			
	Min	Max	Mean	RE_{av}	Min	Max	Mean	RE_{av}
N40-DS20-C78-cd010-ct079	6	12	8.91	**0.115**	5	13	9.04	0.116
N40-DS20-C80-cd010-ct079	7	13	9.62	**0.121**	7	13	10.04	0.153
N40-DS20-C82-cd012-ct073	4	9	6.25	**0.073**	4	11	6.95	0.085
N40-DS20-C95-cd014-ct067	2	8	4.45	0.047	2	7	4.12	**0.044**
N40-DS20-C653-cd084-ct017	2	14	9.37	**0.015**	6	16	10.62	0.018
N40-DS20-C660-cd084-ct017	6	14	9.12	**0.014**	7	6	9.75	0.015
N40-DS20-C751-cd096-ct015	6	13	9.91	0.014	5	13	9.83	0.014
N40-DS20-C752-cd096-ct015	5	17	9.29	0.013	3	13	9.20	0.013
N40-DS20-C756-cd096-ct015	6	15	9.95	0.014	5	16	8.75	**0.012**
N40-DS40-C106-cd014-ct075	7	14	11.08	**0.105**	7	16	11.5	0.109
N40-DS40-C115-cd014-ct075	12	20	15.5	0.135	11	20	15.5	0.135

(continued)

Table 3. *(continued)*

Instance	MLV-WGA				WGA			
	Min	Max	Mean	RE_{av}	Min	Max	Mean	RE_{av}
N40-DS40-C181-cd024-ct055	6	17	12.04	0.067	7	17	11.75	**0.065**
N40-DS40-C196-cd024-ct055	11	12	16.58	0.085	12	20	15.54	**0.080**
N40-DS40-C226-cd030-ct047	7	14	10.91	0.051	7	16	11.16	**0.050**
N40-DS40-C647-cd082-ct021	11	23	15.66	0.025	11	20	15.20	**0.024**
N40-DS40-C658-cd082-ct021	11	22	16.33	**0.025**	13	21	16.70	0.026
N40-DS40-C703-cd092-ct019	9	21	13.41	0.020	8	20	13.58	0.020
N40-DS40-C711-cd092-ct019	12	23	15.75	0.023	8	20	14.87	0.021
N40-DS40-C719-cd092-ct019	8	21	16.54	0.024	10	20	15.16	**0.022**

Table 4. MLV-WGA vs WGA-number of variables 50.

Instance	MLV-WGA				WGA			
	Min	Max	Mean	RE_{av}	Min	Max	Mean	RE_{av}
N50-DS20-C130-cd010-ct071	10	17	14.16	**0.109**	9	19	14.79	0.114
N50-DS20-C146-cd010-ct071	17	27	21.41	**0.147**	18	27	21.87	0.150
N50-DS20-C250-cd022-ct043	5	13	8.87	0.036	5	13	8.25	**0.033**
N50-DS20-C296-cd022-ct043	12	21	16.87	**0.057**	11	23	17.66	0.060
N50-DS20-C365-cd030-ct034	8	18	12	0.033	6	16	11.29	**0.031**
N50-DS20-C1067-cd088-ct013	5	16	10.16	**0.010**	5	17	11.25	0.011
N50-DS20-C1071-cd089-ct013	9	18	11.16	0.016	7	19	11.08	**0.011**
N50-DS20-C1077-cd088-ct013	6	16	11.41	0.011	7	17	11.83	0.011
N50-DS20-C1176-cd096-ct012	7	16	11.62	**0.010**	8	17	12.20	0.011
N50-DS20-C1181-cd097-ct012	10	18	12.87	0.011	9	16	12.29	0.011
N50-DS40-c181-cd016-ct061	9	21	12.91	0.072	5	20	12.66	**0.070**
N50-DS40-c189-cd016-ct061	12	24	16.45	0.088	9	22	14.70	**0.078**
N50-DS40-c217-cd018-ct057	12	24	17.45	0.081	12	22	16.66	**0.077**
N50-DS40-c218-cd018-ct057	13	25	18.37	0.085	11	27	18.16	**0.084**
N50-DS40-c1084-cd088-ct016	16	31	24.62	0.023	14	31	23.12	**0.022**
N50-DS40-c1141-cd094-ct015	15	27	21.66	**0.019**	15	28	22.04	0.020
N50-DS40-c1148-cd093-ct015	15	29	21.08	0.019	13	29	21.25	0.019
N50-DS40-c1152-cd094-ct015	18	32	23.25	0.021	18	30	23.58	0.021
N50-DS40-c1155-cd094-ct015	14	30	21.54	0.021	13	26	20.70	0.021

4 Conclusion

In this work, a multilevel weighted based-genetic algorithm is introduced for MAX-CSP. The weighting strategy proved to be an excellent mechanism to enhance GA in order to achieve a suitable trade-off between intensification and diversification. The results have shown that the multilevel paradigm improves WGA and returns a better solution for the equivalent run-time for most cases. The multilevel paradigm offers a better mechanism for performing diversification in- intensification. This I achieved by allowing GA to view a cluster of variables as a single entity thereby leading the search becoming guided and restricted to only those assignments in the solution space in which the variables grouped within a cluster are assigned the same value. As the size of the clusters gets smaller from one level to another, the size of the neighborhood becomes adaptive and allows the possibility of exploring different regions in the search space while intensifying the search by exploiting the solutions from previous levels in order to reach better solutions.

References

1. Bacanin, N., Tuba, M.: Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators. Stud. Inf. Control **21**(2), 137–146 (2012)
2. Bonyadi, M., Li, X., Michalewicz, Z.: A hybrid particle swarm with velocity mutation for constraint optimization problems. In: Genetic and Evolutionary Computation Conference, pp. 1–8. ACM, New York (2013). ISBN 978-1-4503-1963-8
3. Bouhmala, N.: A variable depth search algorithm for binary constraint satisfaction problems. Math. Prob. Eng. **2015**, 10 (2015). doi:10.1155/2015/637809. Article ID 637809
4. Curran, D., Freuder, E., Jansen., T.: Incremental evolution of local search heuristics. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO 2010, pp. 981–982. ACM, New York (2010)
5. Davenport, A., Tsang, E., Wang, C.J., Zhu, K.: Genet: a connectionist architecture for solving constraint satisfaction problems by iterative improvement. In: Proceedings of the Twelfth National Conference on Artificial Intelligence (1994)
6. Dechter, R., Pearl, J.: Tree clustering for constraint networks. Artif. Intell. **38**, 353–366 (1989)
7. Gent, I.P., MacIntyre, E., Prosser, P., Walsh, T.: The constrainedness of search. In: Proceedings of the AAAI 1996, pp. 246–252 (1996)
8. Fang, S., Chu, Y., Qiao, K., Feng, X., Xu, K.: Combining edge weight and vertex weight for minimum vertex cover problem. FAW **2014**, 71–81 (2014)
9. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
10. Hutter, F., Tompkins, D.A.D., Hoos, H.H.: Scaling and probabilistic smoothing: efficient dynamic local search for SAT. In: Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 233–248. Springer, Heidelberg (2002). doi:10.1007/3-540-46135-3_16
11. Karim, M.R.: A new approach to constraint weight learning for variable ordering in CSPs. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2014), Beijing, China, pp. 2716–2723 (2014)

12. Lee, H.-J., Cha, S.-J., Yu, Y.-H., Jo, G.-S.: Large neighborhood search using constraint satisfaction techniques in vehicle routing problem. In: Gao, Y., Japkowicz, N. (eds.) AI 2009. LNCS (LNAI), vol. 5549, pp. 229–232. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01818-3_30
13. Morris, P.: The breakout method for escaping from local minima. In: Proceeding AAAI 1993 Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 40–45 (1993)
14. Minton, S., Johnson, M., Philips, A., Laird, P.: Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling scheduling problems. Artif. Intell. **58**, 161–205 (1992)
15. Pullan, W., Mascia, F., Brunato, M.: Cooperating local search for the maximum clique problems. J. Heuristics **17**, 181–199 (2011)
16. Schuurmans, D., Southey, F., Holte, R.: The exponentiated subgradient algorithm for heuristic Boolean programming. In: 17th International Joint Conference on Artificial Intelligence, pp. 334–341. Morgan Kaufmann Publishers, San Francisco (2001)
17. Stützle., T.: Local search algorithms for combinatorial problems - analysis, improvements, and new applications. Ph.D. thesis, TU Darmstadt, FB Informatics, Darmstadt, Germany (1998)
18. Wallace, R.J., Freuder, E.C.: Heuristic methods for over-constrained constraint satisfaction problems. In: Jampel, M., Freuder, E., Maher, M. (eds.) OCS 1995. LNCS, vol. 1106, pp. 207–216. Springer, Heidelberg (1996). doi:10.1007/3-540-61479-6_23
19. Xu, W.: Satisfiability transition and experiments on a random constraint satisfaction problem model. Int. J. Hybrid Inf. Technol. **7**(2), 191–202 (2014)
20. Zhou, Y., Zhou, G., Zhang, J.: A hybrid glowworm swarm optimization algorithm for constrained engineering design problems. Appl. Math. Inf. Sci. **7**(1), 379–388 (2013)

Models and Simulations of Queueing Systems

Miloš Šeda[1(✉)], Jindřiška Šedová[2], and Miroslav Horký[1]

[1] Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, 616 69 Brno, Czech Republic
seda@fme.vutbr.cz

[2] Faculty of Economics and Administration, Masaryk University, Lipová 41a, 602 00 Brno, Czech Republic
jsedova@econ.muni.cz

Abstract. In the queueing theory, it is assumed that requirement arrivals correspond to the Poisson process and the service time has the exponential distribution. Using these assumptions, the behaviour of the queueing system can be described by means of the Markov chains and it is possible to derive characteristics of the system. In the paper, these theoretical approaches are presented and focused on systems with several service lines and the FIFO queue when the number of requirements exceeds the number of lines. Finally, it is also shown how to compute the characteristics in a situation when these assumptions are not satisfied.

Keywords: Queue · Markovian chain

1 Introduction

The fundamentals of the queueing theory were laid by the Danish mathematician A. K. Erlang who worked for a telecommunication company in Copenhagen and in 1909 described an application of the probabilistic theory in telephone practice. The further development of the theory is mainly associated with the Russian mathematician A. N. Kolmogorov. The classification of queueing systems, as we use it today, was introduced in the 1950's by the English mathematician D.G. Kendall. Today, the queueing theory belongs to the classic part of logistics, and it is described in several monographs such as [1–4, 6, 8, 9].

Generally, at random moments, customers (requirements) enter the system and require servicing. Service options may be limited, e.g., the number of service lines (or channel operator). If at least one serving line is empty, the demand entering the system is immediately processed. However, the service time is also random in nature because the performance requirements may vary. If all service lines are busy, then the requirements (customers) must wait for their turn in a queue for the processing of previous requirements. However, all requirements are not always handled or queued for later use. For example, a telephone call is not connected because the phone number is busy.

Service lines are frequently arranged in parallel, e.g., at the hairdresser's where customers waiting for a haircut are served by several stylists, or at a gas station, where motorists call at several stands of fuel. However, there is a serial configuration of the queue system.

R. Matoušek (ed.), *Recent Advances in Soft Computing*, Advances in Intelligent Systems and Computing 576, DOI 10.1007/978-3-319-58088-3_2

2 Classification of Queueing Systems

The queue is usually understood in the usual **FIFO** sense – (*first in*, *first out*), but a **LIFO** operation (*last in*, *first out*) is also possible, which is also referred to as a **LCFS** (*last come*, *first served*) strategy.

Besides the FIFO and LIFO service, we can also meet random selection of requirements from the queue to the service system (**SIRO** - selection in random order) and service managed by priority requirements (**PRI** - Priority).

The *queue length* may be *limited* by rejecting additional requirements if a certain (predefined) number of requirements is achieved, such as the number of reservations for the book in a library that is currently checked out or (virtually) *unlimited*.

The requirements in the queue may have *limited* or *unlimited patience*. In the case of unlimited patience, requirements wait for their turn while in, a system with limited patience entering the queue significantly depends on the queue length. Instead of the queue length the concept of system *capacity* may also be used, which means the maximum number of requirements that may be present in the system.

In 1951, Kendall proposed a classification based on three main aspects in the form A/B/C, where

A characterises the probability distribution of random variable period (interval) between the requirement arrivals to the system,

B characterises the probability distribution of random variable *service time of a requirement*, and

C is the number of parallel service lines (or channel number), in the case of "unrestricted" (i.e. very large) number of lines is usual to express the parameter C by ∞.

As already mentioned, the system can be characterised by a larger number of features, so Kendall classification was further extended to the form A/B/C/D/E/F, where the meanings of the symbols D, E and F are as follows:

D integer indicating the maximum number of requirements in the system (i.e. the capacity of the system), unless explicitly restricted, expressed by ∞,

E integer expressing the maximum number of requirements in the input stream (or in a resource requirements), if it is unlimited, ∞ is used,

F queue type (FIFO/LIFO/SIRO/PRI).

Parameter A can have the following values:

M intervals between the arrivals of requirements are mutually stochastically independent and have exponential distribution, this means that the input stream represents a Poisson (Markov) process, for details, see below,

E_k Erlang distribution with parameters λ and k,

K_n χ^2 distribution with n degrees of freedom,

N normal (Gaussian) distribution,

U uniform distribution,

G	general case, the time between the arrivals of requirements is given by its distribution function,
D	intervals between the arrivals of demands are constant (they are deterministic in nature).

Parameter B can have the same value as parameter A, but these values here refer to a requirement-service-time random variable.

Since most of the queueing systems assume that the input current requirements are a *Poisson* (*Markov*) process, we will further describe it. A Poisson process is a stream of events that satisfies the following properties:

1. *Stationarity* (*homogeneity over time*) - the number of events in equally long time intervals is constant.
2. *Regularity* - the probability of more than one event at a sufficiently small interval of length Δt is negligibly small. This means that in, the interval $(t, t + \Delta t)$, there is either exactly one event with probability $\lambda \, \Delta t$ or no event with probability $1 - \lambda \, \Delta t$. In other words, in a Poisson process, the only system transition to the next "higher" state is possible or the system remains in the same condition.
3. *Independence of increases* - the number of events that occur in one time interval does not depend on the number of events in other intervals.

3 The M/M/n/nN/∞/FIFO System

In [7], we studied the M/M/1/∞/∞/FIFO system, here we focus on M/M/*n*/*n*/∞/FIFO system, which is more frequent in practice.

To derive the characteristics of the system, it is convenient to describe the system activity by a *graph of system transitions*. The nodes of the graph represent the states and the directed edges transitions from one state to another. The evaluation of these edges is described by the probability of transition from one state to another. State S_n or more specifically, $S_n(t)$ for fixed $t \in \langle 0, \infty)$, is a random variable and indicates that, at time *t, n* requirements are in the system. If *m* requirements, $m > n > 1$, are in the system M/M/*n*/*n*/∞/FIFO, then each requirement is operating in a service channel and the remaining *m-n* are waiting in the queue. Transitions between states that differ in the number of requirements in a system can be understood as a process of birth and death where the requirement birth represents the requirement entry into the system and death corresponds to a requirement leaving from the system after being processed.

We assume a Poisson stream of requirements with a parameter λ and an exponential distribution of service time with parameter μ, generally $\mu \neq \lambda$, and the queueing system behaviour described by the *Markov processes* (Fig. 1).

Due to the regularity, only transition probabilities $P(S_i \rightarrow S_j)$, where either $i = j$ or *i* and *j* differ by 1 have sense.

Using the regularity property and the method of calculating the total probability and neglecting the powers of the interval length Δt, from the partial probabilities of

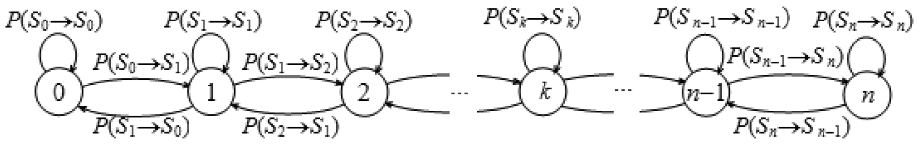


Fig. 1. Graph of M/M/*n*/*n*/∞ system transitions.

conjunction and disjunction of independent events, we get the transition probabilities as follows:

For example, transition probability $P(S_0 \to S_0)$ corresponds to the probability of the event that, during the time interval of length Δt, no requirement enters the system, transition probability $P(S_k \to S_{k-1})$, $n > k \geq 1$, is the probability of the event that, during the time interval of length Δt, no requirement enters the system and at the same time one requirement will be served and leaves the system, transition probability $P(S_k \to S_k)$, $k \geq 1$, is equal to the probability of the event that, during the time interval of length Δt, no requirement enters the system and no requirement leaves the system, or, during this interval, one requirement enters and one requirement will be served, transition probability $P(S_{k+1} \to S_k)$, $n > k \geq 1$, is equal to the probability of the event that, during the time interval of length Δt, one requirement was served in one of the service channels, i.e. $P(S_{k+1} \to S_k) = \mu\,\Delta t + \mu\,\Delta t + \ldots + \mu\,\Delta t = (k + 1)\,\mu\,\Delta t$.

$P(S_n \to S_n)$ would differ in the M/M/*n*/*n*/∞/ (i.e. without the FIFO queue), because no requirement was served and no requirement entered a channel as all were occupied $P(S_n \to S_n) = 1 - (\mu\,\Delta t + \mu\,\Delta t + \ldots + \mu\,\Delta t) = 1 - n\,\mu\,\Delta t$.

Let us summarise the previous considerations:

$$P(S_{k-1} \to S_k) = \lambda\Delta t, k = 1, \ldots, n \tag{1}$$

$$\begin{aligned} P(S_k \to S_k) \;=\; (1 - \lambda\,\Delta t)\,(1 - k\,\mu\,\Delta t) \;=\; 1 - (\lambda + k\,\mu)\Delta t + k\,\lambda\,\mu\,\Delta t^2 \\ 1 - (\lambda + k\,\mu)\,\Delta t, k = 0, \ldots \end{aligned} \tag{2}$$

$$P(S_{k+1} \to S_k) \;=\; (k+1)\mu\Delta t, k = 0, \ldots, n-1 \tag{3}$$

Let $p_k(t)$ denote the probability that, at time t, just k requirements are in the system. Using the previous equations, we can calculate $p_0(t)$, $p_1(t)$, …, $p_k(t)$, …, $p_n(t)$.

$$p_0(t+\Delta t) \;= P(S_0 \to S_0) \;+ P(S_1 \to S_0) \;= p_0(t).(1 - \lambda\Delta t) + p_1(t).\mu\Delta t \tag{4}$$

$$\begin{aligned} p_1(t+\Delta t) \;&= P(S_0 \to S_1) \;+ P(S_1 \to S_1) \;+ P(S_2 \to S_1) \\ &= p_0(t).\lambda\,\Delta t + p_1(t).[1 - (\lambda + \mu)\Delta t] \;+ p_2(t).2\mu\Delta t \ldots \end{aligned} \tag{5}$$

$$\begin{aligned} p_k(t+\Delta t) \;&= P(S_{k-1} \to S_k) \;+ P(S_k \to S_k) \;+ P(S_{k+1} \to S_k) \\ &= p_{k-1}(t).\lambda\,\Delta t + p_k(t).[1 - (\lambda + k\mu)\Delta t] \;+ p_{k+1}(t).(k+1)\mu\Delta t, \quad k = 2, \ldots, n-1 \end{aligned} \tag{6}$$

However, if all channels are occupied and the queue is nonempty, the last equation changes to (7).

$$p_k(t+\Delta t) = p_{k-1}(t).\lambda\,\Delta t + p_k(t).[1-(\lambda+n\mu)\Delta t] + p_{k+1}(t).n\mu\Delta t, k \geq n \tag{7}$$

After easy simplification of Eqs. (4), (6) and (7), we obtain Eqs. (8), (9) and (10).

$$\frac{p_0(t+\Delta t)-p_0(t)}{\Delta t} = -\lambda p_0(t) + \mu p_1(t) \tag{8}$$

$$\begin{aligned}\frac{p_k(t+\Delta t)-p_k(t)}{\Delta t} &= \lambda p_{k-1}(t) - (\lambda+k\mu)p_k(t) + (k+1)\mu p_{k+1}(t),\\ k = 1,2,\ldots,n-1\end{aligned} \tag{9}$$

$$\frac{p_k(t+\Delta t)-p_k(t)}{\Delta t} = \lambda p_{k-1}(t) - (\lambda+n\mu)p_k(t) + n\mu p_{k+1}(t), \quad k \geq n \tag{10}$$

Let us now consider a limit transition for $\Delta t \to 0$ in Eqs. (22) and (23). We get:

$$\lim_{\Delta t\to 0}\frac{p_0(t+\Delta t)-p_0(t)}{\Delta t} = \lim_{\Delta t\to 0}[-\lambda p_0(t) + \mu p_1(t)]$$

$$\begin{aligned}\lim_{\Delta t\to 0}\frac{p_k(t+\Delta t)-p_k(t)}{\Delta t} &= \lim_{\Delta t\to 0}[\lambda p_{k-1}(t) - (\lambda+k\mu)p_k(t) + (k+1)\mu p_{k+1}(t)], k\\ &= 1,2,\ldots,n-1\end{aligned}$$

$$\lim_{\Delta t\to 0}\frac{p_k(t+\Delta t)-p_k(t)}{\Delta t} = \lim_{\Delta t\to 0}[\lambda p_{k-1}(t) - (\lambda+n\mu)p_k(t) + n\mu p_{k+1}(t)], \quad k \geq n$$

The expressions on the left-hand side of the previous two equations are derivatives of the functions $p_0(t)$ and $p_k(t)$ at point t, i.e. $p_0'(t)$ and $p_k'(t)$, while, on their right-hand sides, the limit transition does not have any effect. Hence, we get recurrence Eqs. (11), (12) and (13) as follows:

$$p_0'(t) = -\lambda p_0(t) + \mu p_1(t) \tag{11}$$

$$p_k'(t) = \lambda p_{k-1}(t) - (\lambda+k\mu)p_k(t) + (k+1)\mu p_{k+1}(t), \;\; k = 1,2,\ldots,n-1 \tag{12}$$

$$p_k'(t) = \lambda p_{k-1}(t) - (\lambda+n\mu)p_k(t) + n\mu p_{k+1}(t), \;\; k \geq n \tag{13}$$

These recurrence equations are a set of infinitely many first-order ordinary differential equations. To address them, we need to know the initial conditions, which are given by the state of a system at time $t_0 = 0$. If there are k_0 requirements in a system at time $t_0 = 0$, then the initial conditions are given by (14) and (15)

$$p_{k_0}(0) = 1 \tag{14}$$

$$p_k(0) = 0, \quad k \geq 1, \; k \neq k_0 \tag{15}$$

In the sequel, we assume that $\lambda < \mu$, i.e. $\lambda/\mu < 1$. Denote the ratio λ/μ by the ψ symbol. We call it the *intensity of the system load*. The condition

$$\frac{\lambda}{n\mu} < 1 \tag{16}$$

is a necessary and sufficient condition for a queue not to grow beyond all bounds. This condition also ensures that, after a sufficiently long time from the opening of a queueing system, its situation stabilizes, i.e., there are limits

$$\lim_{t\to\infty} p_k(t) = p_k, \quad k = 0, 1, \ldots, \tag{17}$$

and then, after a sufficiently long time from the opening of a queueing system, the probabilities $p_k(t)$ can be regarded as constant, i.e.,

$$p_k(t) = p_k = \text{const} \tag{18}$$

Since the derivatives of constants are zeros, by Eqs. (11), (12) and (13), we get an infinite set of linear algebraic equations determined by (19), (20) and (21).

$$0 = -\lambda p_0 + \mu p_1 \tag{19}$$

$$0 = \lambda p_{k-1} - (\lambda + k\mu) p_k + \mu p_{k+1}, \quad k = 1, 2, \ldots, n-1 \tag{20}$$

$$0 = \lambda p_{k-1} - (\lambda + n\mu) p_k + n\mu p_{k+1}, \quad k \geq n \tag{21}$$

From the above system of equations, we get:

$$p_k = \frac{\psi^k}{k!} p_0, \quad k = 1, \ldots, n-1 \tag{22}$$

$$p_k = \frac{\psi^k}{n!} \frac{n^n}{n^k} p_0, \quad k \geq n \tag{23}$$

Obviously, we have

$$\sum_{k=0}^{\infty} p_k = 1 \tag{24}$$

Now p_0 remains to be found. To do this, we use Eqs. (19), (20) and (21).

$$p_0 \left[\sum_{k=0}^{n-1} \frac{\psi^k}{k!} + \sum_{k=n}^{\infty} \frac{\psi^k}{n! n^{k-n}} \right] = 1 \tag{25}$$

The second expression in brackets may be simplified using a geometric series with quotient ψ/n as follows

$$\sum_{k=n}^{\infty} \frac{\psi^k}{n! n^{k-n}} = \frac{\psi^n}{n!} \sum_{k=n}^{\infty} \frac{\psi^{k-n}}{n^{k-n}} = \frac{\psi^n}{n!} \sum_{k=n}^{\infty} \left(\frac{\psi}{n}\right)^{k-n} = \frac{\psi^n}{n!} \sum_{i=0}^{\infty} \left(\frac{\psi}{n}\right)^{i} = \frac{\psi^n}{n!} \frac{1}{1-\frac{\psi}{n}}$$

Therefore, by (25), we get

$$p_0 = \left[\sum_{k=0}^{n-1} \frac{\psi^k}{k!} + \frac{\psi^n}{n!} \frac{1}{1-\frac{\psi}{n}}\right]^{-1} \tag{26}$$

These equations may now be used to derive other important characteristics of the M/M/*n*/*n*/∞/FIFO system, which include:

1. *Mean number of requirements in the system*:

$$E(N_s) = \overline{n_s} = \sum_{k=0}^{\infty} k\, p_k \tag{27}$$

2. *Mean number of requirements in the queue* (*mean queue length*):

$$E(N_f) = \overline{n_f} = \sum_{k=n}^{\infty} (k-n) p_k \tag{28}$$

3. *Mean number of free service channels*:

$$E(N_c) = \overline{n_c} = \sum_{k=0}^{n-1} (n-k) p_k \tag{29}$$

4. *Mean time spent by a requirement in the system*:

$$E(T_s) = \overline{t_s} = \frac{\overline{n_s}}{\lambda} \tag{30}$$

5. *Mean waiting time of a requirement in the queue*:

$$E(T_f) = \overline{t_f} = \frac{\overline{n_f}}{\lambda} \tag{31}$$

6. *Factor of service channel idle time*

$$K_0 = p_0 \tag{32}$$

7. *Factor of service channel load*

$$K_1 = 1 - p_0 \tag{33}$$

4 Simulation of Queueing Processes

As, in practice, some assumptions may not be satisfied, particularly the Poisson (Markov) process properties of stationarity and the independence of increases, such as the number of clients in shops and railway stations substantially changing during the daytime, the formulas that we have derived, may not be entirely accurate. However, queueing systems can also be studied by Monte Carlo simulations, which generate random numbers representing the moment of the requirements entering into the system and the service time.

If the values of these random variables should have a certain probability distribution, then it must be provided.

To do this there are many methods, such as the *elimination method* and *inverse function method*. The elimination method may be employed to generate the values of continuous random variables whose probability density f is bounded in an interval $\langle a, b\rangle$ and zero outside this interval. The principle of this method is based on the fact that we generate random points with coordinates (x, y) with uniform distribution in the rectangle $\langle a, b\rangle \times \langle 0, c\rangle$, where c is the maximum value of the probability density f in the interval $\langle a, b\rangle$.

If the point generated is under the graph of the function f, i.e., $y \leq f(x)$, then x is regarded as the generated value of a random variable with the given distribution, otherwise it is not, that it, is discarded from the calculations. In the inverse function method, we first determine the probability distribution function F from the density function f by Eq. (34).

$$F(x) = \int_{-\infty}^{x} f(t)\,dt \tag{34}$$

We generate a random number r with uniform distribution on the interval $\langle 0, 1\rangle$, that we consider as the value of the distribution function at a still unknown point x, i.e., $F(x) = r$. The point x here is obtained by the inverse function (35):

$$x = F^{-1}(r) \tag{35}$$

During the simulation experiments, it is necessary to decide how to express the dynamic properties of the model, i.e., what strategy you choose for recording time. There are two options - a *fixed time step method* and the *method of variable time step*.

In the first case, at fixed intervals of time, changes are monitored. In the variable time step method, the time step bounds are exactly those times at which the system changes such as a new requirement arrives or the required service is terminated and the requirement leaves the system.

In [5], the M/M/n/n/∞/FIFO system was implemented in MATLAB using simulation data from a supermarket.

It makes it possible to enter λ (*mean intensity of the input*), μ (*mean service intensity*), the number of service lines n (then it is checked if $\lambda/n\mu < 1$), and the number of requirements. For these data, the probabilities $p_k(t)$ and the above-mentioned characteristics are computed.

To understand the behaviour of the system, the program also offers graphical output of simulations. In Fig. 2, four graphs are shown for a system with three lines, which show requirement arrival times, requirement service times, requirement waiting times for service and finishing times of services for these requirements.

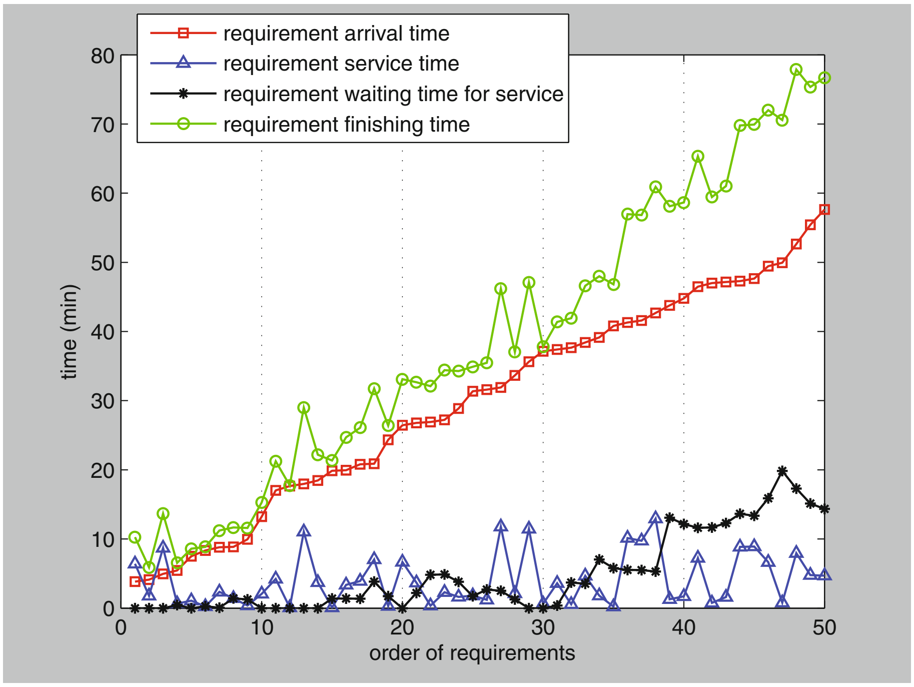


Fig. 2. Simulation of the M/M/n/n/∞ system for $\lambda = 45$, $\mu = 18$, $n = 3$, and 45 requirements.

Now we compare the analytical solution with the values obtained by simulation for different numbers of requirements. In the analytical part, we use formulas (22), (23) and (26), derived in Sect. 3, and the corresponding characteristics (27), etc. Simulations were run for 50, 200, and 500 requirements (clients). Table 1 sums up the results of the analytical formulas and simulations.

We can see that, if the number of requirements increases, then the difference between the analytical and the simulation results decreases. For 50 requirements, the difference is about 12%, but for 500 requirements, only 5%. Based on these achievements, we can conclude that the computer implementation of the simulation model reasonably approximates the M/M/n/n/∞/FIFO system.

Table 1. A comparison of analytical and simulation results

	Analytical evaluation mean values	Simulation results number of requirements		
		50	200	500
$E(T_s)–E(T_f)$	3.33333	3.64353	3.5118	3.45478
$E(T_f)$	4.68165	3.06941	4.17567	5.21177
$E(T_s)$	8.01498	6.71293	7.68747	8.66655
$E(N_s)$	6.01124	5.09125	5.62803	6.23332
$E(N_f)$	3.51124	2.30994	3.05703	3.74851
$E(N_s)–E(T_f)$	2.5	2.78131	2.57101	2.48481

5 Conclusions

This paper describes an approach to modelling of queuing system based on the use of Markov processes and, for a $M/M/n/n/\infty$/FIFO system, derives its characteristics in detail.

As some of the assumptions of the theoretical derivations may not be satisfied, such as the stationarity and the independence of increases, the simulation approach was also presented and implemented in MATLAB. This makes it possible to get all the statistics for any time intervals of real processes given the number of service lines, the times of requirement arrivals, and their service times. However, the approximation of theoretical model by implemented model, which computes with real data, depends on the number of requirements (and, therefore, on the time interval length). The higher the number of requirements, the better precision the simulation model has.

References

1. Bolch, G., Greiner, S., Meer, H., Trivedi, K.S.: Queueing Networks and Markov Chains. Wiley, New York (2006)
2. Bose, S.K.: An Introduction to Queueing Systems. Springer-Verlag, Berlin (2001)
3. Cooper, R.B.: Introduction to Queueing Theory. North Holland, New York (1981)
4. Gross, D., Shortle, J.F., Thompson, J.M., Harris, C.M.: Fundamentals of Queueing Theory. Wiley, New York (2008)
5. Horký, M.: Models of Queueing Systems. Diploma Thesis, Brno University of Technology, Brno (2015)
6. Hrubina, K., Jadlovská, A., Hrehová, S.: Optimisation Algorithms Using Programme Systems. Lecture Notes. Technical University in Košice, Prešov-Košice (2005)
7. Šeda, M.: Models of queueing systems. Acta Logistica Moravica **1**, 16–33 (2011)
8. Virtamo, J.: Queueing Theory. Lecture Notes. Helsinki University of Technology, Espoo (2005)
9. Willig, A.: A Short Introduction to Queueing Theory. Lecture Notes. Technical University, Berlin (1999). 42 pp.

Influence of Random Number Generators on GPA-ES Algorithm Efficiency

Tomas Brandejsky(✉)

Czech Technical University in Prague,
Konviktska 20, 11000 Prague 1, Czech Republic
brandejsky@fd.cvut.cz

Abstract. The presented paper deals with problem of studying of Random Number Generators onto properties (especially efficiency) of GPA-ES algorithm. The first chapter brings simple introduction into the problem followed by description of GPA-ES algorithm from the viewpoint of influence of RNGs onto its function. The third chapter then describes organization of experiments and the next one brings their implementation details and simulation results. Then the fifth chapter discusses results, especially influence of different RNGs onto GPA-ES algorithm function and the sense and form on the next experiments.

Keywords: Genetic programming algorithm · Random number generator · Efficiency · Optimization · Symbolic regression

1 Introduction

For many years researchers discusses influence of random number generator into quality and especially speed, of evolutionary process. There it is hard to recognize significant behaviors and features of the optimal random number generator, see e.g. [1, 2] discussing applicability of non-random functions on the place of RNGs [3]. New standard of C++, revision 11, contains advanced Random Number Generators (RNGs) library <random>. This library simplifies testing of influence of many RNGs on the quality of evolutionary process [4]. Similarly as in the work [1], the GPAES algorithm of precise symbolic regression is used. This algorithm allows discovering of analytical model of data series in the form of first order differential equations set in the quality similar to description produced by human mathematicians. It means, to discover models in minimalist form without overcomplicated structures which are applicable to prediction tasks, etc.

The tests were using all standard RNGs form the <random> library. There was recognized number of evolutionary steps, not the computing time, because the computing time strongly depends on the used HW and it is impossible to compare computational times reached on e.g. Intel Xeon Phi, CUDA and IA64 processor architectures. As the test bed, there was used Lorenz attractor backward symbolic regression problem (symbolic regressing on the base of data generated by Lorenz attractor system equations). On the base of these tests unexpected results were obtained. Especially it is interesting that the structure of searched solution bigger influence to required average number of evolutionary operations than the selection of

RNG, when search of equation describing y variable of Lorenz attractor system needs five times more iterations than discovering of equation describing z variable with the equal precision limit.

This fact causes that dependency of needed number of iterations on properties of used mutation and crossover operators is stronger than influence of selected RNG.

2 GPA-ES Algorithm

The structure of the examined GPA-ES algorithm is outlined on Fig. 1. It was discussed e.g. in the paper [5]. The main component of this algorithm is formed by standard Genetic Programming Algorithm and because this algorithm in inefficient in precise solution identification, especially in identification of structure parameters, presented GPA-ES algorithm calls specialized Evolutionary Strategy algorithm to this work for each individual of the GPA individuals population. As it was discussed in the works [6–8], the parameters pre-optimization by separate ES algorithm increases preciseness and efficiency of parameter identification, gives better quality of discovered structures (prevents blowing of identified structure complexity, increases the structural sensitivity of the algorithm) and in the presented project it allows to separate influence of RNG onto formed structures and parameters.

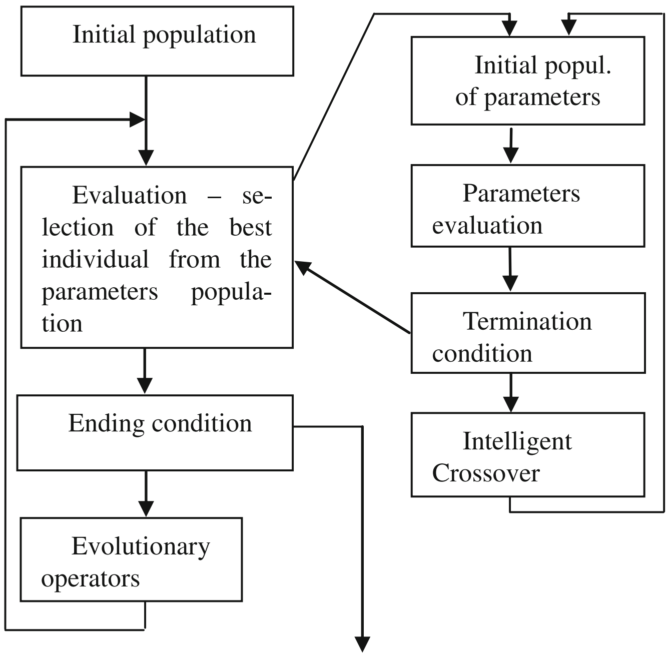


Fig. 1. The structure of hybrid GPA-ES algorithm with GPA (left column) individual parameters optimization by inherited ES (right column of blocks)

The algorithm outlined by Fig. 1 applies RNGs on various places to many uses. In the GPA main loop of the algorithm they are:

- Generating of the initial population - this generator controls occurrence of algebraic operators and terminals (variables and parameters in the initial algebraic expressions). The GPA algorithm expects normal distribution of applied RNG if the structures are formed but there is question if another distribution might improve quality of initial variables selection). Constant magnitudes are generated later by the specialized ES algorithm.
- Control of the GPA run (selection of evolutionary operators) plays significant role for the GPA-ES system efficiency. Also here used GPA expects normal distribution of used RNG.
- Control of the mutation operator is place of application of another RNG which controls selection of place of mutation and generating of new structures. The different RNGs might influence occurrence of specific structures in the genome and different distribution of generated random numbers also influence depth of mutation – if they are strongly changing the generated structures or of they are rather shallow, if they positions are close to terminals.
- Control of the crossover operator has assigned specific RNG too. It enables to evaluate influence of this RNG to behavior of this operator and thus to efficiency of the whole GPA algorithm.
- Regenerating of the population in the case of inefficient extremely long run (only the best individual is preserved in this case). It should be useful if the structures generated in this stage will not be very similar to structures presented in the previous genome.

The ES algorithm uses RNGs to initialization, population regeneration and to control of the ES algorithm intelligent crossover operator because the only uniform intelligent crossover evolutionary operator is applied. This operator provides linear interpolation of original populations understand as points in multidimensional space and then randomly selects new point on this interpolation line. Because the requirements of this algorithm are different, it is possible to expect that different kind of RNG than in the case of GPA will be suitable.

3 Experiment Organization

The experiments were computed on Anselm supercomputer of IT4Innovations National Supercomputing Center. The aim was to provide first study of influence of RNG onto GPA-ES algorithm efficiency. In this study all RNGs are uniform (of the equal type). The examined generators included rand() function defined in <cstdlib> of GNU C++ version 4.8.1 implementing new version 11 of the C++ language standard, true random number generator that produces non-deterministic random numbers on the base of stochastic processes, 32 bit Mersenne Twister 19937 generator, 48 bit and 24 bit subtract-with-carry pseudo-random generator with accelerated advancement and multiplicative congruential pseudo-random number generators - ranlux (a type of linear congruential engine) with 2 different parameter groups of the novel <random> library implemented in the new version 11 of the C++ language. This random library is fully object oriented, contains large number of random number generators and distribution modifiers and it is thread safe.

Each experiment consists of 1200 different initial (seed) values. For each seed magnitude (eliminating influence of random initialization of RNG) the GPA-ES algorithm was runed with population of 1000 GPA individuals and 1000 ES individuals assigned to each GPA individual. As it is written above, the symbolic regression of equations from the data set describing movement of Lorenz attractor system described by (1) and applied parameters by (2) was used as test bed. The regressed data are in the form of table of 600 points describing Lorenz attractor system movement sampled with period of 0.01 s. The estimation process stops when sum of error squares across all 600 points is less or equal to 1 * 10^−7. Because the efficiency of the used evolutionary algorithm depends on structure of searched solution [9], the only one problem is solved to allow comparability of the results. On the opposite side, it should be better to use wide area of problems to cover possible suitability or non suitability of the algorithm to the testing task. In the future, more tasks will be tested but in this moment the research is limited by available amount of computation time.

$$\begin{aligned} x'(t) &= \sigma(y(t) - x(t)) \\ y'(t) &= (\rho - z(t)) - y(t) \\ z'(t) &= x(t)y(t) - \beta z(t) \end{aligned} \tag{1}$$

$$\sigma = 16 \quad \beta = 4 \quad \rho = 45.91 \tag{2}$$

Anselm system as other supercomputers has Non Uniform Memory Architecture and it is useful to respect this fact in experiment preparation. The GPA-ES system runs GPA algorithm and during this run it starts 1000 ES systems to optimize parameters of each potential solution in parallel as particular threads of the original single task. To perform this mechanism it uses OpenMP library and typically it is limited to single node of NUMA system. To use the supercomputer efficiently, many copies of the algorithm working with particular initial setting of the RNG (seed value) are run. High efficiency is done by large parallelism when each node of the supercomputer might be used (the only limitation of the number of parallel tasks is the number of tasks to be run. There is no decrease of efficiency given by communication between tasks and by their synchronization, because there is nothing such. Each task runs independently on the others and tasks might run both in series and in parallel.

4 The Experiments

The first problem solved during the solution of the research project “Random number generators influence onto Evolutionary Algorithm behaviors” of IT4Innovations consortium was the verification of the idea that RNGs influence efficiency of the whole GPA-ES. These experiments will be followed by studying of influence of RNGs onto GPA and ES module separately and latter onto particular operations of these algorithms. The big problem of these studies is the amount of needed experiments.

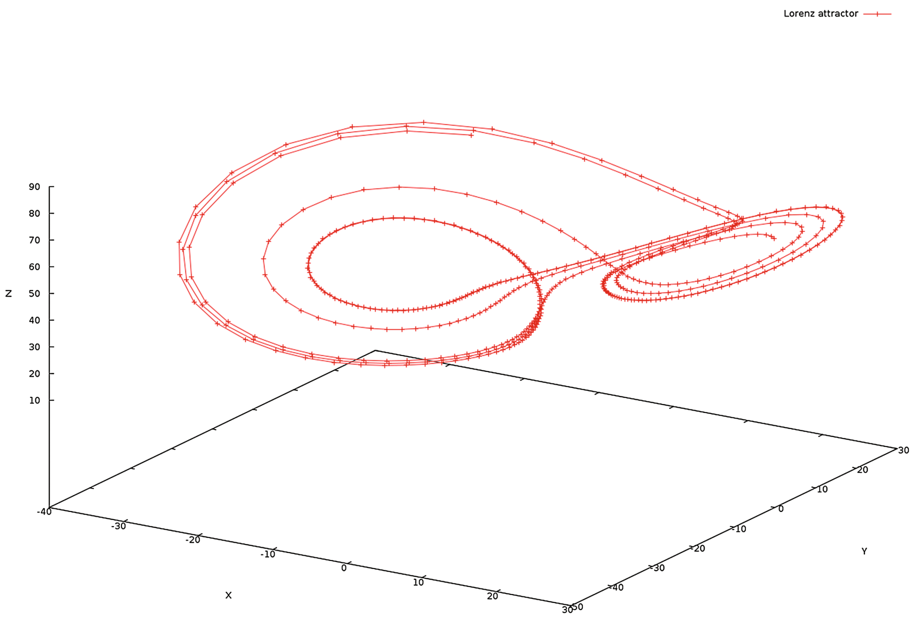


Fig. 2. Data used to Lorenz attractor equations symbolic regression

Full study of all combinations of specific RNGs for each place of use of RNG in GPA-ES algorithm in the case of 10 different RNGs and 18 distribution modifiers offered <random> library requires testing of 180^8 combinations. Even if we suppose 1000 test provided in each case only, the total number of test is bigger than 10^14 and because each test needs about 5 core/hour, the total needed compute time is about 5 * 10^14 h. Such amount is unreachable, but there exist one way of simplification (Fig. 2):

It is possible to form proposition that the influence of each independent application of RNG (e.g. on generating of GPA initial population) is independent on application on another place of GPA-ES algorithm like e.g. to control of ES intelligent crossover operator. If this proposition is valid, it is possible to decrease previous estimation of needed cases to 180 * 10 combinations or 1800000 test cases and 9 * 10^6 core/hours of computation. Such amount seems to be reasonable.

Figures 3 and 4 outline dependency of iterations number on number of experiments. Presented research is forced to use smallest possible number of experiments because the number of experiments multiplies the total time of experiments and because the number of experiments is at least 1.8 * 10^6, good estimation of this number decides about realizability of the project. On the opposite side, such number of experiments gives rather information about order of RNGs than about their precise features and because the properties of averaging, if the number is experiments is multiplied by 10, the precision of the result estimation increases 10 times. The figure representing number of iterations of variable X is not presented, because it is equal to 1 for all RNGs. There is the way how to increase sensitivity – to use smaller populations,

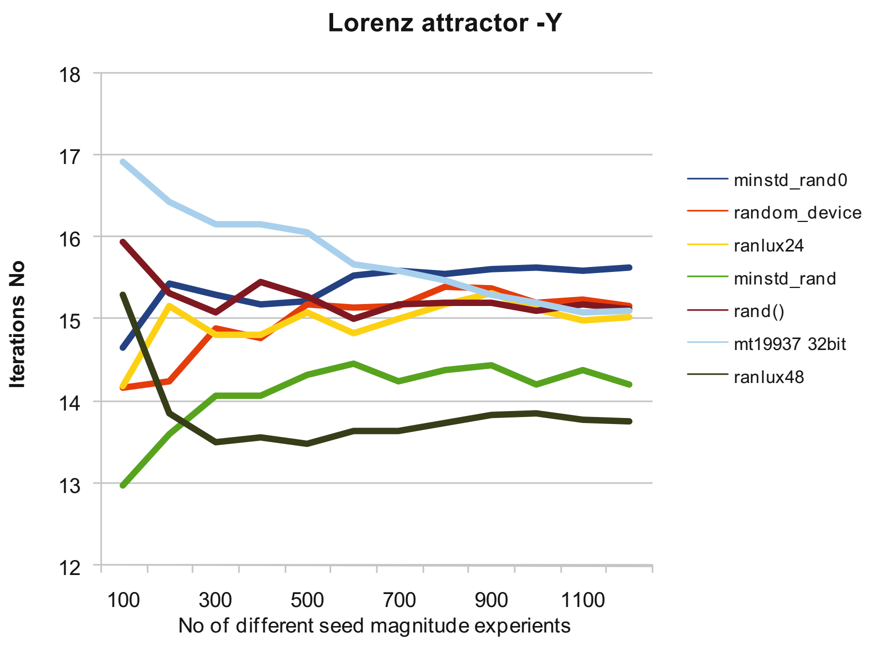


Fig. 3. Dependency of number of iterations on number of experiments of Lorenz Y variable symbolic regression.

but this way tends to larger spread of results, to smaller reliability. In the next research it is need to reason about 10000 experiments for each combination of RNGs.

5 Discussion

The presented Figs. 3 and 4 points that such RNGs as multiplicative congruential pseudo-random number generators gives good results in both tasks – identification of Y and Z function as well as standard rand() function or minimal standard (minstd_rand) generator. On the opposite side, true random number generator (random_device) is not very useful for GPA-ES algorithm and produces poor results as well as 32 bit Mersenne Twister 19937 (mt19937) generator non looking to their statistical qualities and it will be interesting to study this phenomena in the future works.

There is also interesting bigger variance of Z function identification time. It is caused by less precise measurement given by five times less numbers of iteration cycles and concluding increase of measurement errors (Table 1).

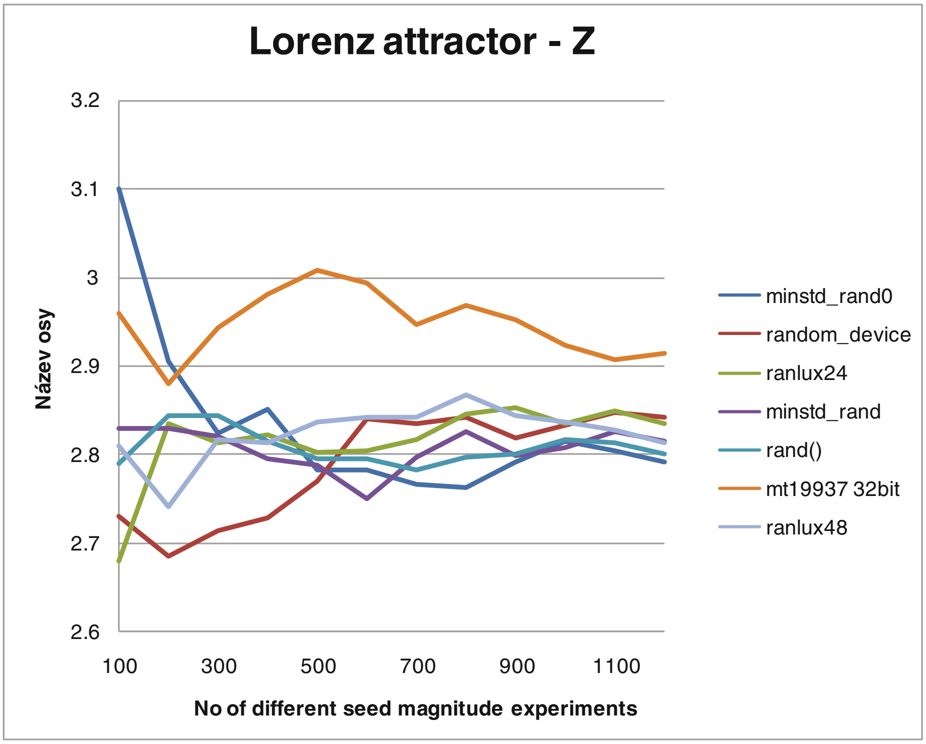


Fig. 4. Dependency of number of iterations on number of experiments of Lorenz Z variable symbolic regression.

Table 1. Comparison of different RNGs (smaller No of iterations is better)

	Minstd0	RDevice	Ranlux	Minstd	Rand()	MT19937	Ranlux15
Lorenz y	15.64	15.17	15.04	14.21	15.12	15.01	13.77
Lorenz z	2.79	2.84	2.84	2.81	2.8	2.91	2.81

6 Conclusions

The structure of searched solution has bigger influence to required average number of evolutionary operations (and needed evaluations of fitness function) than the selection of RNG (the problem of dependency between final solution structure and required amount of GPA-ES iterations was studied in [9]).

This fact causes that dependency of needed number of iterations on properties of used mutation and crossover operators is stronger than influence of selected RNG.

The first experiments also pointed that it is need to use large test sets to get reliable results. The order of RNGs on the base of needed iterations is stable for at least 900 different seed values for variable Y and for 1100 seed values for Z variable.

The magnitude estimation error is about 1% in this case and its increase requires multiple increase of number of iterations.

Because it is probable that GPA a ES algorithms composing applied GPA-ES system has different requirements to RNGs properties, it is need to explore influences of different RNG onto particular GPA-ES modules in detail in the next research.

Presented study opens research within the project studying the relevance of RNGs properties to GPA behaviors. It is also the reason why these study is based on symbolic regression tests and not the standard benchmarks.

Acknowledgements. This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center – LM2015070".

References

1. Brandejsky, T.: Limited randomness evolutionary strategy algorithm. In: Matoušek, R. (ed.) Mendel 2015. AISC, vol. 378, pp. 53–62. Springer, Cham (2015). doi:10.1007/978-3-319-19824-8_5. ISSN 2194-5357, ISBN 978-3-319-19823-1
2. Senkerik, R., Pluhacek, M., Kominkova-Oplatkova, Z.: Simulation of time-continuous chaotic systems for the generating of random numbers. In: Proceedings of the 18th International Conference on Systems (part of CSCC 2014), Santorini Island, Greece, 17–21 July 2014. Latest Trends on Systems, vol. II, pp. 557–561 (2014). ISSN 1790-5117, ISBN 978-1-61804-244-6
3. Skanderova, L., Zelinka, I., Šaloun, P.: Chaos powered selected evolutionary algorithms. In: Zelinka, I., Chen, G., Rössler, O., Snasel, V., Abraham, A. (eds.) Nostradamus 2013: Prediction, Modeling and Analysis of Complex Systems. Advances in Intelligent Systems and Computing, vol. 210, pp. 111–124. Springer, Heidelberg (2013)
4. Cantú-Paz, E.: On Random Numbers and the Performance of Genetic Algorithms. Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore
5. Brandejsky, T.: The use of local models optimized by genetic programming algorithm in biomedical-signal analysis. In: Zelinka, I., Snášel, V., Abraham, A. (eds.) Handbook of Optimization from Classical to Modern Approach, pp. 697–716. Springer, Heidelberg (2012). ISBN 978-3-642-30503-0
6. Brandejský, T.: Genetic programming algorithm with parameters preoptimization - problem of structure quality measuring. In: MENDEL 2005, pp. 138–144. Brno University of Technology, Brno (2005). ISBN 80-214-2961-5
7. Brandejský, T.: Multi-layered evolutionary system suitable to symbolic model regression. In: Recent Researches in Applied Informatics, vol. 1, pp. 222–225. WSEAS Press, Athens (2011). ISBN 978-1-61804-034-3
8. Brandejský, T.: Evolutionary systems in complex signal analysis. In: Interdisciplinary Symposium on Complex Systems, ISCS 2014, pp. 101–108. Springer, Dordrecht (2015). ISSN 2194-7287, ISBN 978-3-319-10758-5
9. Brandejský, T., Zelinka, I.: Specific behaviour of GPA-ES evolutionary system observed in deterministic Chaos regression. In: Zelinka, I., Rössler, O.E., Snášel, V., Abraham, A., Corchado, E.S. (eds.) Nostradamus: Modern Methods of Prediction, Modeling and Analysis of Nonlinear Systems, pp. 73–81. Springer, Heidelberg (2013). ISSN 2194-5357, ISBN 978-3-642-33226-5

Genetic Algorithm Based Random Selection-Rule Creation for Ontology Building

Henrihs Gorskis(✉), Arkady Borisov, and Ludmila Aleksejeva

Riga Technical University, Kalku Street 1, 1658 Riga, Latvia
{henrihs.gorskis,Ludmila.Aleksejeva_1}@rtu.lv,
arkadijs.borisovs@cs.rtu.lv

Abstract. This paper investigates the possibility of creating ontology concepts from information contained in a database, by finding random queries with the help of a genetic algorithm. This is done, with the aim to help ontology building. Based on the structure of the database random chromosomes are created. Their genes describe possible selection criteria. By using a genetic algorithm, these selections are improved. Due to the size of the database, an approach for finding fitness from general characteristics, instead of an in-depth analysis of the data is considered. After the algorithm finished improving the chromosomes in the population, the best chromosomes are chosen. They are considered for implementation as ontology concepts. These ontology concepts can be used as descriptions of the information contained in the database. Because genetic algorithms are not usually used for ontology building, this paper investigates the feasibility of such an approach.

Keywords: Ontology · Databases · Genetic algorithm · Data mining

1 Introduction

The ontology description of concepts can be a useful tool. It can be used as an interface between a raw or complicated data source and the user of the data. This is possible because ontology provides the means of classifying, naming and relating data. Different or the same data can have one or more names. Data, therefore, obtains additional descriptive features in the form of ontology concepts. For example, a person that is older than 10 and younger than 20 years old can be called a teenager. By using the term teenager, one is describing data attributes. If one had a database of persons, the term teenager could be applicable to multiple records. Terms like this can be used as an addition to the database, when the database itself does not use such concepts. When concepts are based on real data, concept names describe features of data. These concepts can be used without showing the data. If one had a list of relevant terms, usable with the specific data, it would be possible to operate with these terms, without having to work with the data directly. In certain situation this can be helpful to the user of the data. For example, a user of a database wishes to find historical or current data about some subjects. The user has to know how to create custom queries to the database using the structures and rules of it. Otherwise, the user has to rely on predefined database reports, which may not provide the desired result. When this user does not know the query language or the intricacies of the structure of the database, he will not be able to get to the data he desires. Had the data an additional layer of ontology concepts linked to it, this task would be made easier. By using names familiar to the user, the user would be able to get to the data, without having to write complex queries.

Ontology capabilities can be added naturally, by a knowledge engineer. The engineer would add concept name and descriptions manually. The creation of ontology concepts can be a time consuming and difficult task [1]. Some concepts are more clear and easy to define. Other concepts are specific to the problem domain. The knowledge engineer would have to be an expert of both ontology and the problem domain, to be able to define usable and useful concepts. Ways of creating ontology concepts in an automated or semi-automated way are desired. However, automated ontology building can also be very complicated. The reason for the complication is that, at this time, no algorithm is capable of automatically grouping multiple data into classes understandable to humans. Even sophisticated mathematical clustering algorithms only group data by a general measure of closeness. Without a sophisticated understanding of human nature and language, automated concept description is impossible. This paper investigates some approaches of automated ontology building and proposes the use of both clustering and genetic algorithms as a way of generating possible candidates for concept descriptions. These descriptions could then be used by a human knowledge engineer as clues for further ontology development.

One approach for automated ontology concept definition creation is clustering. By finding naturally occurring groups of data, it can be assumed that these groups can be described and given names. Clustering algorithms create groups of data with similar features. Since individuals of the same class also should share similar features, one can hope that these groups can serve as templates for classes. Some groups, of cause, will be accidental. This is due to the imperfections of clustering algorithms.

Ontology conceptualization and the descriptions of individuals can be used for many different tasks. Agent based systems, sometimes, use ontology on the communication layer. Some ontology based systems might use the descriptions of concepts directly, for some task. The direction of ontology use, considered the most important by this paper, is the description of data using ontology. It is also called ontology-based data access. From this perspective, ontology concepts, which do not describe data, or are not somehow related to data, matter little. Because of this, ontology based systems should make use of and be related to databases. Databases are the main storage for most information and data in most systems. Describing ontology approaches without considering databases will be lacking crucial aspects. However, when one is dealing with databases it is important to consider, that most databases are very large in size and may be restricted in the resources provided to one user. This leads to the need of using some optimization in regard of these factors. Genetic algorithms are known to be able of searching large spaces and using comparatively little resources to do so. This makes them attractive for any task involving searches in large spaces.

This paper looks for ways of helping a knowledge engineer by finding some groups of data in a database. Groups, in this case, are data from the database, which can be obtained with a dynamically constructed “select” query. The found groups are presented to the knowledge engineer to aid ontology building. The process is shown in Fig. 1.

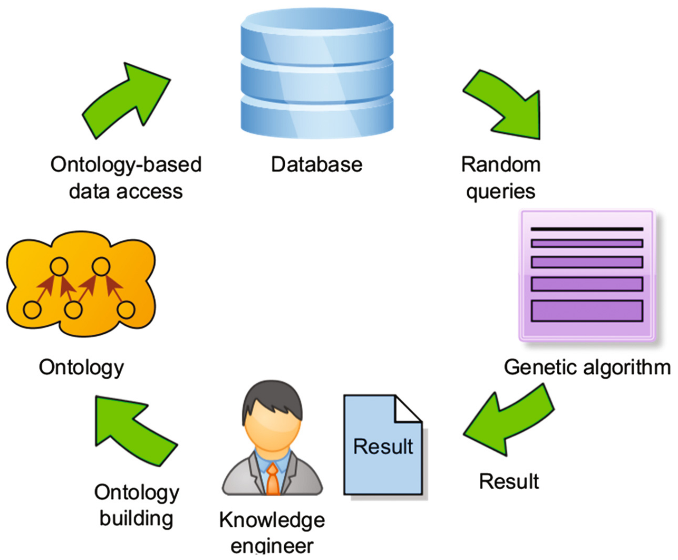


Fig. 1. Proposed aided ontology building process

In order to use the capabilities of the ontology with a database, the ontology must be based on real data from the database. A database contains tables and columns. Tables can be linked in various ways. This makes it difficult to use classical clustering techniques. Because of the size and heterogenic nature of the data in a database, not all classical approaches can be used. It is not uncommon for a database table to have purposefully missing values. Many classical approaches removes records with missing values, or removes columns with many missing values, or inserts average values into fields with missing values. This makes little sense when dealing with database structures. It is also important to consider the relations between tables, and how different combinations of tables influence the number of records. The same real world object can be described using multiple tables. Also columns can describe very different types of data. Some continuous numeric values, others discreet symbolic values. All this leads to a complexity, which not every algorithm is capable of dealing with.

When one is looking for concepts it is important to know what a concept is. A concept describes some idea. Some concepts might be more specific, other more general. This leads to the question of what a meaningful concept is and how to find it. Even though it is difficult to describe what a good concept is, it is possible to name some features of good and bad concepts. By using a genetic algorithm for clustering, it is hoped to achieve the creation of meaningful groups. The features of meaningful concepts are used in the fitness function. By using a genetic algorithm it can be possible to obtain a subset of tables and column characteristics representing a group. This group can become the basis for an ontology concept.

2 Use of Data Mining for Ontology Building

The use of data mining for the purposes of ontology and other knowledge building is not new. Data mining provides a tool-set of mathematical approaches to find and extract features from data as well as the ability to find commonalities. Whenever there are features found in data, the expectation exists, that these features can be named by a concept [2–4]. It is expected, that similar data was generated by similar real-world things. Therefore the data can be classified as describing said thing. When it comes to ontology building, classical data mining approaches are mostly related to the extraction of knowledge from text data [5–8]. These approaches often create ontology descriptions of the text itself and use words and phrases found in the text to name ontology concepts. This of then creates a dictionary like ontology. Other approaches make use of existing data structures describing related terms [9] as well as other knowledge sources.

Some of the previous works related to this paper did investigate the possibility of obtaining ontology concepts from classification algorithms [10]. In the case of classification, the main idea was, that values used for classification, could be indicators for important data points, which could be used to describe ideas. The result of the previous work indicated the complexity of interpreting purely numeric data as human understandable concepts.

The approach in this paper is to some extend inspired by clustering approaches. However, there are several differences. The main difference between classical clustering and the approach described in this paper is that classical clustering requires the full analysis and comparison of all data points to determine the quality of the clusters. In the case of very large data, when it would take a very long time to do so, other approaches have to be considered. This is also true when the data is not numerical, and the data points have varying attributes, as is the case when working with databases. Because of this, the current approach, described by this paper, cannot determine the quality of the created clusters directly. Instead, by using indirect measures of the number of records selected and the descriptive distance of the selections, it is hoped to achieve comparable results.

3 Application of Genetic Algorithms for Selection Improvement

Genetic algorithms can be used in combination with some classical data mining approaches. For example, genetic algorithms have been successfully used with the k-means algorithm [11]. When genetic algorithms are used instead of the k-means algorithm, the chromosomes consist of genes representing cluster centers. Crossing and mutation changes the positions of these centers. Fitness is calculated from a measure of how well the centers in the chromosome divide the value space. However an approach like that required a large number of calculations. For every new or changed chromosome the values, describing every gene, have to be recalculated. In case of a relatively small data set, this is not a problem. However, when databases are involved these calculations become more difficult.

Ontology concept can be of various kinds. They do not always use all the data available in their descriptions. Some concepts can be based on only one data attribute. Therefore the developed approach also investigates a case, when not all available data columns from the database are used. Furthermore, by using or ignoring related tables the number of record combinations changes. For example, when not every record has a related record in another table, the number of combined records will be smaller, compared to when only one table is used. When a record in one table, has multiple related records in another table, the number of combined records grows. Therefore creating a chromosome where every gene is a database column and using all genes at the same time, would lead to only one subset of the data being explored and classified.

The ability of genetic algorithms to handle very complex situations, by using various structures in chromosomes, and the ability to freely define fitness, was the main reason they were chosen to improve the search for random selections.

4 Searching for Adequate Random Queries in the Database

Taking into account the related approaches and the particularities of the defined problem, the following approach has been developed. The approach uses a genetic algorithm to improve randomly found selections. This necessitates the description of a random selection as a chromosome consisting of genes. The chromosome consists of an arrangement of database column values and operations. Figure 2 shows a representation of the chromosome. Each gene in the chromosome can be not active. This indicates that the column is not used by the selection. Otherwise the gene contains a value used by the operation on the column. If a gene has a value in the gene, it is possible to generate a SQL query from the chromosome. Chromosomes will be different for different databases. For example, in the experiment described in the next section gene n was assigned to the column "salary" in the table "salaries" and the operation "min". When the chromosome is being created this gene may remain inactivated or it may be assigned a value. The value, which it can be assigned, is based on the values recorded in the database. Also when to connected genes, using the same column in the same table but with two different operations, are assigned values a simple check is performed, so as to prevent situations when the resulting request is impossible. This is done to slightly speed up the genetic algorithm, instead of waiting for it to fix such small, but impactful errors, naturally. Any numeric columns can be restricted from two directions. Columns using discreet values can only use equality expressions.

Once a random starting population is build, chromosome crossing and mutations is performed. Mutation randomly selects a gene in the chromosome and generates a testing new value for it, or deactivates it.

The chromosomes can be directly translated into SQL queries. The next step is to evaluate the results of the query in order to determine the quality of the chromosome. Each chromosome must be scored to determine its fitness compared to others. Taking into consideration the difficulties in working with a database, meta-scoring approach has been employed. Instead of scoring the obtained cluster directly, it is scored by is characteristics. It has been determined, the following three characteristics can be indicative of a useful chromosome:

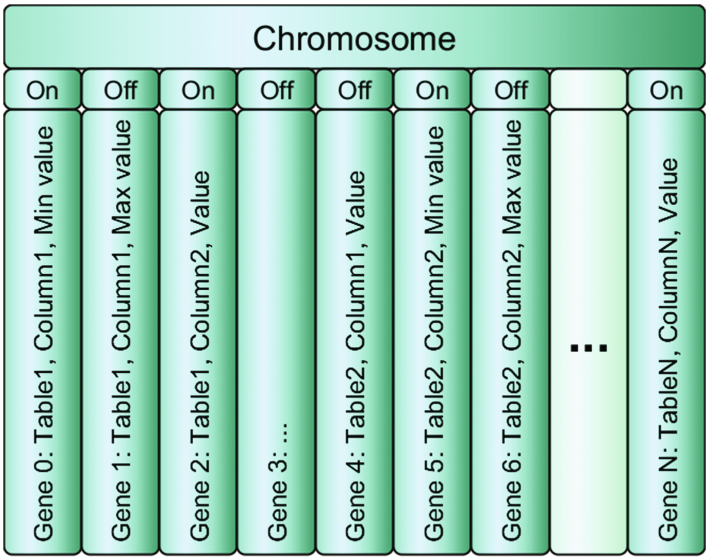

Chromosome
On
Off
On
Off
Off
On
Off
On
Gene 0: Table1, Column1, Min value
Gene 1: Table1, Column1, Max value
Gene 2: Table1, Column2, Value
Gene 3: ...
Gene 4: Table2, Column1, Value
Gene 5: Table2, Column2, Min value
Gene 6: Table2, Column2, Max value
...
Gene N: TableN, ColumnN, Value

Fig. 2. The chromosome structure for the experiment

- A good chromosome divides the value space into reasonable sized chunks;
- A good chromosome is not too simple and not too complex;
- A good chromosome is as different as possible from its siblings.

There can be different concepts and some are more specific than others are. It has been decided that it is too difficult to determine specific concepts and instead to focus on determining more generic concepts. A generic concept would be such, that it encompasses a reasonably sized subset of all available data. A concept, which would have no individuals (database records), would be of no use. A concept, which would be applicable to all individuals, would also be of no use, since it still would not distinguish database records from one another. Therefore, "reasonable" must be somewhere in the middle. For this purpose the information entropy formula (1) was used to calculate one part of the fitness score of a chromosome, where x is the proportion of selected records of all related records.

$$f(x) = -x \times \log_x x - (1 - x) \times \log_2(1 - x) \tag{1}$$

The second bullet point describes the complexity of a chromosome. It was decided to exclude too simple concepts from the search. A too simple concept is based only on one column value. Although these kinds of concepts are very useful, they are also very easy to find without the use of complex solutions. A too complex concept would make use of many columns and many values. These kinds of concept are too difficult to evaluate. It is possible, that they indeed do describe some noteworthy group, but it would be difficult to determine this. Therefore chromosomes, using too many or too few genes are punished by lowering their score, using function 2, where s is the base score obtained in the first function and g is the gene count.

$$f(s,g) = \begin{cases} \frac{s}{m_1 - g + 1}, g < m_1 \\ s, g \geq m_1 \, and \, g \leq m_2 \\ \frac{s}{g - m_2 + 1}, g > m_2 \end{cases} \quad (2)$$

The last bullet point describes how chromosomes are as different as possible from one another. Again, since this approach has some relation to clustering algorithms, it would be of little use if “cluster centers” were located close to each other. In fact, it would be desired that the chromosomes, describing different groups of the database, would be as far away from one another as possible. For this reason another scoring part has been added to the approach. The distance based scoring tries to determine how far away one chromosome is from another. The further away from any other chromosome it is the higher its score. Distance is calculated based on a normalized Euclidian distance with special rules.

The distance used is an artificial measure based on Euclidian distance. If the distance between two chromosomes is zero they are necessarily equal. A distance of one unit represents the maximum possible distance given in a situation. To calculate the distance between two chromosomes their genes are compared. For every gene in the chromosome a distance is calculated. In the case of a gene which restricts continues values, the distance is the difference between the normalized values. Regular normalization is used with the smallest and largest possible values. Therefore the difference can be any value between minus one and one. In the case of a gene restricting discreet values, the difference is zero if they test for the same value and one if they test different values. There are still other cases to consider. If one chromosome uses a gene, whereas the other does not, the difference is also considered as having a distance of one. If both chromosomes do not use the same gene this attribute is skipped. The Euclidian distance is calculated as the square root of the sum of the squares of the differences, as seen in function 3. C1 and C2 are two chromosomes to be compared and g are normalized numeric representations of the gene values of the chromosomes.

$$f(C1, C2) = \sqrt{\sum\nolimits_{i=0}^{n} (g_i^1 - g_i^2)^2} \quad (3)$$

However, because a normalized value between zero and one is desired, the Euclidian distance is divided by the largest possible distance at the number of dimensions used. If the Euclidian distance was calculated between five genes, and all other genes were unused, the distance will be divided by the square root of five to obtain the final distance score, between the two chromosomes. This distance calculation is performed between all chromosomes in the population. The final score, for distances to others, is obtained by first obtaining the distances sum and dividing it by the population size minus one. This measure approaches one as a given chromosome is as different as possible from all others.

All these score are combined to create the final result of the fitness function. There is no guarantee that this approach will indeed find usable concepts, but it should find subgroups which have the attributes of usable concepts

5 Experiments

To test how well the described method is capable of finding concepts an experiment was devised. The example database used for this experiment is the employees sample database provided by MySQL [12]. From the data, contained in the database, groups were found. The groups are evaluated by how many records they describe and how close to each other they are. Once groups are found they are used to create ontology concepts.

5.1 Example Database

The database consists of 6 tales describing historic data about employees of a company. There are around 300000 unique records of basic employee information. Combined with records describing how titles and salary data changed over time, the combined records are almost 3000000 records. When combined together with the data about employment in different departments and the title at the time, the number of combined records is over 5.1 million records. Many of the fields in the database hold dates. Working with dates is not a problem, since dates can be expressed and evaluated as continuous numeric values. However, the nature of the database has to be taken into account during concept creation. Because the information stored in the database is historic in nature, concepts created from it, also have to describe things in an historic context. Figure 3 shows the tables of the database. The yellow squares represent table

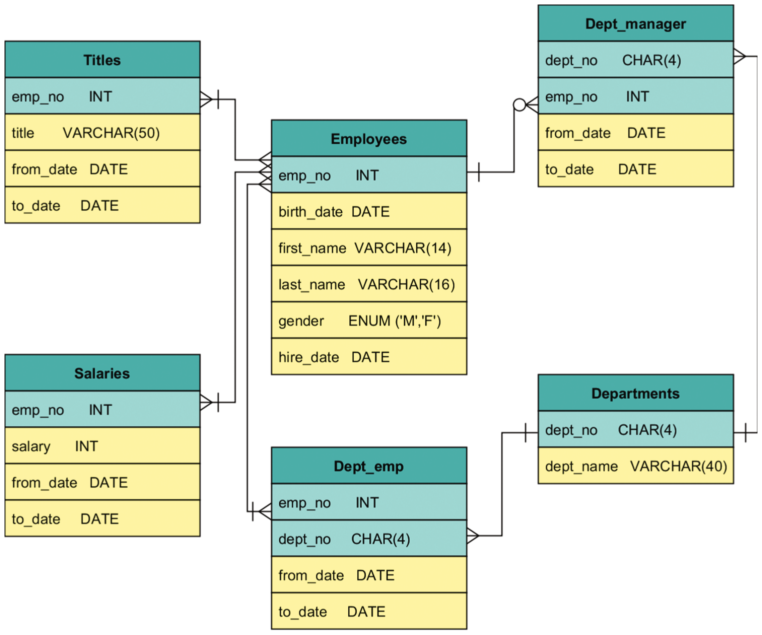


Fig. 3. Example database tables

columns used for the description of employees. The green squares represent columns used to link data. The chromosomes, in the described approach, only use descriptive columns. Linking of database tables is taken care of separately from the concept search part of the approach.

5.2 Database Influence on the Chromosome Structure

The database requires a certain structure of the chromosome. Every database will have different chromosomes and might need additional changes to the evaluation. The database is star shaped, with the “Employees” table at the center of the star. This means that, at its core, the database describes different combinations of employee data. If the database had a more complex structure, it might be necessary to only be looking for one central record type, at a time. This also necessitates that even when a chromosome ignores columns describing basic employee information from the “Employees” table, this table still needs to be used, in order to obtain records correctly and link the together using the employees identification number.

5.3 Results

The experiment has shown that the presented approach generates different data selections, capable of dividing the database into groups of various sizes. In a population of 50 chromosomes, running for 100 generations, 9 groups were created. A group is a collection of chromosomes, whose select statements use the same combination of tables. The chromosomes themselves may be using very different columns and values, but are all based on the combination of the same database tables. Figure 4 shows how the different chromosomes select data in the database. The visualization is based on randomly selected records in the database. As can be seen in the image, because of the addition of a distance measure in the fitness function, most of the chromosomes are very different from one another. However, some of them still are equivalent. Group 6 consists of 3 equivalent chromosomes and can therefore describe only one concept. Groups 3, 8 and 9 contain only one chromosome. The other groups seem to contain more unique chromosomes. Some of the more badly performing chromosomes, for example the one in group 8, were created most recently.

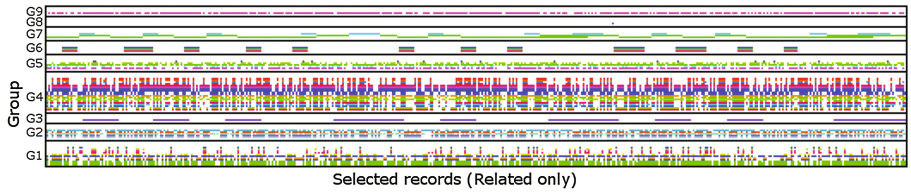


Fig. 4. Overlay of related records

During the execution of the algorithm the added scores of the fitness of the chromosomes have been recorded. Figure 5 shows how the fitness changes over time. As can be seen the overall fitness is less stable, than usual in a genetic algorithm. Usually, the overall fitness of genetic algorithms will continue to rise. Sometimes, the reason why the total fitness will raise is due to the multiplication of the best chromosome in the next generations. The described approach, however, has restrictions preventing certain similarity. The fitness does go up. However, it does so very slowly.

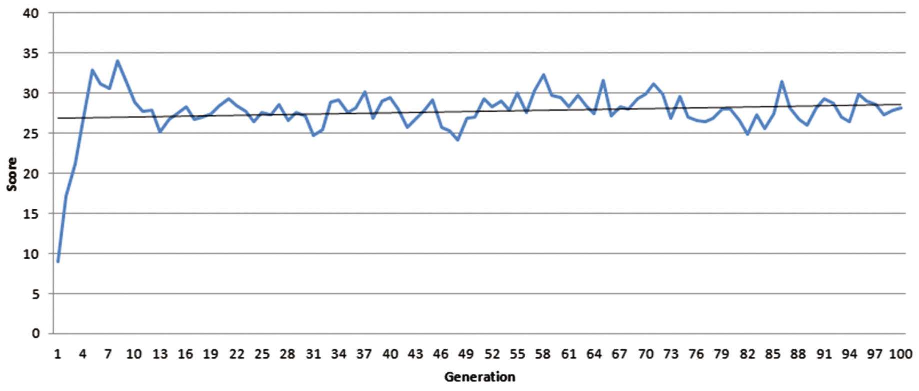


Fig. 5. Score over generations

The obtained groups can be fitted to ontology concepts in the following ways. The most highly rated chromosome in the algorithm defined a concept of an employee, who was born after 1957-05-15 and has been receiving a new salary after 1986-05-16. This selection is applicable to 1683799 out of 2844047 related records. It therefore describes about 59 percent off all records obtained from combining the tables, “Employees” and “Salaries”. When further analyzed this selection can be an indication that a concept “Employees who obtained a change of salary at the age of 19” can be of value. When turned into a ontology concept it can be described as:

```
(has_birth_date some xsd:date[>="1957-05-15"^^xsd:date])
and (has_salary some (Salary and has_to_date some
xsd:date [>="1986-05-16"^^xsd:date]))
```

The next best selection applies to records where employees became department manager after 1994-06-25 and had a change of title before 1996-05-30. This indicates one shortcoming of the developed algorithm. These two attributes are related, because when an employee becomes a department manager, his title changes accordingly. This way these records bypass be implemented restriction on short rules, based on only one attribute.

6 Conclusions and Future Work

This paper investigated the use of a genetic algorithm based approach for the generation of random selections in a database for the purpose of creating ontology concepts. The presented approach has shown some potential to be used as a resource for assisted ontology concept creation. When a knowledge engineer has the task of creating concepts for a completely new domain, analysis of the data can be helpful. The presented approach is capable of generating random queries, which (with some few exceptions) are guaranteed to be relevant to a large part of records. It is due to this and the ability of the algorithm to create many different selections that provide assistance to the ontology building process.

The approach described in this paper is not purely a clustering algorithm, nor does it try to be an alternative to real clustering algorithms. These are many possible approaches for actual clustering. This paper is merely an investigation towards random concept generation using these methods. The presented approach has also many shortcomings, when compared to classic data mining approaches, resulting from the restrictions due to the database access. Even with the implemented evaluation process, which is multiple times faster than in depth analysis of the data, the average time of generating random queries and improving them with the genetic algorithm over 100 generation is still 4 h, for this example database.

The added distance calculations to the fitness function cause the algorithm to become less stable. This is due to individuals getting worse in the next generation, when they did nothing to change. The overall trend does indicate improvement even with the jumps in cumulative fitness. This was the only way of insuring heterogeneity of the selections without doing in-depth analysis of the data points belonging to the selection.

As the obtained concepts show, this approach or an approach similar to this one can be used to help generate concepts from a database. This approach does not guarantee to generate truly unique or important concepts, but they can be helpful to a knowledge engineer who is trying to create a completely new ontology for an unknown domain.

Acknowledgements. The Faculty of Computer Science and Information Technology (Riga Technical University) funds this work by an assigned Doctoral studies grant (3412000-DOK. DITF).

References

1. Nicola, A., Missikoff, M., Navigli, R.: A software engineering approach to ontology building. Inf. Syst. **34**(2), 258–275 (2009). doi:10.1016/j.is.2008.07.002
2. Chemchem, A., Drias, H.: From data mining to knowledge mining: application to intelligent agents. Expert Syst. Appl. **42**(3), 1436–1445 (2015). doi:10.1016/j.eswa.2014.08.024
3. Gullo, F.: From patterns in data to knowledge discovery: what data mining can do. Phys. Procedia **62**, 18–22 (2015). doi:10.1016/j.phpro.2015.02.005
4. Gorskis, H., Cižovs, J.: Ontology building using data mining techniques. Inf. Technol. Manag. Sci. **15**, 183–188 (2012). doi:10.2478/v10313-012-0024-5

5. Djellali, C.: A new data mining system for ontology learning using dynamic time warping alignment as a case. Procedia Comput. Sci. **21**, 75–82 (2013). doi:10.1016/j.procs.2013.09.012
6. Luther, S., Berndt, D., Finch, D., Richardson, M., Hickling, E., Hickam, D.: Using statistical text mining to supplement the development of an ontology. J. Biomed. Inf. **44**, S86–S93 (2011). doi:10.1016/j.jbi.2011.11.001
7. Song, E., Hua Li, C., Cheol Park, S.: Genetic algorithm for text clustering using ontology and evaluating the validity of various semantic similarity measures. Expert Syst. Appl. **36**(5), 9095–9104 (2009). doi:10.1016/j.eswa.2008.12.046
8. Amar, F.B., Gargouri, B., Hamadou, A.: Generating core domain ontologies from normalized dictionaries. Eng. Appl. Artif. Intell. **51**, 230–241 (2016). doi:10.1016/j.engappai.2016.01.014
9. Li, H., Sima, Q.: Parallel mining of OWL 2 EL ontology from large linked datasets. Knowl.-Based Syst. **84**, 10–17 (2015). doi:10.1016/j.knosys.2015.03.023
10. Gorskis, H., Borisovs, A.: Ontology building using classification rules and discovered concepts. Inf. Technol. Manag. Sci. **18**, 37–41 (2015). doi:10.1515/itms-2015-0006
11. Maulik, U., Bandyopadhyay, S.: Genetic algorithm-based clustering technique. Pattern Recogn. **33**(9), 1455–1465 (2000). doi:10.1016/S0031-3203(99)00137-5
12. The MySQL “Employees” example database’s homepage: https://dev.mysql.com/doc/employee/en/. Accessed 3 Apr 2016

Improving Artificial Fish Swarm Algorithm by Applying Group Escape Behavior of Fish

Seyed Hosein Iranmanesh[1,2](✉), Fahimeh Tanhaie[1], and Masoud Rabbani[1]

[1] Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
{hiranmanesh, fahimeh.tanhaie, mrabani}@ut.ac.ir
[2] Research Institute for Energy Planning and Management (RIEMP), Tehran, Iran

Abstract. Artificial fish swarm algorithm is a technique based on swarm behaviors that are inspired from schooling behaviors of fishes swarm in the nature. Group escaping is another interesting behavior of fish that is ignored. This behavior shows all fish change their moving directions rapidly while some fish sense a predator. In this paper, we proposed a new algorithm which is obtained by hybridizing artificial fish swarm algorithm and group escaping behavior of fish which can greatly speed up the convergence. It is presented proper pseudocode of improved algorithm and then experimental results on Traveling Salesman Problem is applied and demonstrated the advantages of the improved algorithm.

Keywords: Swarm behaviors · Artificial fish swarm algorithm · Group escaping behavior

1 Introduction

Artificial fish swarm algorithm (AFSA), is a class of swarm intelligence optimization algorithm based on swarm behaviors that are inspired from social behaviors of fishes swarm in the nature. In a water area, fishes have a self-organizer behavior which enable them to move around their environment with no need to leader. They desire to stay close to the swarm and are most likely distributed around the region where foods are most abundant. From investigation [11], it is obtained that the three most common behaviors fishes will exercise in a school are

– Swarming, gregarious fishes tend to focus towards each other while avoiding overcrowding.
– Following, the behavior of chasing the closest buddy.
– Preying, fishes tend to head towards food.

AFSA, which was proposed in Li [10], emulates the above three basic social behaviors of fish. This algorithm has many advantages, including insensitivity to initial values, strong robustness and simplicity in implementation. This algorithm has been used in applications of optimization such as data mining [17], data clustering [9], image registration [14], PID control [16], and so on. However, the convergence rate of AFSA

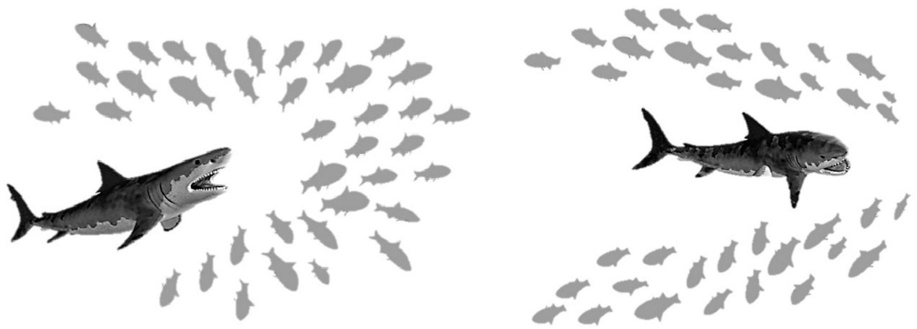


Fig. 1. Group escape behavior of fish schools

algorithm is slow and easy to fall into local optimal solution. Another interesting behavior of fish schools is group escape behavior which is ignored in most of the researches. In order to improve the optimization performance of the AFSA, we proposed a new method based on group escaping behavior of fish. The group escape behavior in fish is very typical and when predator or other stimulus is discovered by one or some fish in the school, the whole school changes the direction very quickly. This behavior is shown in Fig. 1.

2 Literature Review

Literature review showed that researchers have made some attempts to improve the performance of AFSA. In order to increase the diversification of the artificial fishes based on their parents' characteristics, Wu et al. [6] presented Artificial Fish Swarm optimization algorithm with crossover, CAFAC. In this algorithm, the crossover operator is first explored. Numerical results demonstrated that the proposed method can outperform the original AFSA. Jiang and Yuan [8] increased the search quality of AFSA by discussing the possibility of parallelization of AFSA and making the analysis of parallel AFSA.

Some researchers have tried to combine different algorithm with AFSA and improve the performance of AFSA. Huadong Chen et al. [2] presented a hybrid algorithm to train forward neural network using a hybrid of particle swarm optimization and AFSA. The proposed method was more effective than AFSA and PSO. The hybrid in the proposed algorithm not only has the artificial fish behaviors of swarm and follow, but also takes advantage of the information of the particle.

Belacel et al. proposed an AFSA with adaptive visual to improve the performance of fuzzy clustering [7]. In this paper AFSA enhances the performance of the fuzzy C-Means (FCM) algorithm. A numerical result showed the advantages of the proposed algorithm. Also, Chen et al. [3] reported a hybrid algorithm by using the characteristics of AFSA and Chaos Optimization Algorithm. This method is an efficient global optimization algorithm for solving global optimization problem. In this approach, adding chaos is suitable for updating the velocities of artificial fish and improving the convergence rate and the accuracy.

Edite et al. [5] presented a new method with a set of movements, closely related to the random, the searching and the leaping fish behaviors. This algorithm tested on a set of seven benchmark problems. In [15], a new algorithm is presented for optimization in static and continuous environments by hybridizing AFSA and cellular learning automata. Experimental results showed that the proposed algorithm has an acceptable performance. In order to improve the optimization performance of the AFSA, many researchers have presented some improved algorithm. Although all the proposed algorithms can improve the performance of the original algorithm, but it still cannot get satisfactory results for some problems. In this paper, we proposed a new method based on the group escape behavior of fish that is ignored in most of the researches. There are some researches on observing group escape behavior of fish school biologically [13], but there is no research on applying on artificial fish swarm algorithm.

When the predator comes very close to a fish, the fish chooses the group escape behavior and if there is no predator in the visible area of the fish, the fish moves using its schooling behavior.

3 Proposed AFSA with Group Escape Behavior

In this section an artificial fish swarm algorithm based on group escape behavior is proposed which in that both ability of local search and global search to standard AFSA has been increased and improved. Improved AFSA is a random search algorithm based on simulating fish swarm behaviors which contains group escape behavior, preying behavior, swarming behavior and following behavior. It constructs the simple behaviors of artificial fish firstly, and then makes the global optimum appear finally based on animal individuals' local searching behaviors.

Before explaining the improved AFSA, we introduce some definitions [4, 10, 12]:

Assuming in an n-dimensional searching space, there is a group composed of n articles of artificial fish (AF). Situation of each individual AF can be expressed as vector $x = (x_1, x_2, x_3, \ldots, x_n)$ is denoted the current state of AF, where X_n is control variable.

- $f(x)$ is the fitness or objective function of X, which can represent food concentration of AF in the current position.
- $Dist_{ij} = \|X_i - X_j\|$ is denoted the Euclidean distance between fishes.
- Visual and Step are denoted respectively the visual distance of AF and the distance that AF can move for each step.
- X_v is the visual position at some moment. If the state at the visual position is better than the current state, it goes forward ad step in this direction, and arrives the X_{next} state, otherwise, continues an inspecting tour in the vision.
- Try-number is attempt times in the behavior of prey.
- δ is the condition of jamming $(0 < \delta < 1)$.

Supposed X_v is the visual position at some moment and X_{next} is the new position. The movement process is represented as:

$$X_v = X_i + Visual \times Rand(0,1)$$

$$X_{next} = X + \frac{X_v - X}{\|X_v - X\|} \times Step \times Rand(0,1)$$

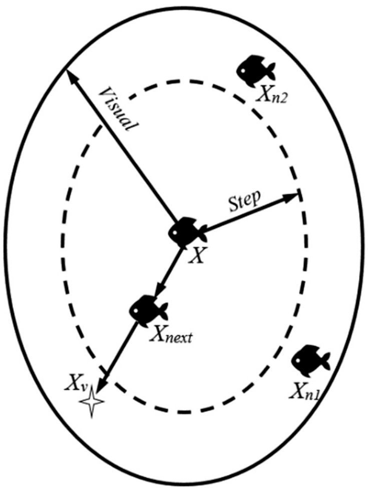


Fig. 2. The movement process of fish

Where rand () produces random numbers between 0 and 1. This behavior is shown in Fig. 2.

It is well known that most of fish schools have a group behavior: group escaping. When predator is discovered by one or more fishes in the school, the whole school changes the direction very quickly. In the proposed algorithm, when an artificial fish could not find better situation in food prey behavior or in swarm behavior with doing a freely movement, this situation saves in problem space. So, other fishes sense this situation as an enemy and change their direction with high speed and moving away from it. We see this group escape behavior as follows:

When the fish discover a water area with more food, they will go quickly toward the area. Consider the state of artificial fish is X_i, Select a state X_j within its sensing range randomly. If $f(X_i) > f(X_j)$, the new situation is not better than current situation and this fish has to informs other fish, so we save this situation as forbidden zone in problem space to prevent other fishes going there.

Forbidden Zone:

$$\left\{X_j = X_i + Visual \times Rand(-1, 1),\ if\, f(X_i) > f(X_j), for\ each\ i\ and\ j\right\}$$

With this introduction, the basic behaviors of improved AFSA are defined as follows (the three most common behaviors of fishes like preying, swarming and following are from the standard AFSA [10]):

Prey Behavior

In general, when the fish discover a water area with more food, they will go quickly toward the area. Consider the state of artificial fish is X_I, Select a state X_j within its sensing range randomly. If $f(X_i) \leq f(X_j)$ and X_j is not in forbidden zone, then fish move to X_j. On the contrary, select randomly state X_j that is not in forbidden zone and determine whether to meet the forward conditions, repeat several time, if still not satisfied forward conditions, then move one step randomly.

$$X_j = X_i + Visual \times Rand(-1, 1)$$

If $f(X_i) \leq f(X_j)$ and X_j is not in forbidden zone, it goes forward a step as follows:

$$\bar{X}_i(t+1) = \bar{X}_i(t) + \frac{\bar{X}_j - \bar{X}_i(t)}{Dist_{i,j}} \times Step \times Rand(0, 1)$$

If $f(X_i) > f(X_j)$, we add this situation to forbidden zone in problem space to prevent other fishes going there.

Updating Forbidden Zone:

$$\{X_j = X_i + Visual \times Rand(-1, 1),\ if f(X_i) > f(X_j), for\ each\ i\ and\ j\}$$

Swarm Behavior

In the process of swimming, the fish will swarm in order to share the food of the swarm. Supposed the current state of artificial fish is X_i, number of artificial fish is n and number of neighbors around the artificial fish X_{center} is n_c, if $\delta > (n_c/n)$ indicates that the partners have more food and less crowded, if $f(X_{center}) \geq f(X_i)$ and X_{center} is not in forbidden zone, then go forward toward the center of the direction of the partnership, otherwise prey behavior.

$$X_{center} = \frac{1}{n}\sum\nolimits_{i=1}^{n} X_i$$

If $f(X_{center}) \geq f(X_i)$ and X_{center} is not in forbidden zone, it goes forward a step as follows:

$$X_i(t+1) = X_i(t) + \frac{X_{center} - X_i(t)}{\|X_{center} - X_i(t)\|} \times Step \times Rand(0, 1)$$

If $f(X_{center}) < f(X_i)$, we add this situation to forbidden zone in problem space to prevent other fishes going there.

Updating Forbidden Zone:

$$\left\{X_j = X_{center} = \frac{1}{n}\sum\nolimits_{i=1}^{n} X_i,\ if f(X_{center}) < f(X_i), for\ each\ i\right\}$$

Follow Behavior

When one fish of the fish swarm discovers more food, the other fish will share with it. Supposed the state of artificial fish is X_i, explore its optimal state X_n from Visual neighbors, the number of partners of X_n is n_n and the number of artificial fish is n, If $\delta > (n_c/n)$ indicates that near distance have more food and not too crowded, further move to the front of X_n position; otherwise perform foraging behavior.

$$X_i(t+1) = X_i(t) + \frac{X_n - X_i(t)}{\|X_n - X_i(t)\|} \times Step \times Rand(0,1)$$

Taking into account the above mentioned behaviors, the pseudocode of improved AFSA can be written as follows:

```
begin
for each Artificial Fish  $i \in [1 \ldots N]$
     Initialize $X_i$          such as Step, Visual, the number of
exploratory try number;
endfor

Blackboard -$^{best}_{X_i} f(X_i)$          to record the current status of
each fish and select the optimal value recorded;

Repeat:
     for each Artificial Fish $i \in [1 \ldots N]$
   Save forbidden zone     based on group escaping behavior
            Perform prey Behavior on $X_i(t)$ and compute $X_{i,prey}$
   Update forbidden zone    based on group escaping behavior
            Perform Swarm Behavior on  $X_{i,prey}(t)$ and compute
$X_{i,swarm}$
   Update forbidden zone          based on group escaping
behavior
            Perform Follow Behavior on $X_{i,prey}(t)$  and compute
$X_{i,follow}$
            If $f(X_{i,swarm}) \le f(X_{i,follow})$
                Then $X_i(t+1) = X_{i,follow}$
            else
                $X_i(t+1) = X_{i,swarm}$
        endfor
            If $f(X_{Best-AF}) \ge f(Blackboard)$
                Then $Blackboard = X_{Best-AF}$      Optimal value in
Blackboard is updated;
Until stopping criterion is met
end
```

Therefore, in the improved AFSA other fishes sense unsuitable situation as a forbidden zone and change their direction with high speed and moving away from it, so, the ability of local search and global search to standard AFSA has been increased.

4 Experimental Results

To test the effectiveness of the proposed algorithm, we solve Traveling Salesman Problem (number of cities: 30 and 45) as a primary step with improved AFSA and standard AFSA. Traveling salesman problem (TSP) is a classic NP-hard problem which can be simply described as: Given n cities and the distance between cities, find a visit to the city once and only once the shortest path [1]. The parameters for artificial fish school algorithm are set as follows. The number of artificial fishes is set to N = 20, with the number of attempts a fish will try in prey behavior is set to try number = 5. The maximum visual distance of a fish is 4.5, and the maximum step size that a fish can make in one movement is step = 0.3. The maximum allowed iteration loops for artificial fish school algorithm to run in the optimization process is set for 2000 iterations.

We use MATLAB 13.0 to run the algorithm, the test results are run on a PC with Intel (R) Core (TM) 2 Duo CPU T9550 @ 2.67 GHz and 4 GB of memory. Do 20 tests for each instance and compare the result of improved AFSA with standard AFSA. The experimental results are in Table 1.

As can be seen from Table 1, the best and average solution of the improved algorithm are better than standard AFSA.

Table 1. The experimental results

Algorithms	Number of cities: 30			
	Total distance (objective function)		Time	
	Mean	Best	Mean	Best
AFSA	143.48041	136.72	166.76941	127.593451
Improved AFSA	141.42411	126.993	159.68995	124.550108
Algorithms	Number of cities: 45			
	Total distance (objective function)		Time	
	Mean	Best	Mean	Best
AFSA	219.53829	211.3384	293.86103	267.737987
Improved AFSA	205.64594	195.9627	280.5638	250.239505

5 Conclusion

AFSA has many advantages, including insensitivity to initial values, strong robustness and simplicity in implementation and so on. However, the convergence rate of AFSA algorithm is slow and easy to fall into local optimal solution. In addition, the experiences of group members are not used for the next moves.

In order to improve the AFSA algorithm and the ability to speed up the convergence rate of the AFSA algorithm, we proposed an improved algorithm which is obtained by hybridizing artificial fish swarm algorithm and group escaping behavior of fish that is ignored in most of the researches. The group escape behavior in fish is very typical and when predator or other stimulus is discovered by one or some fish in the school, the whole school changes the direction very quickly. Experimental results showed that proposed algorithm is of better efficiency than standard AFSA and the time of running the algorithm demonstrated that the convergence rate of the improved algorithm is considerable improvement than standard AFSA. In future works, we will apply the proposed algorithm on different problems to develop the result.

Acknowledgements. The authors would like to thank the anonymous reviewers of this paper for their thought-provoking and insightful comments and corrections.

References

1. Applegate, D., et al.: The Traveling Salesman Problem. Princeton University Press, Princeton (2011)
2. Chen, H., Li, S., Li, Y.: A hybrid of artificial fish swarm algorithm and particle swarm optimization for feedforward neural network training. IEEE Adv. Intell. Syst. Res. (2007). https://www.researchgate.net/publication/264887635
3. Chen, Z., Tian, X.: Artificial Fish-Swarm algorithm with chaos and its application. In: 2010 Second International Workshop on Education Technology and Computer Science, pp. 226–229 (2010)
4. Wang, C.-R., Zhou, C.-L., Ma, J.-W.: An improved artificial Fish-Swarm algorithm and its application in feed-forward neural networks. In: 2005 International Conference on Machine Learning and Cybernetics, vol. 5, pp. 2890–2894 (2005)
5. Fernandes, E., Martins, T., Maria, A.: Fish swarm intelligent algorithm for bound constrained global optimization. In: Proceedings of the International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE (2009). http://www.norg.uminho.pt/emgpf/documentos/cmmse_09_FMR.pdf
6. Gao, X., Wu, Y., Zenger, K., Huang, X.: A knowledge-based artificial Fish-Swarm algorithm. In: 2010 13th IEEE International Conference on Computational Science and Engineering. pp. 327–332 (2010)
7. He, S., Belacel, N., Hamam, H., Bouslimani, Y.: Fuzzy clustering with improved artificial fish swarm algorithm. In: 2009 International Joint Conference on Computational Sciences and Optimization, pp. 317–321 (2009)
8. Jiang, M., Yuan, D.: Parallel artificial fish swarm algorithm. In: Zeng, D. (eds.) Advances in Control and Communication. LNEE, vol 137, pp. 581–589. Springer, Heidelberg (2012)
9. Xiao, L.: A clustering algorithm based on artificial fish school. In: 2010 2nd International Conference on Computer Engineering and Technology, pp. V7-766–V7-769 (2010)
10. Li, X.: A new intelligent optimization method—artificial fish school algorithm. Doctoral thesis, Zhejiang University (2003, unpublished)
11. Li, X., Shao, Z., Qian, J.: An optimizing method based on autonomous animals: Fish-Swarm algorithm. Syst. Eng. Theor. Pract. **22**, 32–38 (2002)
12. Luo, Y., Zhang, J., Li, X.: The optimization of PID controller parameters based on artificial fish swarm algorithm. In: 2007 IEEE International Conference on Automation and Logistics, pp. 1058–1062 (2007)
13. Parrish, J.K., Viscido, S.V., Grunbaum, D.: Self-organized fish schools: an examination of emergent properties. Biol. Bull. **202**, 296–305 (2002)

14. Wang, Y., Zhang, W., Li, H.: Application of artificial fish swarm algorithm in image registration. Comput. Model. New Technol. **18**, 510–516 (2014)
15. Yazdani, D., Golyari, S., Meybodi, M.: A new hybrid algorithm for optimization based on artificial fish swarm algorithm and cellular learning automata. In: 2010 5th International Symposium on Telecommunications, pp. 932–937 (2010)
16. Luo, Y., Wei, W., xin Wang, S.: Optimization of PID controller parameters based on an improved artificial fish swarm algorithm. In: Third International Workshop on Advanced Computational Intelligence, pp. 328–332 (2010)
17. Zhang, M., Shao, C., Li, M., Sun, J.: Mining classification rule with artificial fish swarm. In: 6th World Congress on Intelligent Control and Automation, pp. 5877–5881 (2006)

Genetic Programming Algorithm Creating and Assembling Subtrees for Making Analytical Functions

Tomáš Cádrik(✉) and Marián Mach

Department of Cybernetics and Artificial Intelligence,
Technical University of Košice, Letná 9, 042 00 Košice, Slovakia
{tomas.cadrik,marian.mach}@tuke.sk

Abstract. There are many optimization algorithms which can be used for solving different tasks. One of those is the genetic programming method, which can build an analytical function which can describe data. The function is coded in a tree structure. The problem is that when we decide to use lower maximal depth of the tree, the genetic programming is not able to compose a function which is good enough. This paper describes the way how to solve this problem. The approach is based on creating partial solutions represented by subtrees and composing them together to create the last tree. This approach was tested for finding a function which can correctly calculate the output according to the given inputs. The experiments showed that even when using a small maximal depth, the genetic programming using our approach can create functions with good results.

Keywords: Analytical function · Assembling subtrees · Genetic programming · Partial trees

1 Introduction

Genetic programming [1] (GP) is an optimization method, which can be used for creating an analytical function, which can be used for classification of input data to some class, can be used for prediction or as a program controlling some processes. For instance, GP was used for creating a behavior which was able to control an agent in a computer game, for evolving filters for image processing, etc.

The problem with this method is when we need to describe data which relation is very complicated. Then it is complicated to set the parameters of genetic programming if we do not have an adaptation mechanism like in [2]. Or the tree representing the solution can, in that case, grow very fast and the calculation time will increase. It is unacceptable when we want to use it in our research which is focused on cloud robotics [3].

We are developing a system, which will be used as a support for learning for robots. This system will be based on cloud computing [4]. The cloud will serve as a learning module, so the robot will be able to spare energy for other important tasks. The learning modules will be created as so-called AI bricks [5] which will be accessible from all over the world and will be usable not only for robotics but also in other fields where data analysis is needed. If the calculation takes too much time, it will be a problem if the cloud service receives too many requests. The users who need the solution will be waiting for it.

Another problem is that if the solution is complicated, it is hard for the GP to find it. Because of this reason we were trying the approach which is creating the solution from smaller trees. This methodology is inspired by the research done previously and will be mentioned in one of the next sections.

The paper is organized as follows: in the second section are the basics of GP described. Section 3 describes the previous research of the authors, which shows the basics to the approach done and is presented in this paper. Section 4 describes the way how GP can find the solution by parts assembling the created subtrees. Section 5 shows some experiments done with the method. Section 6 concludes the paper, the experiments and describes the future work according to this research.

2 Genetic Programming

GP is a method based on a family of optimization algorithms called evolutionary algorithms [6]. Those methods are using principles of evolution when searching in the candidate space for optimal solutions. The searching is based on a fitness function which determines the quality of that concrete solution. EA are population algorithms, which means that it is using a set of solutions at once for searching the candidate space. That set is called the population of individuals.

The creation of new individuals is done using so-called genetic operators. Those operators can be asexual, sexual or panmictic. The asexual operator is used to adding a random building block to an individual. Sexual and panmictic operators create a new individual using two respectively more old individuals. The individuals who are selected for the genetic operators are called parents, the individuals created with the application of genetic operators are called children. The parents are always selected pseudo stochasticly; that means that the selection is based on the fitness of the individual but adding some randomness. In each iteration, a set of new individuals is created, and some of them can survive and be in the new population. The solution from the search is the one represented by the best-found individual.

GP is an EA where a solution is represented by a tree made of operators, variables, and constants. The operators can be simple ones as a plus, minus, absolute value, sinus, etc. or more complex ones as condition whether values are equal. The root element can be a function, constant or variable, but leafs of the tree can be only constants and variables. The result from the tree is calculated recursively. Each function returns an according to the children. For instance, if the function is plus, the result is the sum of his children. So the calculation needs to go from the bottom of the tree, where are only variables and constants. So the constant can return itself and the variable a value represented by that variable. Function return the solutions applied to children and the solution from the whole tree is the result from the root node. The variables are the connection with the external world. For instance, if some system has ten inputs, the tree of the individual can contain ten variables. The goal of the GP is to find a tree (which is representing an analytical function) which will be able to approximate the behavior of

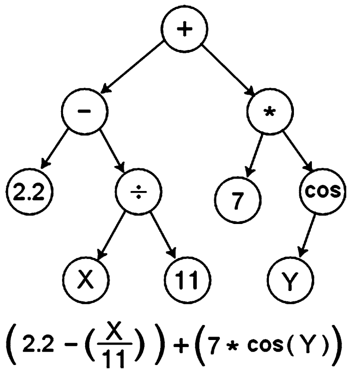


Fig. 1. An example of a tree representing the given equation

some system. Not all variables need to be presented in the tree. If it is not presented, that means that the final result from the tree is independent of the tree. An example of a tree with two variables created by GP is showed in Fig. 1.

GP often uses two genetic operators. The first one takes the tree and selects one node. Then it recursively changes this node to another one and also changes recursively the children, and them children, etc. The result from this operator is tree similar to the previous one but with a changed subtree. The second operator is the crossover, which uses two trees. The first step is the selection of one node in the first tree. After that, a node in the second tree of the same depth as the first one is selected. The last step is the exchange of those two selected nodes between the trees, which will make the new tree.

3 Interference of Waves Based Method

The interference of waves based method (IOW) was introduced in [7]. This method uses parallelization techniques which do not use one algorithm as in [8] but it uses multiple algorithms. This method was using a parallel technic which uses multiple optimization algorithms. Those algorithms can be the same, but also can have different parameters, can search different candidate spaces or can have different fitness functions. The solution was the best find solution or a combination of all solutions from all algorithms. The diagram showing how IOW works is located in Fig. 2. In this paper, we are using only one algorithm, but we are focusing on another feature of IOW.

Another system is the wave “composing”. From this mechanism came the name of this usage of optimization algorithms. That means when two waves meet, the wave created from this state can increase according to the first one, but it can also decrease or disappear. The first wave is the previously found parts and the second wave is represented by the solution from the actual cycle.

This mechanism is arranging the search which is composing the solution part by part. After first search process, the algorithm returns first solution (the best one it found). This solution is saved to the actual solution from the algorithm. Then the algorithm will be restarted, and another optimization process will start with that

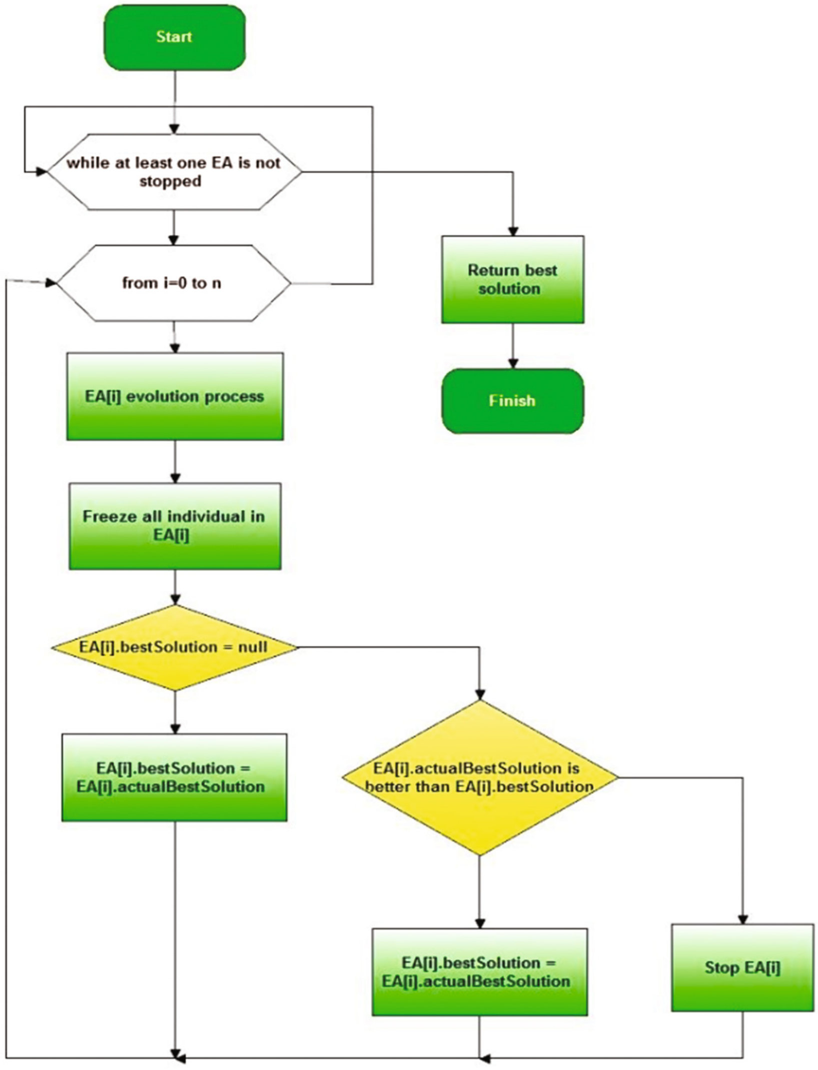

Start
while at least one EA is not stopped
from i=0 to n
Return best solution
Finish
EA[i] evolution process
Freeze all individual in EA[i]
EA[i].bestSolution = null
EA[i].bestSolution = EA[i].actualBestSolution
EA[i].actualBestSolution is better than EA[i].bestSolution
EA[i].bestSolution = EA[i].actualBestSolution
Stop EA[i]

Fig. 2. The diagram of the interference of waves based method from [7]

optimization algorithm. The fitness of each solution from the algorithm is then calculated not only from the solution of the actual search, but it uses a combination of the solution found after previous search process of the algorithm and the one from this cycle. That means that it is testing which solution is good when merging with the part found in the previous process.

When the search process ends, the IOW has two possible ways how to continue. The first one is that the fitness of the merged parts and the actual solution from the algorithm (the best one found in the last search process) is better as it was after the previous search. Then the solution will be saved to the previously found parts, and the search will continue. If the whole solution is worse as it was after the previous search, the algorithm is frozen. When we have multiple algorithms in the IOW, it stops when all algorithms are frozen. The freezing method can also have two variations [9] – local best and global best. On this will depend on whether the parts of the solution are from the algorithm best found a solution or the best from all algorithms. Because we were using only one algorithm, this is irrelevant for our experiments.

4 Genetic Programming Creating and Composing Subtrees Together

Until now the IOW was used only with the structures of real or integer values created from the optimization algorithms. So it was relatively easy to compose the final solution. The problem was, how can be the final solution composed when the algorithm is creating subtrees.

There were some researches done which used the operations with subtrees. For instance, in [10, 11] was the swapping technique of subtrees described. Also in [12] was a method which used operations with subtrees for approving the genetic programming technique. But the possibility of building the final solution using the subtrees and composing them together was not described yet (or only a few types of researches work with this).

So we created a way how the sub-solutions were composed to the final solution. The GP, which is presented in this paper, uses the IOW mechanism described in the previous section. So the solution from the search process is only a part of the whole solution. The biggest problem was how to fuse the smaller solutions to one representing the complete tree. In this paper, two methods are described.

The first method uses the last tree as an input for the next created tree. Let have a tree from the first search process. This tree is representing a function $T_1(x_0, x_1, \ldots, x_n)$, where x_0 to x_n are variables which are representing the inputs from the external world. The solution for the concrete input values is r. The tree in the second search process can not only use the variables same from T_1. But it can also use r. The solution from the newly created tree will be the next r. The whole process of calculating output values is showed in the following equations.

$$T_1(x_0, x_1, \ldots, x_n) \rightarrow r \tag{1}$$

$$T_2(x_0, x_1, \ldots, x_n, r) \rightarrow r \tag{2}$$

$$T_{final}(x_0, x_1, \ldots, x_n, r) \rightarrow r_{final} \tag{3}$$

A graphical example is showed in Fig. 3.

Where T_1 is the function represented by the first tree, T_2 is the function represented by the second tree; T_{final} is the function represented by the last tree before IOW stops the search process and r_{final} is the result from the last function.

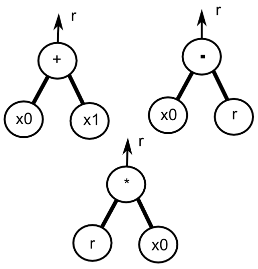


Fig. 3. An example of the first method how to compose the solution from multiple subtrees

The second method uses also uses the previous ones. That means if the result from the first tree is equal r_1, the second tree can use all the variables and r_1. The third tree can use not only the result from T_2 but also result from T_1. The calculation of results from the trees is shown in the next equations.

$$T_1(x_0,\ x_1,\ldots,\ x_n) \rightarrow r_1 \tag{4}$$

$$T_2(x_0,\ x_1,\ldots,x_n,\ r_1) \rightarrow r_2 \tag{5}$$

$$T_{\text{final}}(x_0,\ x_1,\ldots,x_n,\ r_1,\ r_2,\ldots,\ r_{\text{final}-1}) \rightarrow r_{\text{final}} \tag{6}$$

An example of the second method is showed in Fig. 4.

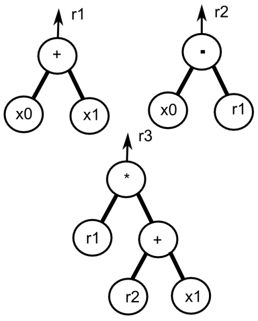


Fig. 4. An example of the second method how to compose the solution from multiple subtrees

The advantage of the second approach is that it can use results from all founded subtrees. That means that if we find a good subtree with the first method which can be used few times in the final tree, we will be not able to do this because it will be lost after following calculations. On the other site, the second methods can return to that subtree any time because it is using it as one input for the current tree.

The disadvantage of the second method according to the fist method is that the number of input values increases after each search process. The first method has always after first searching process only all variables and the result from the previous tree.

5 Experiments

The experiments were done on two functions. First random combinations of input values were generated from the range where the function is defined. Then the function was used for calculating the outputs. So the dataset contained the combinations of inputs and outputs. This dataset served as the training set. The goal for genetic programming was to find a function which was approximating the function used for generating the training set (where the calculated outputs were similar to the output values using the equation with the input values). Those two equations (functions) were the average of input values and the so-called Ackley function. The parameters were the same in both experiments. They are listed in the next table (Table 1).

Table 1. Parameters used in experiments

Parameter name	Value
Number of cycles	1000
Number of individuals	200
Mutation probability	0.15
Number of survived individuals from the old population	100
Selection	5-ary tournament

We chose that we want only to use operators +, −, *, /, sqrt. The leaf of the tree can be a variable or a constant number from 0 to 4. We have done experiments where the depth was equal to four and five, where the root node of the tree had the depth equal to zero. We had chosen the error of the equation represented by the individual as the fitness value. The error was calculated as the average difference between the wanted output value and the one calculated with the equation represented by the individual with the input values.

We tried all modes of searching. The first mode is the standard GP. Mode 2 is the mechanism from Fig. 3. Mode 3 is the mechanism from Fig. 4. We do not want to find the best solution. The goal of our experiments was to show if the approach which is presented in this paper is better than the classical approach (standard genetic programming algorithm when using mode 1 in IOW). So whether we can use smaller trees to compose the solution.

The first experiment showed the results where the training set was generated using the equation showed in (7).

$$f(x_0, x_1, \ldots, x_n) = \frac{\sum_{k=0}^{n} x_k}{n} \tag{7}$$

The input values were integers from the range 0–100. The results are shown in Table 2.

As we can see from the table, our two approaches presented in this paper were better than the standard method (mode 1). Because we have input values from zero to 100, the outputs can also be from this range. We used a depth and possible functions

Table 2. The experiments were done on the equation calculating the average of inputs

Maximal depth	Mode	Error
4	1	6.7625641679856
4	2	1.80705486742458
4	3	1.097680437297865
5	1	3.29645923092001 1
5	2	2.094530230435445
5	3	1.36136581089146

which can be used by GP where it is impossible to compose the final tree. And as we can see, that our methods were able to create an equation which can be used for calculating outputs with good quality. The third mode where we can use any so far composed subtree as a possible input seems to be the best for creating the whole solution.

The next experiments were done on the Ackley function [13]. The randomly generated input values were from the range [−32.768, 32.768]. The outputs from the training set were again calculated according to the equation.

The results are shown in the next table (Table 3).

Table 3. The experiments were done with the Ackley function

Maximal depth	Mode	Error
4	1	10.34525940557325
4	2	1.269621673460215
4	3	0.0033423967614733
5	1	9.11948389381155
5	2	1.039567213981151
5	3	0.176521851185443

As the experiments show, our method can also solve more complicated problems, like approximating a function from data created by the use of the Ackley function. We can also see, that the maximal depth of the tree which can be generated by the GP is not affecting the final results. They are affecting the standard GP, where the results are better but in the other case, when using our approach, the results are the same, or they are worse.

6 Conclusion

As the experiments show, the method which was creating solutions from subtrees was better than the classical use of GP. We can use lower depth, and it can find the solution with high quality. Also, we do not need to focus on the analysis of the given data, and we can simply add only simple functions, and the method can develop the equation.

When comparing the two developed mods we found out that the one where the method can use all so far found subtrees is better. It is because it can make combinations from those subtrees instead of using only the one which was developed in the last searching cycle. It is because there is a possibility that the last created subtree can be the best comparing to the previous one, but it can be worse in the global meaning.

In the future, we want to create an AI brick from this method. The goal of our research shortly is also to develop the Cartesian genetic programming method and apply this approach to it, to see if we can increase the calculation quality of that method. We also want to use this approach for creating a behavior for a robot from data obtained from teleoperation in human-robot interaction.

Acknowledgement. Research supported by the National Research and Development Project Grant 1/0773/16 2016 – 2019 “Cloud Based Artificial Intelligence for Intelligent Robotics”.

References

1. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
2. Riekert, M., Malan, K.M., Engelbrect, A.P.: Adaptive genetic programming for dynamic classification problems. In: Proceedings of IEEE Congress on Evolutionary Computation, CEC 2009, pp. 674–681 (2009)
3. Lorencik, D., Sincak, P.: Cloud robotics: current trends and possible use as a service. In: 2013 IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 85–88 (2013)
4. Mell, P., Grace, T.: The NIST definition of cloud computing recommendations of the National Institute of Standards and Technology. NIST Spec. Publ. **145**, 7 (2011)
5. Cádrik, T.: A cloud-based multi-robot system. Dissertation thesis proposal (2015)
6. Mach, M.: Evolutionary Algorithms: Elements and Principles, p. 250. Elfa, Košice (2009)
7. Cádrik, T., Mach, M., Sinčák, P.: Interference of waves based usage of an optimization algorithm for finding rules in an agent system. In: SCIS and ISIS 2014, pp. 1068–1072. IEEE, Danvers (2014). ISBN 978-1-4799-5954-9
8. Welten, S.: Parallelization of Evolutionary Algorithms. Swiss Federal Institute of Technology Zurich, 62 p. (2008)
9. Cádrik, T., Mach, M.: The method based on interference of waves for solving real value optimisation problems. Mendel 2015, pp. 27–30. University of Technology, Brno (2015). ISSN 1803-3814
10. Poli, R., McPhee, N.F.: General schema theory for genetic programming with subtree-swapping crossover: part I. Evol. Comput. **11**(1), 53–66 (2003)
11. Poli, R., McPhee, N.F.: General schema theory for genetic programming with subtree-swapping crossover: part II. Evol. Comput. **11**(2), 169–206 (2003)
12. Muntean, O., Diosan, L., Oltean, M.: Solving the even-n-parity problems using Best SubTree Genetic Programming. In: Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007), pp. 511–518 (2007)
13. Ackley function. http://www.sfu.ca/~ssurjano/ackley.html

Comparison of Parallel Linear Genetic Programming Implementations

David Grochol(✉) and Lukas Sekanina

Faculty of Information Technology, IT4Innovations Centre of Excellence, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic
{igrochol,sekanina}@fit.vutbr.cz

Abstract. Linear genetic programming (LGP) represents candidate programs as sequences of instructions for a register machine. In order to accelerate the evaluation time of candidate programs and reduce the overall time of evolution, we propose various parallel implementations of LGP suitable for the current multi-core processors. The implementations are based on a parallel evaluation of candidate programs and the island model of the parallel evolutionary algorithm in which the sub-populations are evolved independently, but some genetic material can be exchanged by means of the migration. Proposed implementations are evaluated using three symbolic regression problems and a hash function design problem.

1 Introduction

Linear genetic programming (LGP) is a form of genetic programming in which candidate programs are encoded as sequences of instructions and executed on a register machine. LGP is especially useful for designing relatively short but well-tuned programs simultaneously showing an excellent quality of processing and implementation effectiveness. Examples of evolved programs include hash functions, game strategies, communication protocols, and low-level machine code routines. If the LGP implementation is fast it can autonomously provide the optimized programs to such systems in which the specification is modified in the runtime.

The performance of LGP primarily depends on the problem encoding, search operators, fitness evaluation and efficiency of the implementation. In order to accelerate the evaluation time of candidate programs and reduce the overall time of evolution, we propose parallel implementations of LGP suitable for the current multi-core processors. Contrasted to the research dealing with efficient genetic operators and search methods [4,13], our work if focused on an efficient distribution of workload (represented mainly by the fitness evaluation as the most time consuming component of LGP) among the available computing resources.

The goal of this paper is to develop parallel implementations of LGP and compare their performance in terms of throughput measured as the number of candidate solutions evaluated per second (primary objective) and the quality of evolved programs (secondary objective). The implementations are based on the

© Springer International Publishing AG 2017
R. Matoušek (ed.), *Recent Advances in Soft Computing*, Advances in Intelligent Systems and Computing 576, DOI 10.1007/978-3-319-58088-3_7

parallel evaluation of candidate programs and the island model of the parallel evolutionary algorithm in which subpopulations are evolved independently, but some genetic material can be exchanged by means of the migration. Proposed implementations are evaluated using three symbolic regression problems and a hash function design problem.

The rest of the paper is organized as follows. Section 2 surveys the relevant work. The proposed parallel LGP methods are presented in Sect. 3. Experimental setup and benchmark problems are introduced in Sect. 4. Results are reported in Sect. 5. Conclusions are given in Sect. 6.

2 Related Work

Genetic programming (GP) is an artificial intelligence method capable of automated design of programs in a given programming language [9]. There are several branches of GP such as tree-based GP, linear GP and Cartesian GP. They primarily differ in the representation of candidate programs. This section introduces the field of genetic programing and provides relevant details about LGP and its parallelization.

2.1 Linear Genetic Programming

Linear genetic programming [1,13,18] uses a linear representation of computer programs. Every program is composed of operations, which are called instructions, and operands, which are stored in registers.

Program Representation. An instruction is typically represented by an instruction code, destination register and two source registers, for example [+, r0, r1, r2] denotes $r0 = r1 + r2$. The input data are stored in pre-defined registers or an external memory. The result is returned in a given register. The number of instructions in a candidate program is variable, but the minimal and maximal values are defined. The number of registers available in a register machine is constant. The function set known from GP corresponds with the set of available instructions. The instructions are general-purpose (e.g. addition and multiplication) or domain-specific (e.g. read sensor 1). Conditional and branch instructions are important for solving general problems. Protected versions of instructions (e.g. a division returns a value even if the divisor is zero) are employed in order to execute all programs without exceptions such as the division by zero.

Genetic Operations. LGP is usually used with a tournament selection, one-point or two-point crossover and a mutation operator modifying either the instruction type or register index. Advanced genetic operators have been proposed, see for example, [4].

Fitness Function. The most computationally expensive part of LGP is the fitness function evaluation, in which a candidate program is executed for a set of training inputs, the program's outputs are collected and the fitness value is determined.

There are essentially two types of linear GP: a machine code GP, where each instruction is directly executable by the CPU, and an interpreted linear GP, where each instruction is executable by a virtual machine (simulator) implemented for a given processor.

2.2 Parallel GP

This section provides a brief overview of approaches developed to parallelize genetic programming [2,5,15]. The approaches differ in the algorithms and hardware employed.

The farmer-model, or the master-slave model, operates with a global population of programs. The master processor performs all genetic operations and assigns the candidate solutions to remaining processors (slaves) to be evaluated [14]. One processor typically evaluates a subpopulation of candidate solutions.

In the island model [16], independent populations are evolved concurrently and separately. In every kth generation, each population sends its best individuals, according to a pre-designed communication pattern and network topology, to other populations. Synchronous and asynchronous versions of this model have been developed.

Modern CPUs support special vector instructions. If a code vectorization is enabled, CPU provides a restricted type of parallel processing, often referred to as the single instruction multiple data (SIMD) or the single program multiple data (SPMD). Then, a program response can be obtained for several test vectors in parallel.

Parallel GP implementations were also developed for specialized hardware such as graphic processing units (GPU) [2,5] and general-purpose accelerators (Xeon Phi) [6].

3 Parallel LGP

The most computationally expensive part of LGP is the fitness function evaluation. In this section, two approaches to the fitness evaluation are presented. A method enabling the parallelization of these evaluation approaches is then discussed in Sect. 3.2. Finally, an island-based parallel LGP is introduced.

3.1 Fitness Evaluation

The first method (Method A, see Algorithm 1) determines the fitness value in a straightforward manner according to a common sequential code for genetic programming. Each candidate program is executed for each vector of the training set.

In general, conditional branches make difficult to effectively employ the code vectorization because it has to be determined for each instruction whether it will be executed or not. Method A might be suitable for instruction sets containing conditional branches, because it can skip a number of instructions if a conditional branch is present. This leads to the reduction of the execution time of a candidate solution.

Algorithm 1. Fitness: Method A

```
INPUT: Population, Training set
1: FOR EACH individual of the population DO
2:    FOR EACH vector of the training set DO
3:       Initialize registers by the vector
4:       Sequentially execute instructions of the individual
5:    Update the fitness value
```

The second method (Method B, see Algorithm 2) divides the training set into chunks of training vectors. The chunk size is one of LGP parameters. All individuals are executed for all vectors from a given chunk which can be exploited in the subsequent parallel processing. If candidate programs contain the conditional branches the code vectorization is difficult. This method requires creating an array of registers for each vector form the chunk. The chunk size is determined experimentally. While large chunks can invoke many L2 cache misses, small chunks introduce a significant overhead and make the method inefficient.

Algorithm 2. Fitness: Method B

```
INPUT: Population, Training set, Chunk size $s$
1: Divide the training set into chunks according to $s$
2: FOR EACH chunk DO
3:    FOR EACH individual from the population DO
4:       Initialize registers by the vector from the chunk
5:       Sequentially execute instructions of the individual
         for all vectors from the chunk
6:    Compute a partial fitness function for the chunk
7: At the end compute the final fitness value
```

3.2 Parallel Fitness Evaluation

The parallel evaluation proposed for method A consists in assigning the individuals of the population to processes. As there is no communication among the individuals during the evaluation, the speedup is given by the number of available cores. The same approach is also adopted in method B, but each process evaluates the candidate programs with the subsets of training vectors (chunks).

The parallel fitness evaluation is realized with OpenMP [3] which is a multi-platform shared-memory parallel programming paradigm developed for C/C++ and Fortran. Programs utilizing OpenMP are limited in the number of processes which should be less or equal to the number of logic cores of a given CPU. If the number of processes is higher the processes start to compete for resources and the overall overhead is growing.

3.3 Island Model for Parallel LGP

The proposed implementation utilizes the island-based model with a ring topology. The individuals occupying a given island are evaluated using method B. The communication between the islands is asynchronous. As the evaluation of population(s) on the islands may consume a different time, faster islands do not have to wait for slower islands. After a predefined number of generations, every island sends the best individuals to its neighbors. All islands try to receive some individuals from other islands in every generation. Newly incoming individuals replace randomly chosen individuals of the population, but the best scored individuals are always preserved. The individuals are sent as integer array messages. The implementation is based on MPI [10].

4 Setup and Benchmark Problems

This section defines the experimental setup and benchmark problems utilized for comparison of the proposed LGP implementations.

The LGP code was written in C and compiled with gcc version 4.9.3, with full optimization (O3) and vectorization enabled. All experiments were carried out on a Linux machine equipped with the Intel Xeon E5-2630 processor. There is a limitation of 12 processes in the parallel model.

4.1 Setup

Two scenarios were developed to evaluate LGP implementations. The purpose of the first one is to measure the performance in terms of the number of instructions/generations that can be executed within a given time. LGP parameters are summarized in Table 1. The second scenario is used to evaluate the quality of evolution (i.e. the obtained fitness). This scenario is employed for testing the island model. LGP parameters are given in Table 1. The fitness function for symbolic regression problems is defined as the mean absolute error between the program outputs and desired output.

In both scenarios, the program size is constant which allows for a straightforward comparison of the execution time. In order to investigate the impact of branch instructions, two function sets are defined. Both function sets contain arithmetic operations. However, the second function set ($T2$) contains a branch instruction which introduces, in principle, numerous difficulties during the code vectorization. Evaluation of these candidate solutions is thus slower.

Table 1. LGP parameters for symbolic regression

Parameter	Performance tests	Island model
Population size	1000	1000
Crossover probability	90%	90%
Mutation probability	15%	15%
Program length	40	40
Registers count/type	16/double	16/double
Instruction set T_1	{+, −, *, /}	{+, −, *, /}
Instruction set T_2	{+, −, *, /, IF}	-
Terminal set	{1, indepen. variables}	{1, indepen. variables}
Tournament size	4	4
Maximum number of generations	1000	1000
Crossover type	One-point	One-point
Migration	-	40 generations

4.2 Benchmarks Problems

Symbolic Regression. Three standard symbolic regression problems were taken from [7,17]. Table 2 gives intervals used for construction of the training sets.

F1:

$$f(x,y) = \frac{x^2 + y^2}{2y}$$

F2:

$$f(x) = x^4 + x^3 + x^2 + x$$

F3:

$$f(x,y) = \frac{x^3}{5} + \frac{y^3}{2} - y - x$$

Table 2. Parameters of training sets generated using benchmark functions

Function	Performance tests			Island model		
	Training set size	Range	Step	Training set size	Range	Step
F1	10000	$x, y \in <0, 100)$	1	10000	$x, y \in <0, 100)$	1
F2	10000	$x \in <0, 10000)$	1	100	$x \in <0, 100)$	1
F3	10000	$x, y \in <0, 100)$	1	900	$x, y \in <0, 30)$	1

Hash Function Design. In order to evaluate LGP on a real-world problem, we will employ the hash function design problem. The objective is to find a hash function which maps the data of an arbitrary size to the data of a fixed size. A good hash function should have some important properties, for example, a small input change should invoke a large output change to minimize potential collisions, it has to be deterministic and easy to compute. Detailed description of the principles of hash functions is available in [8]. A hash function is often used in hash tables. A hash function produces an index to the table from input data. In the case of a collision, several possibilities exist to solve it [12]. One of the approaches is a perfect hashing in which a special hash function is created for a given data set such that it does not produce any collisions for this set [11].

In the area of computer networks, it is often needed to track specific users or devices. They can be identified by IP addresses. However, the number of IP address may vary over time. It makes sense to develop a specialized hash function in order to eliminate disadvantages of universal hash functions that can produce a large amount of collisions for a specific set of IP addresses. It is time expensive and very difficult to create good hash functions manually. Automated methods are sought that can quickly provide a good hash function for monitored traffic. Providing a good hash function in a short time enables to start network monitoring sooner and catch more important data. LGP can be employed to search for the perfect hash function for a given set of IP addresses.

For this study, a set of IP addresses was randomly selected from the firewall in our computer network. Experiments will be performed with two data sets containing 1000 and 5000 IP addresses and a 16 bit hash table. The fitness function computes the number of collisions produced by a candidate hash function on a given training set. The goal of LGP is to minimize the number of collisions.

The instruction set contains operations that can be found in common hash functions (RightRotation, NOT, AND, OR, XOR, +, *) and a set of 10 prime numbers used in the initialization phase of the SHA-2 function represent the constants available to LGP. The program length is restricted to 20 instructions as common hash functions are of this size. LGP uses 8 integer registers. Other settings are given in Table 1.

5 Results

Results of experiments are structured into two parts in this section: symbolic regression benchmarks and hash function design.

5.1 Symbolic Regression Benchmarks

In total, LGP was employed to solve 60 specifications which differ in the method (A or method B with 5 different chunk sizes - 500, 1000, 2500, 5000 and 10,000 vectors), function set (T_1, T_2) and the number of cores (1, 2, 4, 6 and 12) used. In order to obtain basic statistics, 20 independent LGP runs were performed for each specification. Each run produced 1000 generations.

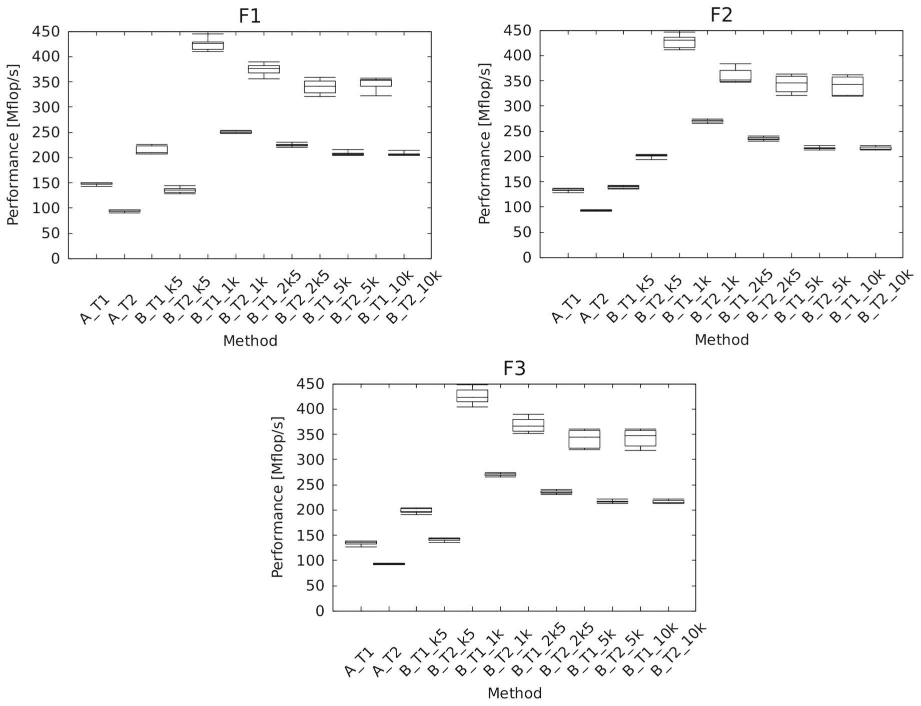
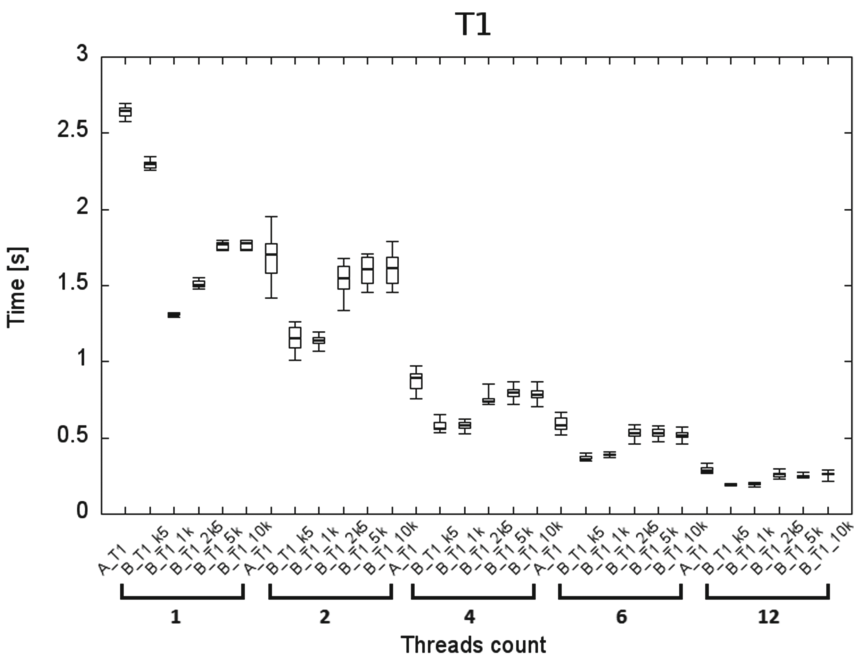


Fig. 1. Sequential performance of methods A and B on test problems F1, F2 and F3. Method B is used with chunk sizes 500, 1000, 2500, 5000 and 10000 vectors. The x-axis is in the format m_t_c, where m is the method (A or B), t is the instruction set ($T1$ or $T2$) and c is the chunk size (for method B).

In the first experiment, we compared the sequential implementations of both methods. The boxplots shown in Fig. 1 give LGP performance per one generation in MFLOPs (measured by PAPI). The boxplots used in these figures represent the minimum, first quartile, median, third quartile and maximum. The experiments confirmed our assumption that method B provides higher performance than method A. An optimal chunk size seems to be 1000 vectors. The chosen instruction set significantly influences the performance. Instruction set $T1$ (without IF) leads to higher performance than $T2$, because the code vectorization is more efficient.

In the second experiment, the scalability of the parallel implementations was evaluated using 2, 4, 6, and 12 cores. The time needed to evaluate one generation is reported in the form of boxplots in Fig. 2. The implementation scales almost linearly, but a small communication overhead is present for 6 and 12 cores. The impact of the function set selection on performance is identical to previous experiments. In order to maximize the performance, it is important to correctly choose the chunk size (1000 vectors). Results are only presented for F1 because the results for F2 and F3 are almost identical with F1.



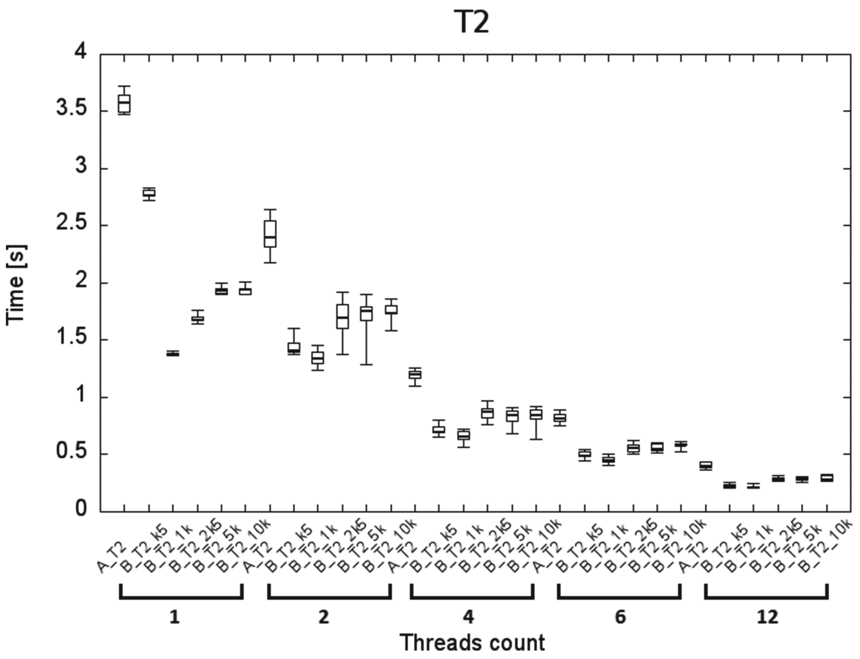


Fig. 2. The time needed to evaluate one generation using 1, 2, 4, 6 and 12 cores (F1 test problem) for instruction sets $T1$ and $T2$.

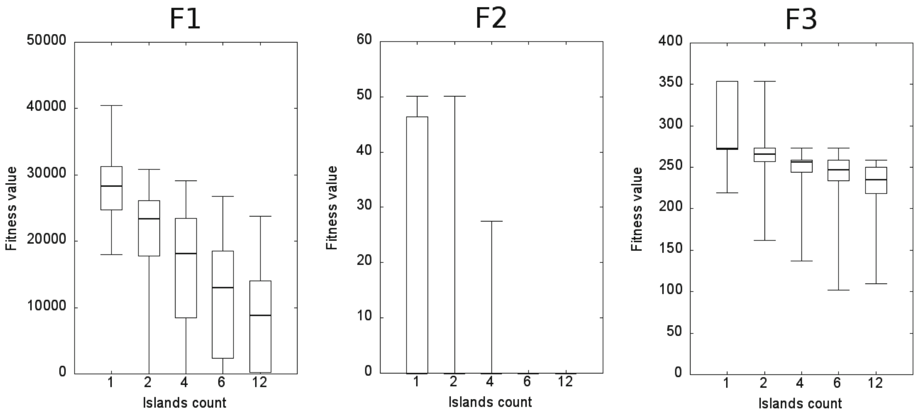


Fig. 3. The best obtained fitness values for F1, F2 and F3 from 20 independent runs on the island model.

The last experiment was devoted to the island-based LGP. In this case, the boxplots show the best fitness values obtained at the end of 20 independent runs for 1, 2, 4, 6 and 12 islands (Fig. 3). The execution time is identical for all runs. Let a perfect solution be such a solution which obtains a zero fitness. It can be seen that a perfect solution was discovered when more than one island is used for F1, independent of the islands count for F2 and never for F3. Especially for F1 and F3, increasing the number of islands has a positive impact on the quality of discovered solutions.

5.2 Hash Function Design

Figure 4 summarizes the execution time (performance) and fitness values obtained for the hash function design problem using different LGP implementations.

A comparison of sequential versions of methods A and B (with the chunk size of 1000 vectors) revealed that method B can save about 40% of the design time. The results are given for 1000 and 5000 IP addresses in the data set. Note that 20 independent runs with identical seeds for method A and B in each run were performed to obtain these boxplots.

The third and fourth graph in Fig. 4 compares the quality of solutions evolved using the island model. Results were obtained using 10 independent randomly seeded runs. It can be seen that if the training IP data set is small (1000 IP addresses), a solution is discovered on one of the islands before any migration is carried out. Hence other islands are not needed. LGP utilizing the island model becomes useful for larger data sets - see the rightmost graph in Fig. 4 showing a significant improvement in the fitness on 6 islands for a data set containing 5000 IP addresses. Example of evolved hash function is given in Fig. 5.

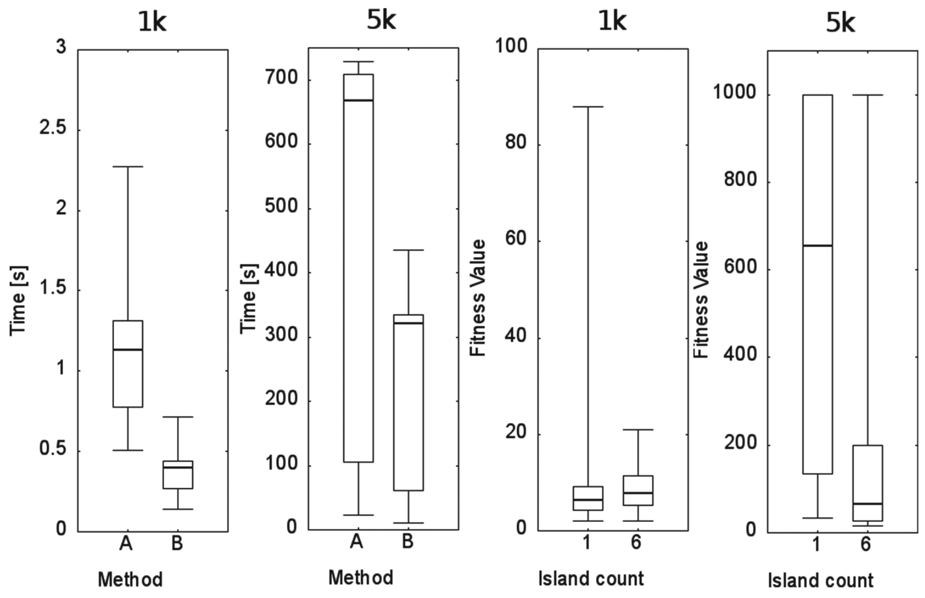


Fig. 4. Sequential execution time for methods A and B (graph 1 and 2) and fitness values for the island model (graph 3 and 4) in the evolutionary design of the hash function using 1k and 5k IP addresses in the dataset.

```
int Hash (int x ){
  r[0] = x
  r[4] = 0x5be0cd19
  r[6] = r[4]
  r[1] = r[0]
  r[7] = r[4] and 0x3c6ef372
  r[0] = RightRotation(r[0], r[7])
  r[0] = r[6] xor r[0]
  r[4] = not r[1]
  r[0] = r[4] xor r[0]
  return r[0]
}
```

Fig. 5. Example of LGP individual.

6 Conclusions

In this paper, we presented several parallel implementations of LGP devoted to the common multi-core processors. In particular, we proposed an efficient method for evaluation of candidate programs based on dividing the training data into chunks. The main criterion for our evaluation was the throughput, i.e. how many candidate programs can be evaluated in a given time. The best performing implementation utilizes the island model and the training data partition into chunks.

These implementations were compared using three symbolic regression problems and hash function design problem. Unfortunately, the available literature does not provide results from other parallel LGP implementations suitable for a fair comparison. Our future work will be focused on the evolutionary design of efficient hash functions for network applications using the proposed parallel LGP.

Acknowledgments. This work was supported by Brno University of Technology project FIT-S-14-2297 and The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science - LQ1602.

References

1. Brameier, M., Banzhaf, W.: Linear Genetic Programming. Springer, New York (2007)
2. Cheang, S.M., Leung, K.S., Lee, K.H.: Genetic parallel programming: design and implementation. Evol. Comput. **14**(2), 129–156 (2006)
3. Dagum, L., Enon, R.: Openmp: an industry standard API for shared-memory programming. IEEE Comput. Sci. Eng. **5**(1), 46–55 (1998)
4. Platel, M., Clergue, M., Collard, P.: Maximum homologous crossover for linear genetic programming. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 194–203. Springer, Heidelberg (2003). doi:10.1007/3-540-36599-0_18
5. Harding, S., Banzhaf, W.: Fast genetic programming on GPUs. In: Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I. (eds.) EuroGP 2007. LNCS, vol. 4445, pp. 90–101. Springer, Heidelberg (2007). doi:10.1007/978-3-540-71605-1_9
6. Hrbacek, R.: Bent functions synthesis on Intel Xeon Phi coprocessor. In: Hliněný, P., Dvořák, Z., Jaroš, J., Kofroň, J., Kořenek, J., Matula, P., Pala, K. (eds.) MEMICS 2014. LNCS, vol. 8934, pp. 88–99. Springer, Cham (2014). doi:10.1007/978-3-319-14896-0_8
7. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 70–82. Springer, Heidelberg (2003). doi:10.1007/3-540-36599-0_7
8. Knuth, D.E.: The art of computer programming. Sorting Search. **3**, 426–458 (1973)
9. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, vol. 1. MIT press, Cambridge (1992)
10. Lusk, E., Huss, S., Saphir, B., Snir, M.: MPI: a message-passing interface standard (2009)
11. Majewski, B.S., Wormald, N.C., Havas, G., Czech, Z.J.: A family of perfect hashing methods. Comput. J. **39**(6), 547–554 (1996)
12. Maurer, W.D., Lewis, T.G.: Hash table methods. ACM Comput. Surv. (CSUR) **7**(1), 5–19 (1975)
13. Oltean, M., Grosan, C.: A comparison of several linear genetic programming techniques. Complex Syst. **14**(4), 285–314 (2003)

14. Oussaidene, M., Chopard, B., Pictet, O.V., Tomassini, M.: Parallel genetic programming and its application to trading model induction. Parallel Comput. **23**(8), 1183–1198 (1997)
15. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: A field guide to genetic programming (2008). Lulu.com
16. Tomassini, M.: Spatially Structured Evolutionary Algorithms. Springer, Heidelberg (2005)
17. Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, R.I., Galván-López, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Program. Evol. Mach. **12**(2), 91–119 (2011)
18. Wilson, G., Banzhaf, W.: A comparison of cartesian genetic programming and linear genetic programming. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., Falco, I., Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 182–193. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78671-9_16

Hybridization of Multi-chaotic Dynamics and Adaptive Control Parameter Adjusting jDE Strategy

Roman Senkerik[1(✉)], Michal Pluhacek[1], Ivan Zelinka[2], Adam Viktorin[1], and Zuzana Kominkova Oplatkova[1]

[1] Faculty of Applied Informatics, Tomas Bata University in Zlin, Nam T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
{senkerik, pluhacek, aviktorin, oplatkova}@fai.utb.cz

[2] Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava, 17. Listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
ivan.zelinka@vsb.cz

Abstract. This research deals with the hybridization of several approaches for evolutionary algorithms, which are the adaptive control parameter adjusting strategy and multi-chaotic dynamics driving the selection of indices in Differential Evolution (DE). The novelty of the paper is given by the experiments with the multi-chaos-driven adaptive DE concept inside adaptive parameter adjusting DE strategies. These experiments are representing the investigations on the mutual influences of several different randomizations types together with adaptive DE strategies. The multi-chaotic concept is representing the adaptive switching between two different chaotic systems based on the progress of individuals within population. This paper is aimed at the embedding of discrete dissipative chaotic systems in the form of multi-chaotic pseudo random number generators for the jDE, which is the state of the art representative of simple adaptive control parameter adjusting strategy for DE. Repeated simulations for two different combinations of 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 interconnection of the two recent techniques for evolutionary algorithms, which are the adaptive control parameter adjusting strategy and multi-chaotic dynamics driving the selection of indices in Differential Evolution. This paper is aimed at investigating the influence of complex multi-chaotic dynamics to the performance of simple adaptive Differential Evolution (DE) algorithm [1]. The adaptive control parameter adjusting strategy of interest within this paper is the state of the art representative jDE [2].

A number of DE variants have been recently developed with the emphasis on adaptivity/selfadaptivity [2], ensemble approach [3] or utilization for discrete domain problems. The importance of randomization as a compensation of limited amount of search moves is stated in the survey paper [4]. This idea has been carried out in subsequent studies describing various techniques to modify the randomization process [5, 6] and especially in [7], where the sampling of the points is tested from modified distribution. The importance and influence of randomization operations was also deeply experimentally tested in jDE strategy [8]. Together with this persistent development in such mainstream research topics, the basic concept of chaos driven DE have been introduced.

Recent research in chaos driven heuristics has been fueled with the predisposition that unlike stochastic approaches, a chaotic approach is able to bypass local optima stagnation. A chaotic approach generally uses the chaotic map in the place of a pseudo random number generator [8]. 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 (or combination of chaotic maps) as the pseudo random number generator (PRNG).

The focus of our research is the direct embedding of chaotic dynamics in the form of chaos pseudo random number generator (CPRNG) for heuristic. The initial concept of embedding chaotic dynamics into the evolutionary/swarm algorithms is given in [9]. Later, the initial study [10] was focused on the simple embedding of chaotic systems for DE and Self Organizing Migration Algorithm (SOMA) [11]. Also the PSO (Particle Swarm Optimization) algorithm with elements of chaos was introduced as CPSO [12] followed by the introduction of chaos embedded PSO with inertia weigh strategy [13], further PSO strategy driven alternately by two chaotic systems [14] and finally PSO with ensemble of chaotic systems [15]. Recently the chaos driven heuristic concept has been utilized in ABC algorithm [16] and applications with DE [17].

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 experiments with multi-chaos driven DE concept [18] and initial research with connection of jDE algorithm and chaotic dynamics [19]. The motivation and novelty for this research are based on following basis and assumptions:

- Recent advances in connection of complexity and heuristic [20] together with the research focused on selection of indices in DE [21] supported the previous very promising experimental results obtained through the utilization of different chaotic dynamics as CPRNGs driving the selection, mutation, crossover or other processes in particular heuristics.
- The idea was then to connect into the one complex concept two different influences on the performance of DE, which are control parameters adaptability and complex randomization framework, which is the multi-chaotic dynamics as complex CPRNGs utilizing several known different influences on the heuristics.

- The novelty of the paper is given by the aforementioned concept and the investigations on the mutual influences of several different randomizations types together with simple parameter adaptive DE strategies. The goal is to investigate whether the complex adaptive randomization is beneficial within adaptive strategies or is suppressed by the control parameter self-adjustment. The results may be used for the possibility of creating the multi-chaotic framework with adaptively driven pool of many chaotic systems for any adaptive modern DE strategy.

The adaptive jDE strategy is hybridized here with multi-chaotic CPRNGs. Furthermore we wanted to show, that it is possible to successfully implement as the plug-in the multi-chaotic CPRNGs not only to the canonical versions of DE/PSO/Firefly but also to the more advanced adaptive strategies.

3 The Concept of Chaotic jDE

DE is a population-based optimization method that works on real-number-coded individuals [1, 22]. DE is quite robust, fast, and effective, with global optimization ability. There are essentially five inputs to the heuristic. *D* is the size of the problem, *Gmax* 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.

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 [1] 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, 23].

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.

Previous successful initial experiments with chaos driven PSO and DE algorithms [15, 18] have manifested that very promising experimental results were obtained through he utilization of Delayed Logistic, Lozi, Burgers, Lozi, Delayed Logistic map and Tinkerbelt maps. These chaotic maps have unique properties with connection to DE: Either strong progress towards global extreme, but weak overall statistical results, like average cost function value and std. dev., and tendency to premature stagnation; Or the continuously stable and very satisfactory performance. In this research, two representatives of these aforementioned different classes of influences were connected together and embedded into jDE, thus Multi-Chaos_jDE concept was introduces and tested.

4 Chaotic Maps

Following two discrete dissipative chaotic maps were used as the multi-chaotic pseudo random generators for jDE: Burgers (1), and Lozi map (2).

The Burgers mapping is a discretization of a pair of coupled differential equations which were used by Burgers [24] to illustrate the relevance of the concept of bifurcation to the study of hydrodynamics flows. The Lozi map is a discrete two-dimensional chaotic map. With the typical settings as in Table 1, systems exhibits typical chaotic behavior [25].

Table 1. Definition of chaotic systems used as CPRNGs

Chaotic maps equations	Parameter settings
$X_{n+1} = aX_n - Y_n^2$, $Y_{n+1} = bY_n + X_nY_n$ (1)	$a = 0.75$ and $b = 1.75$
$X_{n+1} = 1 - a\lvert X_n\rvert + bY_n$, $Y_{n+1} = X_n$ (2)	$a = 1.7$ and $b = 0.5$

The illustrative simulation outputs of two chaotic systems used for obtaining of pseudo-random numbers are depicted in Fig. 1.

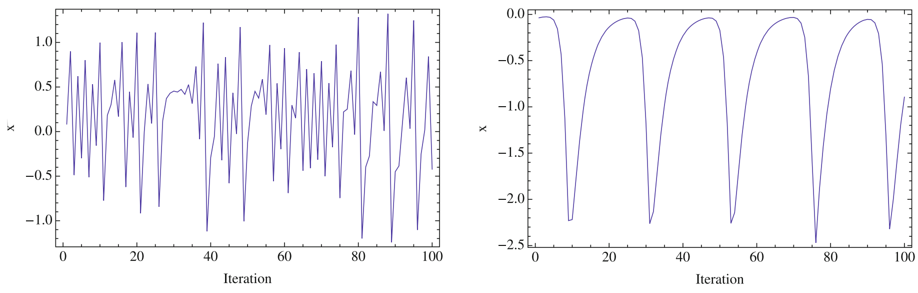


Fig. 1. Simulation outputs – chaotic series used for obtaining of pseudo random numbers and generated by means of the chaotic Lozi map (left) and Burgers map (right).

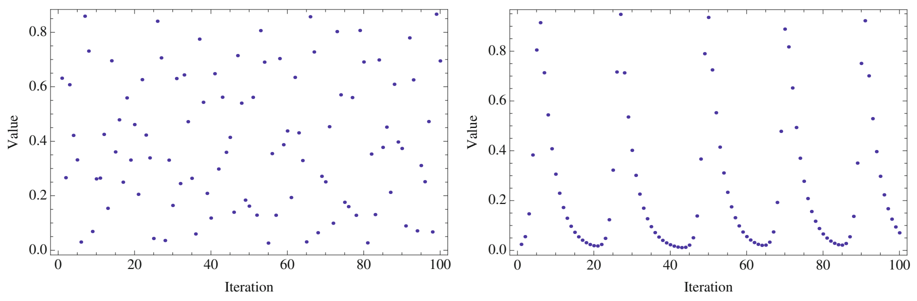


Fig. 2. Detailed sequencing and dynamics of real pseudo-random numbers transferred into the range <0-1> generated by means of the chaotic Lozi map (left) and Burgers map (right).

Furthermore Fig. 2 shows sequencing and dynamics of pseudo-random numbers transferred into the range <0-1> obtained from chaotic series (See Fig. 1).

5 Results

IEEE CEC 2013 benchmark set [26] was utilized within this experimental research for the purpose of performance comparison of two versions of multi-chaotic jDE and original (canonical) jDE,

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 jDE variants within a limited number of generations and cost function evaluations.

To maximize the benefit from the influences of different chaotic dynamics, the simple adaptive control approach was used. The exact transition point is determined by following simple rule: If the change of global best value between two subsequent generations is less than 0.01 over more than 1% of total number of generations, the chaotic systems used as the CPRNGs are alternated.

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

- *Multi-Chaos_jDE LoziBur:* Initialized with Lozi map, subsequently the adaptive switching between Lozi and Burgers map is applied.
- *Multi-Chaos_jDE BurLozi:* Initialized with Burgers map, subsequently the adaptive switching between Lozi and Burgers map is applied.
- State of the art adaptive representative canonical jDE (with default PRNG).

Results of experiments are shown in Tables 2 and 3. Table 2 contains the average cost function (CF) error values (CF – fmin), whereas Table 3 shows the minimum error values of the found CF values (CFmin – fmin) representing the best individual solution for all 50 repeated runs of two versions of Multi-Chaos_jDE, and canonical jDE. Finally the Within Tables 2 and 3, the bold values represent the best performance, italic equal; based on summary characteristics. The significant differences (even for known extreme) for *f2-f4* are given by the high complexity of composite function.

Table 2. Final average CF values: Performance comparison for two versions of Multi-Chaos_jDE and canonical jDE on CEC 13 Benchmark Set, *dim* = 30, max. Generations = 1500

DE Version/Function	f min	jDE	Multi-Chaos_jDE LoziBur	Multi-Chaos_jDE BurLozi
f1	−1400	9,63E − 13	5,00E − 13	**1,67E − 13**
f2	−1300	1,09E + 07	3,26E + 06	**1,87E + 06**
f3	−1200	3,13E + 06	2,29E + 06	**1,93E + 06**
f4	−1100	2,39E + 04	**4,93E + 03**	4,98E + 03
f5	−1000	1,94E − 08	**1,23E − 08**	2,16E − 09
f6	−900	1,98E + 01	**1,97E + 01**	2,56E + 01
f7	−800	1,14E + 01	6,06E + 00	**5,75E + 00**
f8	−700	*2,10E + 01*	*2,10E + 01*	*2,10E + 01*
f9	−600	3,43E + 01	3,00E + 01	**1,66E + 01**
f10	−500	9,89E − 01	1,66E − 01	**3,80E − 02**
f11	−400	**3,66E + 01**	4,72E + 01	6,59E + 01
f12	−300	1,68E + 02	1,58E + 02	**1,24E + 02**
f13	−200	1,79E + 02	1,74E + 02	**1,65E + 02**
f14	−100	**2,26E + 03**	2,80E + 03	4,78E + 03
f15	100	**7,15E + 03**	7,22E + 03	7,38E + 03
f16	200	2,67E + 00	**2,66E + 00**	2,68E + 00
f17	300	**8,68E + 01**	9,75E + 01	1,46E + 02
f18	400	2,16E + 02	2,15E + 02	**2,12E + 02**
f19	500	**7,45E + 00**	8,31E + 00	1,11E + 01
f20	600	1,26E + 01	1,24E + 01	**1,23E + 01**
f21	700	3,12E + 02	**2,93E + 02**	3,12E + 02
f22	800	**2,76E + 03**	3,29E + 03	4,89E + 03
f23	900	*7,49E + 03*	*7,49E + 03*	7,63E + 03
f24	1000	2,06E + 02	*2,04E + 02*	*2,04E + 02*
f25	1100	2,92E + 02	2,64E + 02	**2,46E + 02**
f26	1200	*2,00E + 02*	*2,00E + 02*	*2,00E + 02*
f27	1300	6,06E + 02	4,10E + 02	**3,88E + 02**
f28	1400	*3,00E + 02*	*3,00E + 02*	*3,00E + 02*

Table 3. Best solutions - minimal CF values: Performance comparison for two versions of Multi-Chaos_jDE and canonical jDE on CEC 13 Benchmark Set, *dim* = 30, max. Gen. = 1500

DE Version/Function	f min	jDE	Multi-Chaos_jDE LoziBur	Multi-Chaos_jDE BurLozi
f1	−1400	2,27E − 13	2,27E − 13	**0,00E + 00**
f2	−1300	5,04E + 06	1,09E + 06	**3,95E + 05**
f3	−1200	3,44E + 04	4,30E + 04	**1,25E + 03**
f4	−1100	1,31E + 04	**1,84E + 03**	1,85E + 03
f5	−1000	5,48E − 09	4,15E − 09	**4,74E − 10**
f6	−900	1,54E + 01	1,46E + 01	**1,41E + 01**
f7	−800	3,32E + 00	1,86E + 00	**8,31E-01**
f8	−700	*2,09E + 01*	*2,09E + 01*	*2,09E + 01*
f9	−600	2,89E + 01	8,63E + 00	**5,85E + 00**
f10	−500	3,36E − 01	8,87E − 03	**7,40E − 03**
f11	−400	**2,67E + 01**	2,89E + 01	4,71E + 01
f12	−300	1,43E + 02	1,30E + 02	**2,79E + 01**
f13	−200	1,52E + 02	1,48E + 02	**7,02E + 01**
f14	−100	**1,66E + 03**	2,31E + 03	4,03E + 03
f15	100	**6,30E + 03**	6,57E + 03	6,63E + 03
f16	200	1,98E + 00	**1,78E + 00**	2,10E + 00
f17	300	**7,90E + 01**	8,30E + 01	1,20E + 02
f18	400	1,96E + 02	1,95E + 02	**1,80E + 02**
f19	500	**6,11E + 00**	6,62E + 00	8,03E + 00
f20	600	1,20E + 01	1,17E + 01	**1,16E + 01**
f21	700	*2,00E + 02*	*2,00E + 02*	*2,00E + 02*
f22	800	**2,10E + 03**	2,63E + 03	4,07E + 03
f23	900	7,07E + 03	**6,84E + 03**	6,97E + 03
f24	1000	2,02E + 02	2,01E + 02	**2,00E + 02**
f25	1100	2,59E + 02	2,01E + 02	**2,00E + 02**
f26	1200	*2,00E + 02*	*2,00E + 02*	*2,00E + 02*
f27	1300	3,87E + 02	3,23E + 02	**3,11E + 02**
f28	1400	*3,00E + 02*	*3,00E + 02*	*3,00E + 02*

The different influences of two CPRNGs around switching point are visible from the Figs. 3 and 4 (around generations No. 1350), which depicts for selected test functions the illustrative examples of time evolution of average CF values for all 50 runs of two versions of Multi-Chaotic_jDE and canonical jDE.

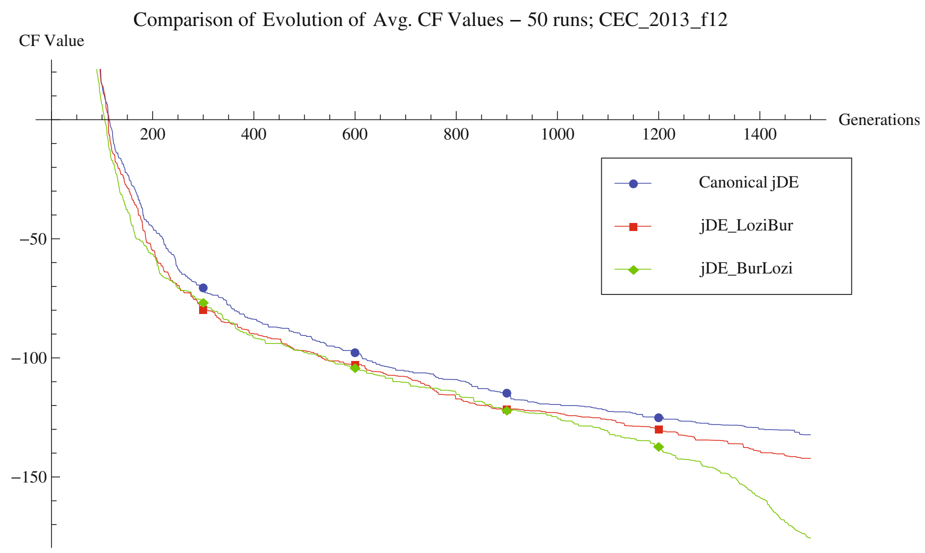


Fig. 3. Comparison of the time evolution of avg. CF values for the all 50 runs of Canonical jDE, and all two versions of Multi-Chaos_jDE, Visible switching point approx. gen. No. 1350; CEC2013_f12, D = 30.

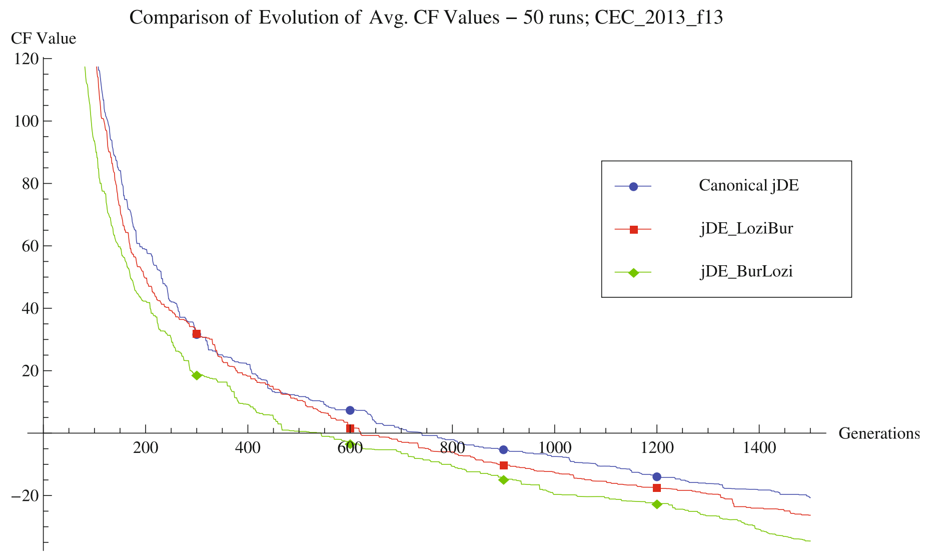


Fig. 4. Comparison of the time evolution of avg. CF values for the all 50 runs of Canonical jDE, and all two versions of Multi-Chaos_jDE, Visible switching point approx. gen. No. 1350; CEC2013_f13, D = 30.

6 Conclusion

The primary aim of this work is to use and test the hybridization of complex combination of natural chaotic dynamics as the chaotic pseudo random number generators with adaptive switching and self-adaptive mechanism for differential evolution algorithm. In this paper the novel concept of Multi-Chaos_jDE strategy driven by two chaotic maps (systems) is introduced. Two different influences on the performance of DE were connected here into the one adaptive multi-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:

- It is fully manifested here the high sensitivity of the jDE to the complex dynamics and sequencing of the different chaotic PRNGs driving the selection of indices (solutions) in jDE.
- When comparing the Multi-Chaos_jDE and original jDE, one version of multi-chaos driven heuristic has outperformed the original jDE in both cases of final average CF values and min. CF values. Also the transition phases are visible as in Figs. 3 and 4. This means that the chaos driven indices selection keeps its unique properties also in multi-chaotic framework and adaptive DE strategy environment. The previously successful single_chaos_jDE [19] was not compared here, as the motivation for the paper is given by the investigations on the mutual influences of several different randomizations types together with simple parameter adaptive DE strategies. Thus to investigate whether the complex adaptive randomization is beneficial within adaptive strategies or it is suppressed by the control parameter self-adjustment. The findings may be used as the basis for the building of the multi-chaotic framework with adaptively driven pool of many chaotic systems for any adaptive modern DE strategy (EPSDE, SHADE…).
- Based on the obtained results, we can assume that in case of differential evolution, the sensitivity to the CPRNG driving the selection of individuals and crossover process may be very beneficial together with the influence of adaptive tuning of control parameters.
- Furthermore many previous implementations of chaotic dynamics into the evolutionary/swarm based algorithms (not-adaptive/adaptive/ensemble based) showed that it is advantageous, since it can be easily implemented into any existing algorithm as a plug-in module.

Acknowledgements. This work was supported by Grant Agency of the Czech Republic - GACR P103/15/06700S, further by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Pro- gramme project No. LO1303 (MSMT-7778/2014) 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. SP2016/175 of VSB - Technical University of Ostrava, Czech Republic and by Internal Grant Agency of Tomas Bata University under the project No. IGA/CebiaTech/2016/007.

References

1. Price, K.V.: An introduction to differential evolution. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 79–108. McGraw-Hill Ltd., London (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. 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**(2), 1679–1696 (2011)
4. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. **33**(1–2), 61–106 (2010)
5. Weber, M., Neri, F., Tirronen, V.: A study on scale factor in distributed differential evolution. Inf. Sci. **181**(12), 2488–2511 (2011)
6. Neri, F., Iacca, G., Mininno, E.: Disturbed Exploitation compact Differential Evolution for limited memory optimization problems. Inf. Sci. **181**(12), 2469–2487 (2011)
7. Iacca, G., Caraffini, F., Neri, F.: Compact differential evolution light: high performance despite limited memory requirement and modest computational overhead. J. Comput. Sci. Technol. **27**(5), 1056–1076 (2012)
8. Zamuda, A., Brest, J.: Self-adaptive control parameters′ randomization frequency and propagations in differential evolution. Swarm Evol. Comput. **25**, 72–99 (2015)
9. 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)
10. Davendra, D., Zelinka, I., Senkerik, R.: Chaos driven evolutionary algorithms for the task of PID control. Comput. Math Appl. **60**(4), 1088–1104 (2010)
11. Zelinka, I.: SOMA — self-organizing migrating algorithm. In: Onwubolu, G.C., Babu, B.V. (eds.) New Optimization Techniques in Engineering. STUDFUZZ, vol. 141, pp. 167–217. Springer, Heidelberg (2004)
12. dos Santos Coelhoa, L., 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)
13. 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)
14. 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), 20–23 June 2013, pp 2444–2449 (2013)
15. Pluhacek, M., Senkerik, R., Davendra, D.: Chaos particle swarm optimization with Eensemble of chaotic systems. Swarm Evol. Comput. **25**, 29–35 (2015)
16. Metlicka, M., Davendra, D.: Chaos driven discrete artificial bee algorithm for location and assignment optimisation problems. Swarm Evol. Comput. **25**, 15–28 (2015)
17. Coelho, L.D.S., Ayala, H.V.H., Mariani, V.C.: A self-adaptive chaotic differential evolution algorithm using gamma distribution for unconstrained global optimization. Appl. Math. Comput. **234**, 452–459 (2014)
18. Senkerik, R., Pluhacek, M., Davendra, D., Zelinka, I., Oplatkova, Z.K.: Performance testing of multi-chaotic differential evolution concept on shifted benchmark functions. In: Polycarpou, M., Carvalho, A.C.P.L.F., Pan, J.-S., Woźniak, M., Quintian, H., Corchado, E. (eds.) HAIS 2014. LNCS (LNAI), vol. 8480, pp. 306–317. Springer, Cham (2014). doi:10.1007/978-3-319-07617-1_28

19. Senkerik, R., Pluhacek, M., Davendra, D., Zelinka, I., Oplatkova, Z.K., Janostik, J.: Hybridization of adaptivity and chaotic dynamics for differential evolution. In: Matoušek, R. (ed.) Mendel 2015. AISC, vol. 378, pp. 149–158. Springer, Cham (2015). doi:10.1007/978-3-319-19824-8_12
20. Zelinka, I.: A survey on evolutionary algorithms dynamics and its complexity – mutual relations, past, present and future. Swarm Evol. Comput. **25**, 2–14 (2015)
21. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution using a neighborhood-based mutation operator. IEEE Trans. Evol. Comput. **13**(3), 526–553 (2009)
22. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution - A Practical Approach to Global Optimization. Natural Computing Series. Springer, Heidelberg (2005)
23. 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. ISRL, vol. 38, pp. 423–449. Springer, Berlin Heidelberg (2013)
24. 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 (2007)
25. Sprott, J.C.: Chaos and Time-Series Analysis. Oxford University Press, New York (2003)
26. 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)

Geometric Particle Swarm Optimization and Reservoir Computing for Solar Power Forecasting

Sebastián Basterrech(✉)

Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB–Technical University of Ostrava, Ostrava, Czech Republic
Sebastian.Basterrech.Tiscordio@vsb.cz

Abstract. Solar irradiance is an alternative of renewable resource that can be used for covering a relevant part of the growing demand of electrical energy. To have accurate solar irradiance predictions can help to integrate the solar power resources into the grid. We analyse the performance of an automatic procedure for selecting the most significant input features that impacts on the solar irradiance. The approach is based on a generalisation of swarm optimisation named *Geometrical Particle Swarm Optimization (GPSO)*. Once, a good combination of weather information is defined, we use a reservoir computing model as forecasting technique. In particular, we use the Echo State Networks (ESN) model that is a Recurrent Neural Network often used for solving temporal learning problems. We evaluate our approach on a well-known public meteorological dataset obtaining promising results.

Keywords: Geometric Particle Swarm Optimization · Solar energy · Echo State Networks · Forecasting · Time-series problems

1 Introduction

In order to protect our environment the global interest of reducing the Carbon emissions has augmented during the last years. In parallel, the demand for electrical energy has also augmented over the years due to the increasing demand for electric products around the globe. Solar energy is a clean renewable resource that can cover a part of the growing demand of electrical energy. Besides, the performance of photovoltaic systems have had a significant growth, and their costs have been decreasing over the last years. As a consequence, electricity produced by solar energy starts to be integrated to the energetic grid infrastructure. The prediction of the solar power resources can be useful for that integration, good predictions can help in the definition of good plans and mitigate the negative impacts of instable energy sources.

The main goals of this article are the following ones. To evaluate the performance of an heuristic procedure for defining a set of meteorological variables that impact on the solar power estimation. Besides, we are interested in creating a forecasting system with two types of input features: a selected group of meteorological variables at current time, and a time-series of estimated values of previous solar irradiance until current time, such that can estimate (well as possible) the solar irradiance at current time. In that way, we present a procedure for selecting weather variables and for predicting the solar irradiance. Our approach is based on a recently introduced bio-inspired optimisation technique called *Geometric Particle Swarm Optimization* [1] that is one derivation of the popular *Particle Swarm Optimization (PSO)* [2] and Evolutionary Algorithms [3]. We apply GPSO for selecting the most significant input features for the forecasting model. Afterward, the forecasting is done using the *Echo State Networks (ESN)* [4] model. An ESN is a Recurrent Neural Network often used for solving temporal learning problems. We evaluate our approach on a well-known public meteorological dataset obtaining promising results. Note that the forecasting have been made using the solar irradiance instead of solar power, but exists a great correlation between those variables: solar irradiance and power of PV systems [5]. For this reason, in related works is most often predicted the solar irradiance [5]. Several related works of forecasting solar irradiance has been presented during the last years, for instance see [6–11].

The article is organised as follows. Section 2 starts presenting a definition of the problem of forecasting a time-series, next the GPSO and the ESN models are introduced. Section 3 introduces the proposed procedure that is a combination of GPSO and ESN. In Sect. 4 is presented the benchmark dataset. Next, we present an analysis of the experimental results. As usual, at the end we present some conclusions and an outlook.

2 Background

This section presents the problem of generating a mapping between input features and output variables when we have temporal information, as well as a background of the techniques used for computing the predictions: GPSO and ESN.

2.1 Formalization of the Problem

We begin by specifying a temporal learning problem. The goal of forecast a time-series is obtaining information about the future using the knowledge about the past. Given a time-series $\mathbf{a}(1), \mathbf{a}(2), \dots, \mathbf{a}(t)$ each one in a N_a-dimensional real space, the goal consists in computing a parametric learning tool $\phi(\cdot, \mathbf{w})$ such that is able to predict the value of any new observation $\mathbf{a}(t+\tau)$ $(\tau \geq 1)$ better as possible. More formal, we define an objective function in an arbitrary range of time $[1, T]$ given by the *Mean Squared Error (MSE)*:

$$MSE = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N_\mathrm{a}} (\phi_i(\mathbf{a}(t), \mathbf{w}) - a_i(t+\tau))^2, \tag{1}$$

the goal is to find the parameters of $\phi(\cdot, \mathbf{w})$ such that the expression (1) is minimised.

2.2 Geometric Particle Swarm Optimization

The *Particle Swarm Optimization (PSO)* is a population-based optimisation technique widely used in the literature for searching points on complex spaces [2]. The technique has been proven to be efficient and robust in several applications [12]. It is based on the social behaviors of a set of particles called *swarm* in a simplified environment. The GPSO is a generalisation of the canonical PSO for solving combinatorial optimization problems on any type of searching space. We denote N as the number of particles in the swarm and M the dimension of the searching space. Each particle is represented as a point in the space, instead of the canonical PSO that uses two points for representing each particle (position and velocity) the GPSO uses only the position. The method has a mutation rate in the same way that Genetic Algorithms. The key operation is the position update, which is given for a convex combination of three multidimensional points. The points are weighted by the model parameters which we denote by w_1, w_2 and w_3. Each particle i is denoted by $\mathbf{p}_i \in A^M$, where A is a metric space. Here, we consider $\mathbf{p}$ as a binary string ($A = \{0, 1\}$). The procedure is iterative, at each epoch a fitness function for each particle is computed, next the particle positions are updated. We denote by $\mathbf{p}_i^l(t)$ the best position of i ever found until time t, and we denote by $\mathbf{p}^g(t)$ the best position among the all particles that has ever found until time t. The position update for a binary space is given by the following stochastic rule [1]: $\mathbf{p}_i(t+1) = \mathbf{p}_i(t)$ has a probability of w_1, $\mathbf{p}_i(t+1) = \mathbf{p}_i^l(t)$ has a probability w_2 and $\mathbf{p}_i(t+1) = \mathbf{p}_i^g(t)$ has a probability w_3. Besides, for each particle at each epoch is computed the mutation operation [1]. The procedure is extremely easy to compute, is robust, and has only the following few parameters: number of particles, mutation rate, the weights w_1, w_2 and w_3.

2.3 Echo State Neural Networks

Recurrent Neural Networks (RNNs) are dynamical systems used for modelling complex systems that evolve in time. They have been applied for solving temporal learning problems, although their applications have been limited due to the fact that is hard to optimise the network parameter values. Since the beginning of the 2000s, a type of RNN named Reservoir Computing (RC) has been studied in the community. A RC model overcomes the limitations on the training process of RNNs. Besides, it has obtained well performances for solving temporal learning tasks [13]. The Echo State Network (ESN) is one of the initial RC models [4]. A canonical ESN has at least three structures connected in a forward schema: an input layer (input neurons), a reservoir (interconnected hidden neurons) and a readout layer (output neurons). The ESN has circuits only in the reservoir (in the hidden-hidden weights). The circuits permit to model temporal information and discover correlations between input patterns. The reservoir weights are random initialized following some algebraic conditions and they are fixed during the learning process. The training algorithm only focuses in adjusting the weight connections from the reservoir to readout neurons. The goal of the reservoir structure is encoding the temporal information and transforming the input patterns in a new linear separable space. In addition, the learning algorithm is fast and robust due to the fact that only the readout weights are updated during the training steps. The ESN model has proven to obtain well performances in many applications [13].

We follow by specifying the notation, let N_{a}, N_{x} and $\mathrm{N_o}$ be the number of input, reservoir and output neurons, respectively. The weights are collected in the following matrices: let $\mathbf{w}^{\mathrm{in}}$ be a $N_{\mathrm{x}} \times N_{\mathrm{a}}$ matrix collecting input-reservoir weights, let $\mathbf{w}^{\mathrm{r}}$ be a $N_{\mathrm{x}} \times N_{\mathrm{x}}$ matrix collecting hidden-hidden weights, and let $\mathbf{w}^{\mathrm{out}}$ a $N_{\mathrm{o}} \times (N_{\mathrm{a}} + N_{\mathrm{x}})$ matrix with the parameters from input and projected space to the output space. The circuits on the reservoir define a dynamical system with the following dynamics:

$$x_m(t) = \psi\Bigg(w^{\mathrm{in}}_{m0} + \sum_{i=1}^{N_{\mathrm{a}}} w^{\mathrm{in}}_{mi} a_i(t) + \sum_{i=1}^{N_{\mathrm{x}}} w^{\mathrm{r}}_{mi} x_i(t-1)\Bigg), \tag{2}$$

for all $m \in [1, N_{\mathrm{x}}]$ where most often $\psi(\cdot)$ is the hyperbolic tangent function ($\tanh(\cdot)$). The output of the model $\mathbf{y}(t) \in \mathbb{R}^{N_{\mathrm{o}}}$ at time t is computed using a linear regression from the reservoir and inputs to the output space, which is expressed as follows:

$$y_s(t) = w^{\mathrm{out}}_{m0} + \sum_{i=1}^{N_{\mathrm{a}}} w^{\mathrm{out}}_{mi} a_i(t) + \sum_{i=1}^{N_{\mathrm{x}}} w^{\mathrm{out}}_{mi} x_i(t), \quad \forall s \in [1, N_{\mathrm{o}}]. \tag{3}$$

In our experimental results we add a new parameter to the model that is called *leaky rate* and is used for controlling the dynamics update [14]. Therefore, the reservoir state in this new ESN model with leaky-integrator neurons is computed as follows. First, a vector state $\mathbf{x}'$ is computed using the expression (2), next the reservoir state is computed following:

$$x_m(t) = (1-\alpha)x'_m(t) + \alpha x_m(t-1), \tag{4}$$

where the parameter α is the leaky-integrator neuron parameter.

The ESN model has some global parameters that impact in its performance. The most relevant ones are: the reservoir size (given by the number of reservoir neurons), the input scaling factor, the spectral radius of the reservoir matrix, the density and topology of the reservoir matrix [13]. A large pool of interconnected reservoir neurons (large reservoir size) improves the linear separability of the projected input patterns. However, a too large reservoir can provoke overfitting. The input scaling factor weighs the input patterns over the reservoir state [15]. In our experiments, we normalise the input data and we consider that all the inputs have equal relevance (equal input scaling factor). The spectral radius controls the stability of the dynamical system and impacts in the memory capacity of the model. According to the Echo State Property [13], the stability of the reservoir state $\mathbf{x}(t)$ only depends of the reservoir weight matrix $\mathbf{w}^{\mathrm{r}}$. The network stability is controlled by the spectral radius of $\mathbf{w}^{\mathrm{r}}$, that we denote by $\rho(\mathbf{w}^{\mathrm{r}})$, if $\rho(\mathbf{w}^{\mathrm{r}}) < 1$ the stability of the ESN can be ensured [13]. As a consequence, the random initialised matrix $\mathbf{w}^{\mathrm{r}}$ is scaled as follows: $\mathbf{w}^{\mathrm{r}} \leftarrow (\beta/\rho(\mathbf{w}^{\mathrm{r}}))\mathbf{w}^{\mathrm{r}}$, where β is a constant in $(0, 1]$. The sparsity of the reservoir matrix is often set on 20% non-zero values.

3 Methodology

For setting up the model parameters, we select three parts of the solar global irradiance time-series which present different trends on one year. The part A has a grown increasing trend in an arbitrary range of time $[a_1, a_2]$ that corresponds to the solar irradiance in days of February and March, the part B does not present any evident trend in the period $[b_1, b_2]$ (solar irradiance in days of May and June), and a part C has a downward trend in $[c_1, c_2]$ (solar irradiance in days of October and November). For each part A, B and C we evaluate the accuracy of several ESNs with different global parameters. For setting the ESN parameters, we forecast the solar power using only information of the past of the solar power series, in other words we do not use any other meteorological variables. At first, we set the global parameters of the ESN. We empirically find the best values of the reservoir size N_{x}, the spectral radius $\rho(\mathbf{w}^{\mathrm{r}})$ and the leaky rate α. The evaluated ESN parameter values were: reservoir size $N_{\mathrm{x}} \in \{20, 30, 50, 70, 100\}$, spectral radius $\rho(\mathbf{w}^{\mathrm{r}}) \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and leaky rate $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Once we obtain the *best* ESN global parameters, denoted by N_{x}^*, ρ^* and α^*, we apply GPSO for automatically selecting other meteorological variables and for improving the forecasting model. As a consequence, we use for estimating the solar irradiance, the precedent information of solar irradiance and a set of other automatically selected meteorological variables. Without loss of generality we enumerate the input features by $\{1, \ldots, M\}$, where M is the number of meteorological variables. A first goal is to find a best combination of meteorological variables for predicting the solar power, this feature selection can not be done using a *brute-force* strategy (due to the size of M). Therefore, we have a searching space $\{0, 1\}^M$, where the feature solutions have the form $\mathbf{s} = [s_1, s_2, \ldots, s_M]$ where $s_i = 0$ represents that the input feature i is omitted for generating the model, and $s_i = 1$ represents that the variable i is an input of the learning tool. For each combination we evaluate the accuracy of an ESN with parameters N_{x}^*, ρ^* and α^*, the objective is to find $\mathbf{s} \in \{0, 1\}^M$ such that the MSE is minimized. We set the parameters of the GPSO following the instructions in [1]. The number of particles is 50, the mutation rate is given by uniform random selecting only one feature s_i at each epoch, and the weights are: $w_1 = 1/6$, $w_2 = 5/12$ and $w_3 = 5/12$. The accuracy evaluation of the proposed method (GPSO-ESN) is made applying *free-run prediction* for the solar irradiance variable. The method uses the real information of the other input features at time t (for instance: temperature) and predicted values of the solar power series during a time windows $[t-k, t]$. As usually, we divide the time-series in two parts. The first part (named training) is used for finding the best configuration of the input features and the best global parameters of the ESN. The second part (named validation) is used for evaluated the adjusted model. We use the fitted model for predicting the values on the validation time-series, and the predicted values of power irradiance as well as the other meteorological variables are used as input patterns for predicting new values. All codes of data processing were carry out with Matlab (Mathworks Inc. Natick, MA, USA).

4 Experimental Results

This section contains two parts, one is devoted to the data description and another one presents the experimental results.

4.1 Data Description

In this paper we are using the meteorological dataset provided by the National Renewable Energy Laboratory and Solar Technology Acceleration Center (SolarTAC) [16]. The collected dataset corresponds to the period started in January

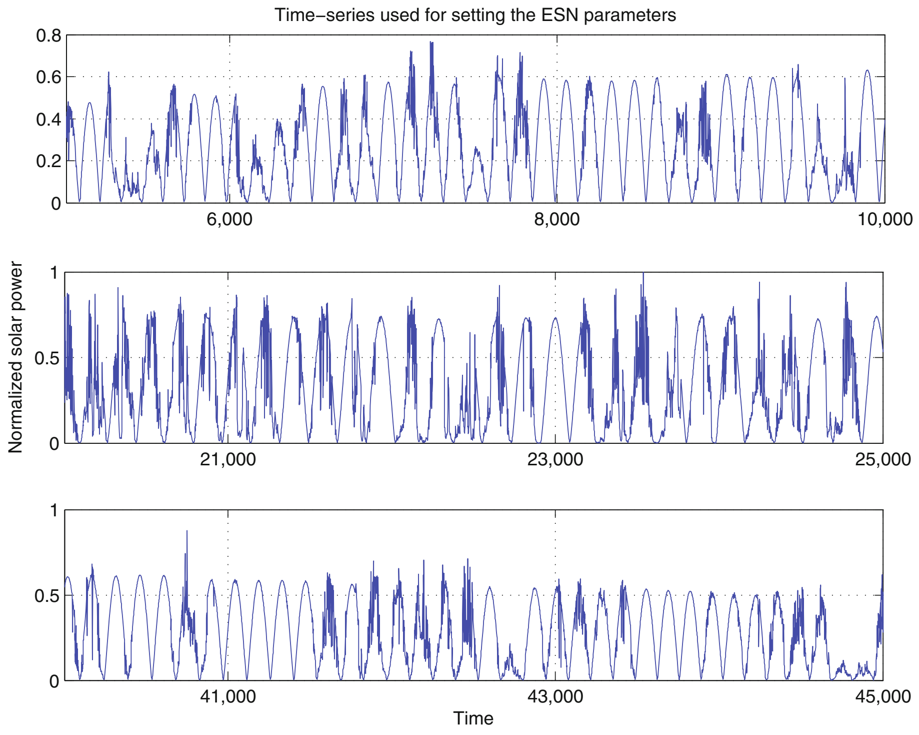


Fig. 1. Training data used for finding the *best* ESN global parameters. The first graphic covers the period since Feb. 3 till Mar. 10, the second graphic covers the period since May 18 till Jun 22, and the third graphics covers the period since Oct. 4 till Nov. 8.

1, 2015 until December 5, 2015. The temporal precision of the data is 1 min. The output variable is the global irradiance given by the *Global Horizontal Irradiance* in W/m^2, and the external input variables are: Air Temperature, Wind Chill

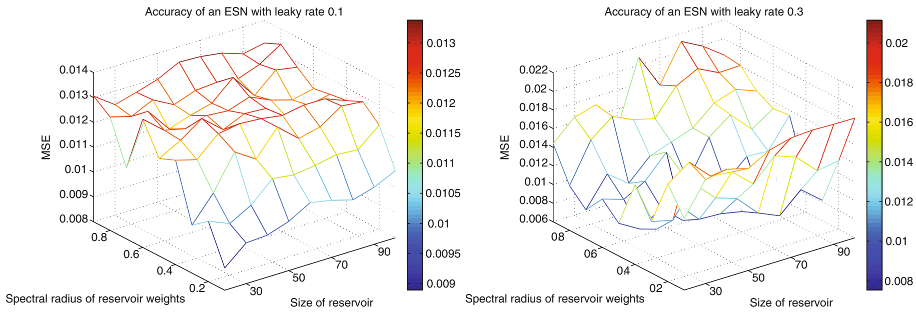


Fig. 2. Sensitivity analysis of the ESN parameters. Example of the accuracy reached by an ESN with leaky rate 0.1 (left side) and leaky rate 0.3 (right side), and $N_{\mathrm{x}} \in [30, 100]$ and $\rho(\mathbf{w}^{\mathrm{r}}) \in [0.1, 0.9]$.

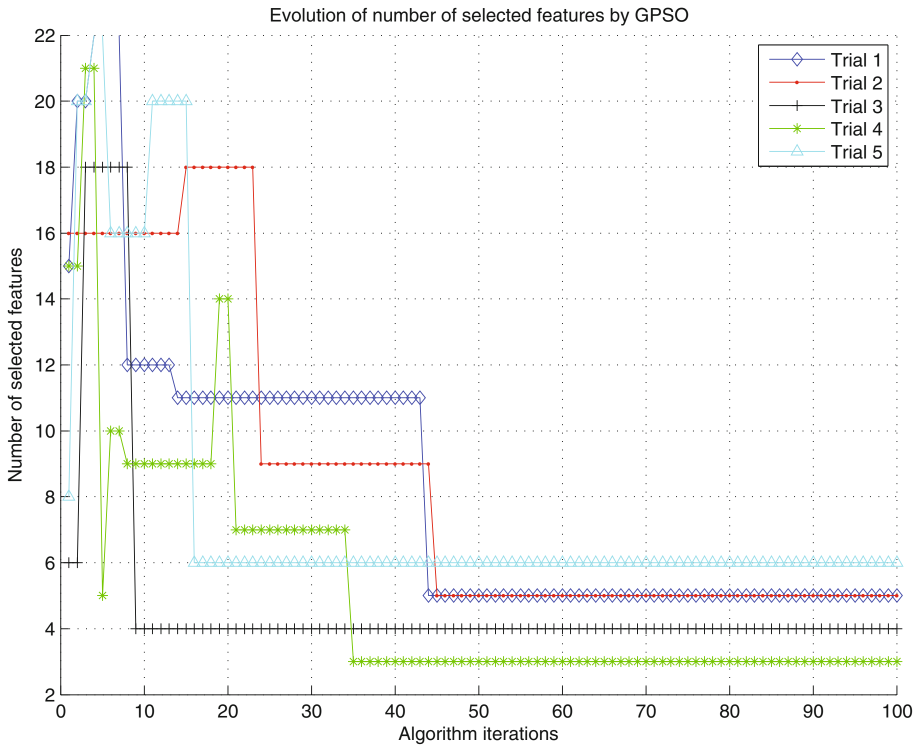


Fig. 3. Example of the evolution of the number of selected features according to the first 100 algorithm iterations.

Temp, Dew Point Temp, Relative Humidity, Wind Speed, Pk Wind Speed, SDev Wind Speed, Wind Direction, Wind Dir at Pk WS, SDev Wind Direction, Station Pressure, Precipitation, Accumulated Precipitation, Zenith Angle, Azimuth Angle, Airmass, CMP22 Temp, CR1000 Temp, CR1000 Battery, CR1000 Process Time. For more information about those variables and the protocol used for collecting them, it can be seen [16]. First at all, we pre-process the data. This consisted in changing the temporal precision from 1 min. to 10 min. The time-series with a new temporal precision covers an arbitrary selected period with 50232 points. In addition, all the variable measures were normalised in $[0, 1]$. Figure 1 presents the three periods of the new pre-processed dataset that are used for setting the parameters of the ESN model. The GPSO can provide several different solutions, due to its stochastic characteristics. Therefore, we evaluate our approach on different 30 initial feature configurations (experiment trials). For each one, we start the GPSO method by randomly selecting a group of the input features.

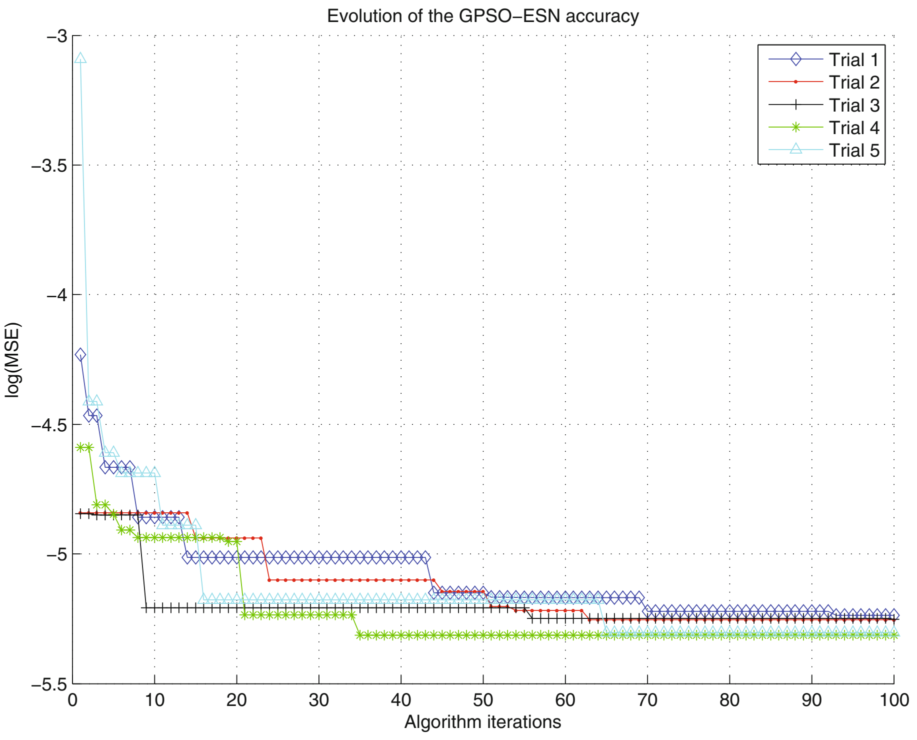


Fig. 4. Example of the evolution of the model accuracy according to the first 100 algorithm iterations.

4.2 Discussion of the Results

Figure 2 shows two graphics, on the left side we can see the MSE reached by an ESN with leaky rate equal to 0.1 in respect of reservoir size and spectral radius of the reservoir matrix. In the right side, it is shown the reached accuracy by an ESN with leaky rate of 0.3 in respect of the other global parameters. We can see how the ESN with a leaky rate with 0.3 reaches higher accuracy, although, on the other side is more unstable according different values of spectral radius and size of reservoir matrix. According to the obtained results for each combination of global parameters on the grid $N_{\mathrm{x}} \in \{20, 30, 50, 70, 100\}$, $\rho(\mathbf{w}^{\mathrm{r}}) \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and leaky rate in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$, we decide to use as *better* global ESN parameters: $N_{\mathrm{x}}^{*} = 40$, $\rho^{*} = 0.7$ and leaky rate $\alpha^{*} = 0.3$. Figure 3 shows the evolution of the number of input features by the GPSO-ESN over iterations. Figure 4 presents the evolution of the accuracy of the model over algorithm iterations. The accuracy is presented using logarithmic scale. In last two figures, for a better visibility we present only 5 randomly selected trials of the GPSO-ESN among the 30 experiments. The best configuration of external input features found by the GPSO technique was composed by the following variables: air temperature, wind chill temperature, wind speed, Zenith angle, and the temperature of CR1000 datalogger panel.

5 Conclusions and Future Work

In this article we present a procedure for predicting the solar power irradiance using several meteorological variables. Our approach can help to optimize energy yield from photovoltaic systems. The approach uses the recently introduced technique *Geometric Swarm Optimization (GPSO)* for selecting the most significant input features. The forecast is done using a specific type of Recurrent Neural Network named *Echo State Networks (ESN)*. We evaluate the proposed method over a real meteorological dataset provided by the Solar Technology Acceleration Center (SolarTAC), Colorado, USA. The advantage of our approach consists in using an automatic optimisation technique for finding a *good* combination of meteorological variables, which affect the solar power estimation. We are interested in the near future to analyse the group of meteorological variables computed by the GPSO with other feature selection techniques, as well as to extend the period used for training the network model.

Acknowledgement. This work is partially supported by Grant of SGS No. SP2016/97, VŠB - Technical University of Ostrava, Czech Republic.

References

1. Moraglio, A., Di Chio, C., Togelius, J., Poli, R.: Geometric particle swarm optimization. J. Artif. Evol. Appl. **2008**, 11:1–11:14 (2008). http://dx.doi.org/10.1155/2008/143624

2. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
3. Friedrich, T., He, J., Jansen, T., Moraglio, A.: Genetic and evolutionary computation. Theory Comput. Sci. **561**, 1–2 (2015). http://dx.doi.org/10.1016/j.tcs.2014.11.022
4. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. German National Research Center for Information Technology. Technical report 148 (2001)
5. Chen, C., Duan, S., Cai, T., Liu, B.: Online 24-h solar power forecasting based on weather type classification using artificial neural network. Sol. Energy **85**, 2856–2870 (2011)
6. Letendre, S., Makhyoun, M., Taylor, M.: Predicting solar power production: irradiance forecasting models, applications and future prospects. Solar Electric Power Association, Washington, DC, USA, Technical report, March 2014. www.solarelectricpower.org
7. Pelland, S., Remund, J., Kleissl, J., Oozeki, T., Brabandere, K.D.: Photovoltaic and solar forecasting: state of art. International Energy Agency Photovoltaic Power Systems Programme, Technical report IEA-PVPS T14-01:2013 (2013). http://www.iea-pvps.org
8. Basterrech, S., Prokop, L., Burianek, T., Misak, S.: Optimal design of neural tree for solar power prediction. In: Proceedings of the 2014 15th International Scientific Conference on Electric Power Engineering (EPE), pp. 273–278, May 2014
9. Basterrech, S., Zjavka, L., Prokop, L., Misak, S.: Irradiance prediction using echo state queueing networks and differential polynomial neural networks. In: 2013 13th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 271–276, December 2013
10. Pedro, H.T., Coimbra, C.F.: Assessment of forecasting techniques for solar power production with no exogenous inputs. Sol. Energy **86**(7), 2017–2028 (2012)
11. Bacher, P., Madsen, H., Nielsen, H.A.: Online short-term solar power forecasting. Sol. Energy **83**, 1772–1783 (2009)
12. 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)
13. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**, 127–149 (2009)
14. Jaeger, H., Lukoševičius, M., Popovici, D., Siewert, U.: Optimization and applications of Echo State Networks with leaky-integrator neurons. Neural Netw. **20**(3), 335–352 (2007)
15. Butcher, J.B., Verstraeten, D., Schrauwen, B., Day, C.R., Haycock, P.W.: Reservoir computing and extreme learning machines for non-linear time-series data analysis. Neural Netw. **38**, 76–89 (2013)
16. Andreas, A., Wilcox, S.: Aurora, colorado (data). Solar Technology Acceleration Center (SolarTAC), Colorado, USA, Technical report DA-5500-56491 (2011). doi:10.5439/1052224

WalkSAT Based-Learning Automata for MAX-SAT

N. Bouhmala[1](✉), M. Oseland[2], and Ø. Brådland[2]

[1] Department of Maritime Technology and Innovation, University College of Southeast Norway, Notodden, Norway
noureddine.bouhmala@usn.no

[2] Department of Technology, Agder University, Kristiansand, Norway

Abstract. Researchers in artificial intelligence usually adopt the Satisfiability paradigm as their preferred methods when solving various real worlds decision making problems. Local search algorithms used to tackle different optimization problems that arise in various fields aim at finding a tactical interplay between diversification and intensification to overcome local optimality while the time consumption should remain acceptable. The WalkSAT algorithm for the Maximum Satisfiability Problem (MAX-SAT) is considered to be the main skeleton underlying almost all local search algorithms for MAX-SAT. This paper introduces an enhanced variant of WalkSAT using Finite Learning Automata. A benchmark composed of industrial and random instances is used to compare the effectiveness of the proposed algorithm against state-of-the-art algorithms.

Keywords: Walksat · Learning automata · Combinatorial optimization

1 Introduction

The satisfiability problem (SAT) which is known to be NP-complete [6] plays a central role problem in many applications in the fields of VLSI Computer-Aided design, Computing Theory, Artificial Intelligence and defence. SAT is an increasingly used paradigm that can model a wide spectrum of combinatorial optimization problems. It has become an important field of study in both theoritical and applied computer science. Generally, a SAT problem is defined as follows. A propositional formula $\Phi = \bigwedge_{j=1}^{m} C_j$ with m clauses and n Boolean variables is given. Each Boolean variable, takes one of the two values, True or False. A clause, in turn, is a disjunction of literals and a literal is a variable or its negation. Each clause C_j has the form:

$$C_j = \left(\bigvee_{k \in I_j} x_k \right) \vee \left(\bigvee_{l \in \bar{I}_j} \bar{x}_l \right),$$

where $I_j, \bar{I}_j \subseteq \{1, ...n\}$, $I \cap \bar{I}_j = \emptyset$, and $\bar{x}_i$ denotes the negation of x_i. The task is to determine whether there exists an assignment of values to the variables under which Φ evaluates to True. Such an assignment, if it exists, is called a satisfying assignment for Φ, and Φ is called satisfiable. Otherwise, Φ is said to be unsatisfiable. The maximum satisfiability problem is the optimization variant of SAT. More formally, let w_i denote the weight of clause C_i. Then Eq. 1 is the objective function to be maximized, with $S(C_i)$ is equal to 1 when C_i is true and 0 otherwise.

$$\sum_{k=1}^{m} w_i \cdot S(C_i), \tag{1}$$

There exists two important variations of the Max-SAT problem. The weighted Max-SAT problem is the Max-SAT problem in which each clause is assigned a positive weight. The goal of the problem is to maximize the sum of weights of satisfied clauses. The unweighted Max-SAT problem is the Max-SAT problem in which all the weights are equal to 1 and the goal is to maximize the number of satisfied clauses. In this paper, the focus is restricted to formulas in which all the weights are equal to 1 (i.e.unweighted Max-SAT). Several state-of-the-art local search algorithms are enhanced versions of WalkSAT (WSAT) [15] algorithm. Examples include WalkSAT/Tabu [13], Novelty+ and R-Novelty+ heuristics [10]. WalkSAT starts by generating a random initial assignment. The next step of the algorithm involves picking randomly an unsatisfied clause. If there exists a variable with break count equals to zero, this variable is flipped, otherwise a random variable or the variable with minimal break count is selected with a certain probability (noise probability). The break count of a variable is defined as the number of clauses that would be unsatisfied by flipping the chosen variable. It turns out that the choice of unsatisfied clauses, combined with the randomness in the selection of variables, can enable WalkSat to avoid local minima and to better explore the search space. The flips are repeated until a pre-set value of the maximum number of flips is reached and this phase is repeated as needed up to a certain number of times. While The aforementioned meta-heuristics, work only with a single neighborhood structure, other meta-heuristics choose to operate on a set of different neighborhood structures giving rise to Variable neighborhood search algorithms [2,3,9]. To avoid manual parameter tuning, new methods have been designed to automatically adapt parameter settings during the search [12]. As the quality of the solution improves when larger neighborhood is use, the work proposed in [19] uses a restricted 2 and 3-flip neighborhoods and better performance has been achieved compared to the 1-flip neighborhood for structure problems. Clause weighting based SLS algorithms [4] have been proposed to solve SAT and Max-SAT problems. Although these clause weighting SLS algorithms differ in the manner clause weights should be updated (probabilistic or deterministic) they all choose to increase the weights of all the unsatisfied clauses as soon as a local minimum is encountered. In [16], a stochastic local search algorithm, called Iterated Robust Tabu Search (IRoTS), was presented for MAX-SAT that combines an Iterated Local Search and Tabu Search. Finally, a new strategy based on an automatic procedure for integrating selected components from various existing solvers have been devised in order to build new efficient algorithms that draw the strengths of multiple algorithms [11]. The work conducted in [20] proposed an adaptive memory based local search algorithm that exploits various strategies in order to guide the search to achieve a suitable trade-off between intensification and diversification. The computational results show that it competes favorably with some state-of-the-art MAX-SAT solvers.

2 The Algorithm

We base our work on the principles of Learning Automata [17]. Learning Automata have been used to model biological systems [18], and have recently attracted considerable interest because they can learn the optimal actions when operating in (or interacting with) unknown stochastic environments. Furthermore, they combine rapid and accurate convergence with low computational complexity. Learning Automata solutions have been proposed for several other combinatorial optimization problems [8] The work reported in [7] was the first to combine the traditional random walk with learning automata for the satisfiability problem. Inspired by the success of the above solution scheme, we will in the following propose how the classical GSAT-Random-Walk algorithm can be enhanced with learning capability, using Learning Automata.

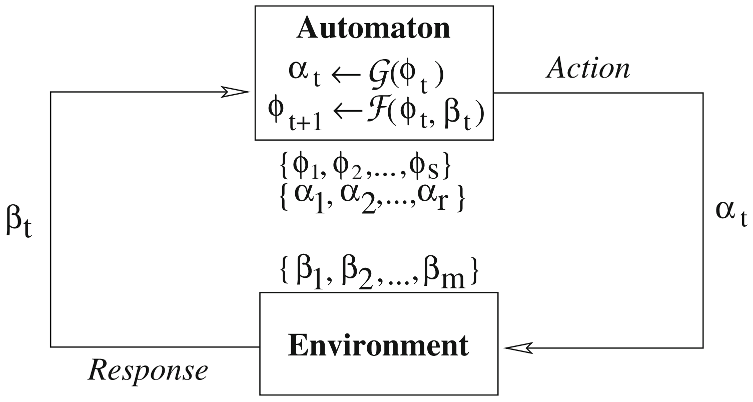

Automaton
$\alpha_t \leftarrow \mathcal{G}(\phi_t)$
$\phi_{t+1} \leftarrow \mathcal{F}(\phi_t, \beta_t)$
Action
$\{\phi_1, \phi_2, \ldots, \phi_s\}$
$\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$
β_t
α_t
$\{\beta_1, \beta_2, \ldots, \beta_m\}$
Environment
Response

Fig. 1. A learning automaton interacting with an environment

2.1 The Automata and Its Environment

Generally stated, a finite learning automaton performs a sequence of actions on an *environment*. The environment can be seen as a generic *unknown* medium that responds to each action with some sort of reward or penalty, perhaps *stochastically*. Based on the responses from the environment, the aim of the finite learning automaton is to find the action that minimizes the expected number of penalties received. Figure 1 illustrates the interaction between the finite learning automaton and the environment. Because we treat the environment as unknown, we will here only consider the definition of the finite learning automaton. The finite learning automaton can be defined in terms of a quintuple [14]:

$$\{\underline{\Phi}, \underline{\alpha}, \underline{\beta}, \mathcal{F}(\cdot,\cdot), \mathcal{G}(\cdot,\cdot)\}.$$

$\underline{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_s\}$ is the set of internal automaton states. $\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of automaton actions. And, $\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the set of inputs that can be given to the automaton. An output function $\alpha_t = \mathcal{G}[\phi_t]$ determines the next action performed by the automaton given the current automaton state. Finally, a transition function $\phi_{t+1} = \mathcal{F}[\phi_t, \beta_t]$ determines the new automaton state from (1) the current automaton state and (2) the response of the environment to the action performed by the automaton. Based on the above generic framework, the crucial issue is to design automata that can learn the optimal action when interacting with the environment. In this paper we target the SAT problem, and our goal is to design a team of Learning Automata that seeks the solution of SAT problem instances. We build upon the work of Tsetlin and the linear two-action automaton [14,18]. For each literal in the SAT problem instance that is to be solved, we construct an automaton with

- States: $\underline{\Phi} = \{-N-1, -N, \ldots, -1, 0, \ldots, N-2, N\}$.
- Actions: $\underline{\alpha} = True, False$
- Inputs: $\underline{\beta} = \{reward, penalty\}$.

Figure 2 specifies the $\mathcal{G}$ and $\mathcal{F}$ matrices. The $\mathcal{G}$ matrix can be summarized as follows. If the automaton state is positive, then action True will be chosen by the automaton. If on the other hand the state is negative, then action False will be chosen. Note that since we initially do not know which action is optimal, we set the initial state of the Learning SAT Automaton randomly to either '*−1*' or '*0*'.

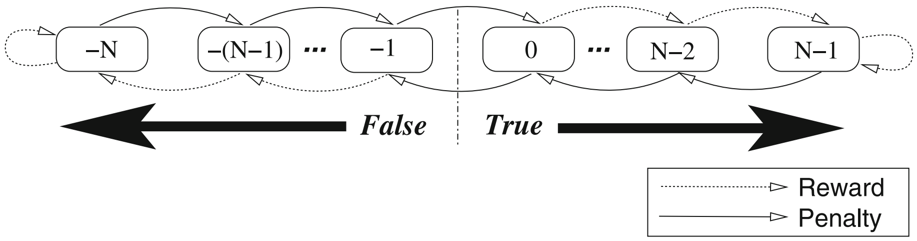


Fig. 2. The state transitions and actions of the learning SAT automaton

The state transition matrix $\mathcal{F}$ determines how learning proceeds. As seen in the figure, providing a *reward* input to the automaton *strengthens* the currently chosen action, essentially by making it less likely that the other action will be chosen in the future. Correspondingly, a *penalty* input *weakens* the currently selected action by making it more likely that the other action will be chosen later on. In other words, the automaton attempts to incorporate past responses when deciding on a sequence of actions.

2.2 WalkSAT Based Learning Automata (LA-WSAT)

Overview: In addition to the definition of the LA, we must define the environment that the LA interacts with. Simply put, the environment is a MAX-SAT problem instance as defined in Sect. 1. Each variable of the MAX-SAT problem instance is assigned a dedicated LA, resulting in a team of LA. The task of each LA is to determine the truth value of its corresponding variable, with the aim of satisfying all of the clauses where that variable appears. In other words, if each automaton reaches its own goal, then the overall MAX-SAT problem at hand has also been solved.

Pseudo-code: With the above perspective in mind, we will now present the details of the LA-WSAT that we propose. Algorithms 2–4 contain the complete pseudo-code for solving MAX-SAT problem instances, using a team of LA. An ordinary WSAT strategy is used to penalize an LA when it "disagrees" with WSAT, i.e., when WSAT and the LA suggest opposite truth values. Additionally, we use an "inverse" WSAT strategy for rewarding an LA when it agrees with WSAT. Note that as a result, the assignment of truth values to variables is indirect, governed by the states of the LA. At the core of the LA-WSAT algorithm is a punishment/rewarding scheme that guides the team of LA towards the optimal assignment. In the spirit of automata based learning, this scheme is incremental, and learning is performed gradually, in small steps.

Remark 1: Like a two-action Tsetlin Automaton, our proposed LA seeks to minimize the expected number of penalties it receives. In other words, it seeks finding the truth assignment that minimizes the number of unsatisfied clauses among the clauses where its variable appears.

Remark 2: Note that because multiple variables, and thereby multiple LA, may be involved in each clause, we are dealing with a game of LA [14]. That is, multiple LA interact with the same environment, and the response of the environment depends on the actions of several LA. In fact, because there may be conflicting goals among the LA involved in the LA-WSAT, the resulting game is competitive. The convergence properties of general competitive games of LA have not yet been successfully analyzed, however, results exists for certain classes of games, such as the Prisoner's Dilemma game [14]. In our case, the LA involved in the LA-WSAT are non-absorbing, i.e., every state can be reached from every other state with positive probability. This means that the probability of reaching the solution of the SAT problem instance at hand is equal to 1 when running the game infinitely. Also note that the solution of the MAX-SAT problem corresponds to a Nash equilibrium of the game.

Remark 3: In order to maximize speed of learning, we initialize each LA randomly to either the state '-1' or '0'. In this initial configuration, the variables will be flipped relatively quickly because only a single state transition is necessary for a flip. Accordingly, the joint state space of the LA is quickly explored in this configuration. However, as learning proceeds and the LA move towards their boundary states, i.e., states '$-N$' and '$N-1$', the flipping of variables calms down. Accordingly, the search for a solution to the MAX-SAT problem instance at hand becomes increasingly focused.

```
input  : Problem in CNF format
output: Number of Unsatisfied clauses
begin
    for $i \leftarrow 1$ to $n$ do
        /* The initial state of each automaton is set to either '-1' or '1' */
        state[i] = random_element($\{-1, 0\}$);
        /* And the respective literals are assigned corresponding truth values */
        if (state[i] == -1) then
            $x_i$ = False;
        else
            $x_i$ = True;
        end
    end
    while (Not Stop) do
        $C_k$ ←Random-Unsatisfied-Clause ();
        if ($\exists$ a literal $\in C_k$ with breakcount = 0) then
            chosen-literal ←Random-Chosen-Literal-Candidate($C_k$) ;
            Reward-LA (LA,chosen-literal) ;
        end
        else if ($random(0,1) \leq p_{noise}$) then
            chosen-literal ←Random-Literal($C_k$);
            Punish-LA (LA,chosen-literal) ;
        end
        else
            chosen-literal ←Random-Lowest-Breakcount($C_k$) ;
            Reward-LA(LA,chosen-literal);
        end
    end
end
```

Algorithm 1. Walksat-Learning-Automata Based Algorithm

3 Experimental Results

3.1 Test Suite and Parameter Settings

The performance of LA-WSAT is evaluated against WSAT using a set of real industrial problems and random problems. This set is taken from the sixth Max-SAT 2011 organized as an affiliated event of the Fourteenth International Conference on Theory and Applications of Satisfiability Testing (SAT-2011). Due to

```
input  : A literal and its learning automata
output: Penalize the learning automata
begin
    if (state[i] < N -1 ) then
        state[i] ← $state[i] + 1$;
    end
    /* Flip literal when automaton changes its action */ ;
    if (state[i]== 0) then
        Flip ($l_i$) ;
    end
    if (state[i] > -N ) then
        state[i] ← $state[i] - 1$;
    end
    /* Flip literal when automaton changes its action */ ;
    if (state[i]== -1) then
        Flip ($l_i$) ;
    end
end
```

Algorithm 2. Punishment Mechanism

```
input  : A literal and its learning automata
output: Reward the learning automata
begin
    if (state[i] $\geq$ 0 And state[i] < N-1) then
        state[i] ← $state[i] + 1$;
    end
    if (state[i] < 0 And state[i] < N-1) then
        state[i] ← $state[i] - 1$;
    end
end
```

Algorithm 3. Reward Mechanism

the randomization nature of both algorithms, each problem instance was run 50 times with a cut–off parameter (max-time) set to 30 min. The tests were carried out on a DELL machine with 800 MHz CPU and 2 GB of memory. The code was written in C++ and compiled with the GNU C compiler version 4.6. The following parameters have been fixed experimentally and are listed below:

- The number of states N is set to 3.
- Noise probability: Peak performance with respect to the lowest number of unsatisfied clauses is achieved when the walking probability was set to 10.

3.2 Results and Discussions

Figure 3 shows the evolution of the mean of unsatisfied clauses for both algorithm as a function of time on a logarithmic scale. Both algorithms start with

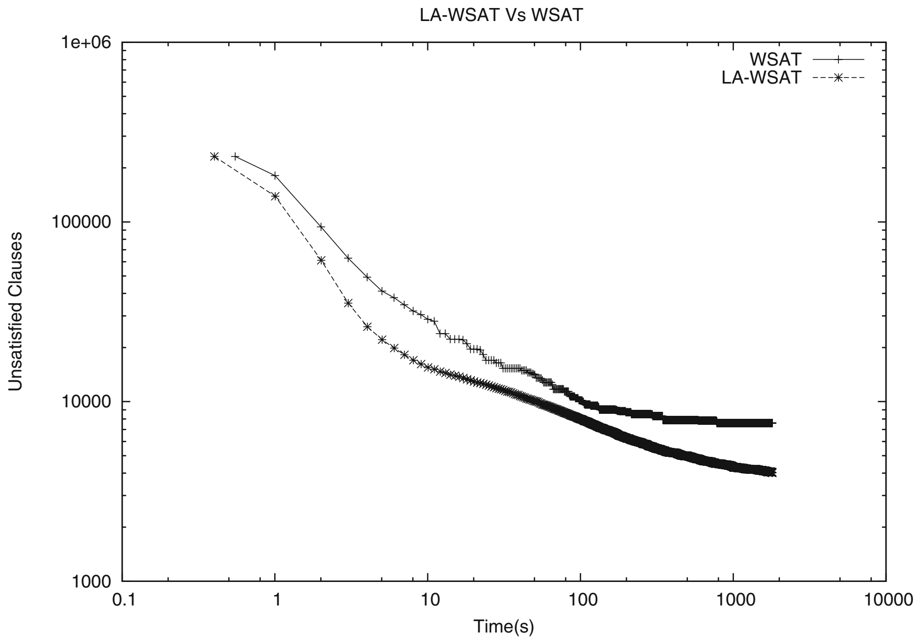


Fig. 3. Log-Log plot: C2-DD-S3-f1-e2-v1-bug-fourvec-gate.dimacs.seq.filtered.cnf: $|V| = 400085$, $|C| = 1121810$.

Table 1. SAT2013 Industrial benchmarks: LA-WSAT Vs WSAT

Instances	$\|V\|$	$\|C\|$	WSAT	LA-WSAT
fpu-multivec1-problem.dimacs-14.filtered	257168	928310	6125	1769
fpu-fsm1-problem.dimacs15.filtered	160200	548843	3716	422
fpu8-problem.dimacs24.filtered	160200	548843	3585	412
i2c-problem.dimacs.filtered	521672	1581471	615	161
b15-bug-fourvec-gate-0.dimacs	581064	1712690	7241	1569
c1-DD-s3-f1-e2-v1-bug-fourvec-gate-0.dimacs	391897	989885	955	41
c4-DD-s3-f1-e1-v1-bug-gate-0.dimacs	797728	2011216	3761	1911
c4-DD-s3-f1-e2-v1-bug-fourvec-gate-0.dimacs	448465	1130672	1834	640
c5-DD-s3-f1-e1-v2-bug-gate-0.dimacs	200944	540984	8	8
c6-DD-s3-f1-e1-v1-bug-gate-0.dimacs	298058	795900	3188	1827
divider-problem.dimacs11.filtered	215964	709377	9992	2675
divider-problem.dimacs2.filtered	228874	750705	10688	3073
divider-problem.dimacs3.filtered.cnf	216900	711249	6204	2117
divider-problem.dimacs5.filtered	228874	750705	11194	3271
divider-problem.dimacs8.filtered	246943	810105	7257	3363

almost an identical initial solution. Already during the early stage of the search, the difference in solution quality between the two algorithms start to become more distinctive. In the first phase which corresponds to the early part of the search, both algorithms behave as a hill-climbing method. The mean number of unsatisfied clauses decreases rapidly at first, and then flattens off when entering the so-called a plateau region marking the start of the second phase. The plateau region spans a region in the search space where flips typically leave the number of unsatisfied clauses unchanged, and this phenomenon occurs earlier with WSAT leading to a possibly premature convergence. From these plots, LA-WSAT algorithm dominates WSAT throughout the run offering a better asymptotic convergence compared to WSAT. Tables 1 and 2 compares LA-WSAT against WSAT using industrial instances. The first and second column show the number of variables and clauses respectively. LA-WSAT is capable of delivering solutions of excellent quality compared to WSAT. LA-WSAT dominates WSAT in 26 cases out 30 cases. The difference in the quality ranges within 10% for 7% of the cases, 20% for 5% of the cases, and above 30% for the remaining cases, getting as high as 96%. Table 3 and 4 compares LA-WSAT with highly efficient solvers such as CCLS [5] and Optimax which is a modified version of glucose SAT solver [1] ranked 1st at the 2011 SAT competition. CCLS won four categories of the incomplete algorithms track of MaxSAT Evaluation 2013. The instances used in the benchmark belong to random and crafted categories used

Table 2. SAT2013 Industrial benchmarks: LA-WSAT Vs WSAT

Instances	$\|V\|$	$\|C\|$	WSAT	LA-WSAT
dividers10.dimacs.filtered	45552	162874	694	64
dividers-multivec1.dimacs.filtered	106128	397650	3015	361
i2c-problem.dimacs25	521672	1581471	2686	2498
i2c-master1.dimacs.filtered.cnf	82429	285987	317	95
mim-ctr1-dimacs.filtered	1128648	4422185	10198	1275
rsdecoder1-blackboxKESblock-problem.dimacs	707330	1106376	3757	3280
rsdecoder1-blackbox-CSEEblock	277950	806460	3479	1868
rsdecoder4.dimacs.filtered	237783	933978	277	928
rsdecoder5.dimacs.filtered	238290	936006	280	708
rsdecoder-debug.dimacs	847501	2223029	19654	7746
rsdecoder-fsm2.dimacs.filtered	238290	936006	247	651
rsdecoder-multivec1.dimacs.filtered	394446	1626312	2143	4441
rsdecoder-multivec1-problem.dimacs38.filtered	1199012	3865513	29810	19318
rsdecoder-problem.dimacs39.filtered	1199602	3868693	28425	18005
rsdecoder-problem.dimacs41.filtered	1186710	3829036	27442	18328

Table 3. Comparing LA-WSAT with CCLS and Optimax

Instance	CCLS		Optimax		LA-WSAT	
	Quality	Time	Quality	Time	Quality	Time
MANNa-27.clq	404	0.29	486	0.10	404	1.08
MANNa-45.clq	418	0.19	518	0.13	418	1.02
MANNa-81.clq	399	0.06	441	0.12	399	1.08
MANN-a9.clq	422	1.16	584	0.10	422	12.03
brock200-1.clq	238	0.95	349	0.10	238	37.03
brock200-2.clq	141	1.11	221	0.13	141	4.01
brock200-3.clq	214	0.96	232	1.75	214	6.53
brock-200-4.clq	209	1.04	299	0.11	209	4.40
brock400-1	255	0.38	340	0.08	256	13.02
brock400-2	252	0.84	310	0.08	252	9.89
brock400-3	238	1.27	278	0.08	239	17.01
brock400-4	249	0.73	374	0.08	250	7.61
brock800-1	205	0.95	273	0.09	205	2.18
brock800-2	207	0.97	270	0.09	207	7.05
brock800-3	203	0.47	315	0.07	203	1.08
brock800-4	200	0.32	310	0.13	200	4.27
cfat200-1.clq	4	0.01	4	0.01	4	1.07
cfat200-2.clq	26	0.71	26	3.93	26	1.2
cfat200-5.clq	116	0.47	172	0.08	116	1.8
c-fat500-1.clq	2	0.01	2	0.01	2	10.04

at SAT2013 competition. Compared to CCLS, LA-WSAT gave similar results in 35 cases out of 40 cases. However the time of CCLS is several order of magnitude faster that of LA-WSAT. The remaining cases where LA-WSAT was beaten, the difference in quality ranges from 1% and 6%. Another interesting remark to mention is that the time required by LA-WSAT does vary significantly depending on the problem instance while the variations observed with CCLS remain very low. The comparison between Optimax and LA-WSAT shows that Optimax converges very fast at the expense of delivering solutions of poor quality compared to LA-WSAT. LA-WSAT was capable of delivering solutions of better quality than Optimax in 35 out of 40 cases. The improvement ranges from 5% and 44%. The cases where LA-WSAT and Optimax gave similar results, Optimax was several order of magnitude faster.

Table 4. Comparing LA-WSAT with CCLS and Optimax

Instance	CCLS		Optimax		LA-WSAT	
	Quality	Time	Quality	Time	Quality	Time
hamming10-2	400	0.09	532	0.08	400	1.08
hamming10-4	319	0.42	341	0.09	320	5.05
hamming6-2	832	1.18	1100	0.13	844	30.04
hamming6-4	192	1.00	312	0.13	192	1.06
hamming8-2	441	0.12	551	0.13	441	1.09
hamming8-4	176	1.17	254	0.14	176	4.70
johnson16-2-4.clq	215	0.68	344	0.08	215	1.07
johnson32-2-4.clq	329	1.51	492	0,09	329	16.09
johnson8-2-4.clq	75	0.34	100	0.08	75	1.09
johnson8-4-4.clq	770	1.59	1069	0.09	779	90.04
keller4.clq	199	0.74	344	1.01	199	2.02
keller5.clq	250	0.49	317	1.01	250	12.87
p-hat1000-1.clq	52	0.54	52	8.59	52	1.10
p-hat1000-2.clq	142	0.79	181	0.13	142	56.04
p-hat1000-3.clq	238	1.04	362	0.09	238	8.33
p-hat300-1.clq	49	0.50	49	5.38	49	2.19
p-hat300-2.clq	135	0.80	184	0.08	135	70.06
p-hat300-3.clq	269	1.41	327	0.12	269	69.06
phat500-1.clq	75	0.50	108	0,08	75	4.02
phat500-2.clq	176	0.59	244	0.09	176	65.01

4 Conclusions

In this work, a new approach based on combining learning Finite automaton with Walksat for MAX-SAT is introduced. Thus, in order to get a comprehensive picture of the new algorithms performance, a set of large industrial instances is used. The results indicate that learning finite automaton can enhance the convergence behavior of the walksat algorithm. It appears clearly from the results that LA-WSAT can find excellent solutions compared to those of WSAT at a faster convergence rate. The results have shown that in most of the studied cases, LA-WSAT is capable of delivering solution of similar quality compared to CCLS while the time invested is several order of magnitude slower than CCLS. When compared to Optimax, LA-WSAT showed a better performance as it provides solution of better quality. For the time being, further work is mainly conducted on improving the solution quality of LA-WSAT by combining it with the multilevel paradigm. The approach suggests looking at the solution of the problem as a multilevel process operating in a coarse-to-fine strategy. This strategy involves recursive coarsening to create a hierarchy of increasingly smaller and coarser versions of the original problem. The reduction phase works by grouping the variables representing the problem into clusters. This phase is repeated until the size of the smallest problem falls below a specified reduction threshold. A solution for the problem at the coarsest level is generated, and then successively projected back onto each of the intermediate levels in reverse order. The solution at each child level is improved using LA-WSAT before moving to the parent level.

References

1. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009), July 2009
2. Bouhmala, N.: A variable neighborhood search structure based-genetic algorithm for combinatorial optimization problems. Int. J. Hybrid Intell. Syst. Theor. Appl. **15**(2), 127–146 (2016)
3. Bouhmala, N.: Variable neighborhood walksat-based algorithm for MAX-SAT problems. Sci. World J. **2014**, 11 (2014). Article ID 798323
4. Frank, J.: Learning short-term clause weights for GSAT. In: Proceedings of IJCAI 1997, pp. 384–389. Morgan Kaufmann Publishers (1997)
5. Cai, S., Luo, C., Su, K.: CCASat: solver description. In: Proceedings of SAT Challenge 2012: Solver and Benchmark Descriptions, pp. 13–14 (2012)
6. Cook, S.: The complexity of theorem-proving procedures. In: Proceedings of the Third ACM Symposium on Theory of Computing, pp. 151–158 (1971)
7. Granmo, O.C., Bouhmala, N.: Solving the satisfiability problem using finite learning automata. Int. J. Comput. Sci. Appl. **4**(3), 15–29 (2007). Special Issue on Natural Inspired Computation
8. Granmo, O.C., Oommen, B.J., Myrer, S.A., Olsen, M.-G.: Learning automata-based solutions to the nonlinear fractional Knapsack problem with applications to optimal resource allocation. IEEE Trans. Syst. Man Cybern. **SMC–37(B)**, 166–175 (2007)
9. Hansen, P., Jaumard, B., Mladenovic, N., Parreira, A.D.: Variable neighborhood search for maximum weighted satisfiability problem. Technical report G-2000-62, Les Cahiers du GERAD, Group for Research in Decision Analysis (2000)
10. Hoos, H.: On the run-time behavior of stochastic local search algorithms for SAT. In: Proceedings of AAAI 1999, pp. 661–666 (1999)
11. KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: SATenstein: automatically building local search SAT solvers from components. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2009) (2009)
12. Li, C.M., Wei, W., Zhang, H.: Combining adaptive noise and look-ahead in local search for SAT. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 121–133. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72788-0_15
13. McAllester, D., Selman, B., Kautz, H.: Evidence for invariants in local search. In: Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI 1997), pp. 321–326 (1997)
14. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice Hall, Upper Saddle River (1989)
15. Selman, B., Kautz, H.A., Cohen, B.: Noise strategies for improving local search. In: Proceedings of AAAI 1994, pp. 337–343. MIT Press (1994)

16. Smyth, K., Hoos, H., *Stützle*, T.: Iterated robust tabu search for MAX-SAT. Lecture Notes in Artificial Intelligence, vol. 2671, pp. 129–144 (2003)
17. Thathachar, M.A.L., Sastry, P.S.: Network of Learning Automata: Techniques for on Line Stochastic Optimization. Kluwer Academic Publishers, Boston (2004)
18. Tsetlin, M.L.: Automaton Theory and Modeling of Biological Systems. Academic Press, New York (1973)
19. Yagiura, M., Ibaraki, T.: Efficient 2 and 3-flip neighborhood search algorithms for the MAX SAT: experimental evaluation. J. Heuristics **7**(5), 423–442 (2001)
20. Zhipeng, L., Jin-Kao, H.: Adaptive memory-based local search for MAX-SAT. Appl. Soft Comput. **12**(8), 2063–2071 (2012)

A Self-adaptive Artificial Bee Colony Algorithm with Incremental Population Size for Large Scale Optimization

Doğan Aydın(✉) and Gürcan Yavuz

Department of Computer Engineering, Dumlupinar University, 43000 Kutahya, Turkey
{dogan.aydin,gurcan.yavuz}@dpu.edu.tr

Abstract. Large scale optimization is challenging area due to the curse of dimensionality. As a result of the increase of the problem space, computational cost becomes expensive and performance of the optimization algorithms decrease. To overcome this problem, a self-adaptive Artificial Bee Colony (ABC) algorithm called "Self-adaptive Search Equation-based Artificial Bee Colony" (SSEABC) is proposed in this paper. In SSEABC, the canonical ABC is modified with two strategies which are self-adaptive search equation determination and incremental population size. The first strategy determines the appropriate search equations for a given problem instance adaptively during execution. On the other hand, incremental population size strategy adds new food sources to the population biased towards the best-so-far solution. This leads to performance improvement. SSEABC was tested on the benchmark set provided for the special issue of Soft Computing Journal on "Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems" (SOCO). We compared the results of the proposed algorithm to the five ABC variants and the fifteen SOCO participant algorithms. The comparison results indicate that SSEABC is more effective than the considered ABC variants and very competitive with regards to the fifteen SOCO competitor algorithms.

Keywords: Artificial Bee Colony · Large scale continuous optimization · Self-adaptation

1 Introduction

Many evolutionary algorithms (EA) have been studied intensively by researchers. These algorithms may give good results for low dimensional problems. However for Large Scale Global Optimization (in shortly LSGO) problems, the performance degradation may be occurred because of high dimensionality.

In recent years, LSGO problems are attracted researchers' attention because of the problem nature. LSGO problems are hard to solve and many real-world problems such as gene recognition, data mining and large electronic system design can be put into LSGO category. Therefore, several algorithms such as Cooperative coevolution [24,33], Differential Evolution (DE) [18,30], Genetic Algorithms (GA) [24] and Particle Swarm Optimization algorithms (PSO) [15,20,26] have been proposed to tackle LSGO problems.

Artificial Bee Colony Algorithm (ABC) is a meta-heuristic method initially for the numerical function optimization and they can also be applied on LGSO problems. It is a Swarm Intelligence (SI) technique which was proposed by Derviş Karaboğa in 2005 by imitating the foraging behavior of honey bee colonies in the nature [16]. The algorithm consists of four steps: Initialization, employed bees, onlooker bees and scout bees. The crucial steps of ABC are employed and onlooker bees which dominate the search behavior of the algorithm. The search equations used in these steps control the balance between exploration and exploitation. Hence, various variants of ABC based on modifications on search equations have been proposed to improve performance. However, efficiency of an algorithm usually changes depending on the type of problems and problem size. Consequently, there is not a variant which tackles all types problems effectively. Therefore, an adaptive search equations determination mechanism is needed.

To overcome this challenge, a self-adaptive search equation determination strategies for employed and onlooker bees steps are proposed in this paper. In this strategy, instead of suggesting a new or modified search equation, a pool of randomly generated search equations is used. According to problem type and problem size, the appropriate equation from the pool is tuned adaptively during the execution. On the other hand, incrementing population size strategy which enhances the search ability of the population is also proposed in SSEABC. With the strategy a new candidate solution is added to the population until a maximum population size is achieved. The resulted algorithm having these two strategies is called “Self-adaptive Search Equation-based Artificial Bee Colony” (SSEABC).

The performance of the SSEABC has been investigated for large-scale global optimization. The test suite of Soft Computing Journal Special Session on Large Scale Global Optimization (SOCO) [14] has been used as a benchmark set. According to the experimental results, SSEABC shows better performance than several ABC variants and performs similar to the SOCO competitor algorithms.

The rest of the article is organized as follows. In Sect. 2 briefly describes ABC. Then, the proposed algorithm is given in Sect. 3. Experiments and results are provided in Sect. 4. Finally, Sect. 5 concludes the article.

2 Artificial Bee Colony Algorithm

ABC algorithm contains initialization step and the following three steps that repeat until the termination condition is reached. These three steps are employed bees, onlooker bees and scout bees steps. In the initialization phase of algorithm, the food sources which represent candidate solutions are randomly located in D-dimensional search space. The employed bees visit these food sources and they try to investigate better ones biased towards the visited solutions.

After that, onlooker bees choose food sources with a selection probability. Then, they try to find new finer food sources like as the employed bee do. If a food source can not improved for several times, it is abandoned by an employed bee. This bee turns into a scout bee which discovers a new food source in the search space. The following is a more detailed description of the four steps of the canonical ABC algorithm:

Initialization. At this stage, SN numbers of food sources are created randomly in search space according to the following equation

$$x_{i,j} = x_j^{min} + \varphi_{i,j}(x_j^{max} - x_j^{min}) \tag{1}$$

where $\varphi_{i,j}$ is a uniform random number in $[0, 1]$ for decision variable j and food source i. Also, *limit* parameter that is set equal for all food sources is assigned for each food sources. This parameter controls the maximum limit of unsuccessful visits.

Employed Bees Step. In ABC, each employed bee is responsible for a food source. Therefore, the number of the employed bees is equal to the number of food sources. Each employed bee visits its food source and tries to discover new better food source by modifying the only one dimension as follows:

$$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{r1,j}), \tag{2}$$

where j is a randomly selected dimension ($j \in \{1, 2, \ldots, D\}$), $\phi_{i,j}$ is a uniform random number in $[-1, 1]$, $x_{i,j}$ and $x_{r1,j}$ represent the visited food source i and a randomly selected food source $r1$ at dimension j, respectively. When the visited food source i is worse than the candidate food source, greedy selection is applied on the visited food source.

Onlooker Bees Step. As in the employed bees step, the onlooker bees try to find better food sources by using Eq. 2. The only differences is that a stochastic selection of a food source is applied for onlooker bees. The selection probability of a food source is based on a knowledge that contains information about the quality of the source shared by an employed bee. It is calculated for a food source i is by:

$$p_i = \frac{\mathit{fitness}_i}{\sum_{n=1}^{SN} \mathit{fitness}_n}. \tag{3}$$

In Eq. 3, $\mathit{fitness}_i$ is the *fitness* value of the food source i and it is defined as

$$fitness_i = \begin{cases} \frac{1}{1+f_i}, & f_i \geqslant 0, \\ 1 + abs(f_i), & f_i < 0, \end{cases} \tag{4}$$

where f_i is the objective function value of food source i. The food sources with high probability have a better chance of being selected by the onlooker bees. This enforces intensification behavior of the algorithm.

Scout Bees Step. A food source can be visited by the colony members several times, however, if this food source location can not be improved, then it is seen as exhausted. The responsible employed bee becomes a scout bee which abandons the food source and replaced it with a randomly generated randomly using the Eq. 1. Owing to this, the algorithm tries to escape from local optima.

3 Self-adaptive Search Equation Based Artificial Bee Colony Algorithm

SSEABC is a new self-adaptive ABC variant in which self-adaptive search equation determination strategy tries find appropriate search equations for a given problem instance and the population size increases according to a particle addition schedule. In the following subsection, we describe the algorithm used in our study by focusing on these proposed modifications. The pseudo-code of the proposed algorithm is also presented in Algorithm 1.

Algorithm 1. Pseudo-code of the self adaptive ABC algorithm

```
Initialization of $SN$ numbers of food sources and other parameters such as $limit$ and $ps$
$counter = 0$
Fill the pool with randomly generated search equations
while termination condition is not met do
    for each employed bee $x_i$ do                                    ▷ Employed Bees Step
        Select equation from the pool indexed in ($counter \mod ps$)
        Apply search equation of employed bees step
        if $v_i$ is better than $x_i$ then
            $x_i = v_i$, $trial_i = 0$
            increase success counter for the search equation at $counter \mod ps$ index of the pool
        else
            $trial_i = trial_i + 1$
        end if
    end for
    Compute selection probabilities of solutions
    for each onlooker bee $x_i$ do                                    ▷ Onlooker Bees Step
        Select equation from the pool indexed in ($counter \mod ps$)
        Apply search equation of employed bees step
        if $v_i$ is better than $x_i$ then
            $x_i = v_i$, $trial_i = 0$
            increase success counter for the search equation at ($counter \mod ps$) index of the
                pool
        else
            $trial_i = trial_i + 1$
        end if
    end for
    Determine the abandoned food source, if exists,                   ▷ Scout Bees Step
    replace it with a randomly generated food source using Eq. 1
    $counter = counter + 1$
    if $SN < SNmax$ and ($counter \mod g == 0$) then    ▷ Incremental population size strategy
        add new solution to the current population by using Eq. 7
    end if
    if $counter$ is equal to $ps$ then                 ▷ Eliminating worst search equations
        sort the candidate search equations in the pool according to their success counter values
        eliminate worst candidates in pool and decrease the $ps$ with Eq. 5
        add a new randomly generated search equation to the pool
    end if
end while
Return the best food source as the solution
```

3.1 The Self-adaptive Search Equation Determination Strategy

Aydın [2] showed that search equations in employed bees and onlooker bees steps are the most curial components of the ABC algorithms. When they are tuned carefully related to the problem instance, then the performance of the algorithm can be improved significantly. To do so, while tacking problem instance, each terms and the number of dimensions to be changed in the search equation should be effectively determined [21]. Therefore, in this paper, a self-adaptive search equation selection strategy is proposed. With this strategy, a component-based generalized search equation is defined and suitable values of each elements of the generalized search equation is determined self-adaptively while the algorithm is running.

The generalized search equation defined in Algorithm 2 consists of four terms and a parameter, m, which is the number of dimensions to be changed. These four terms and m parameter have alternative settings which are listed in Table 1. SSEABC has a search equation pool in which search equations are generated based on the generalized search equation form. Each search equation is generated by selection one of the alternative for each component independently and randomly.

Algorithm 2. The general form of the search equation

```
1: for $t = 1$ to $m$ do
2:     select random dimension $j$ $(1 \leq j \leq D)$
3:     $x_{i,j} = term_1 + term_2 + term_3 + term_4$
4: end for
```

In initialization step of SSEABC, several randomly generated search equations fill the search equation pool. Then, a search equation is selected from the pool successively and used in the employed bees and onlooker bees steps at each iteration. To measure the effectiveness of the selected search equation, the number of successful solution updates are calculated. SSEABC reduces the pool size, ps, according to the success rates of the search equations after all search equations are used in employed bees and onlooker bees steps. The number of remaining successful search equations (also the updated new pool size, ps) is calculated using the following equation

$$ps = \frac{ps^2}{itr_{MAX}} \tag{5}$$

where itr_{MAX} is the approximated value of the maximum number of iterations which is calculated by

$$itr_{MAX} = \frac{MAXFES}{2 \times SN} \tag{6}$$

where $MAXFES$ is the maximum number of function evaluations for one execution and $2 \times SN$ is the number of function evaluations at each iteration. itr_{MAX} is an approximated value due to the incremental population size strategy which changes SN over time.

Table 1. The possible alternatives for each component of the proposed search strategy. ϕ_1, ϕ_2 and ϕ_3 can take two possible ranges: $[-1,-1]$ and $[-SF, SF]$ where SF $(0 \leq SF \leq 4)$ is a randomly selected positive real value. These ranges are decided randomly while creating each component of each randomly generated search equation. X_G and X_{GD} are the global-best food source and global-best distance food sources, respectively. More information can be found in [5].

m	$term1$	$term2$	$term3$	$term4$
1	$x_{i,j}$	$\phi 1(x_{i,j} - x_{G,j})$	$\phi 2(x_{i,j} - x_{G,j})$	$\phi 3(x_{i,j} - x_{G,j})$
k $(1 \leq k \leq D)$	$x_{G,j}$	$\phi 1(x_{i,j} - x_{r1,j})$	$\phi 2(x_{i,j} - x_{r1,j})$	$\phi 3(x_{i,j} - x_{r1,j})$
$rand(t, k)$ $(1 \leq t < k \leq D)$	$x_{r1,j}$	$\phi 1(x_{G,j} - x_{r1,j})$	$\phi 2(x_{G,j} - x_{r1,j})$	$\phi 3(x_{G,j} - x_{r1,j})$
		$\phi 1(x_{r1,j} - x_{r2,j})$	$\phi 2(x_{r1,j} - x_{r2,j})$	$\phi 3(x_{r1,j} - x_{r2,j})$
		$\phi 1(x_{i,j} - x_{GD,j})$	$\phi 2(x_{i,j} - x_{GD,j})$	$\phi 3(x_{i,j} - x_{GD,j})$
		do not use	do not use	do not use

3.2 The Incremental Population Size Strategy

It has been proven that incremental population size strategy increases algorithm performance [2,22,28]. In the incremental population size strategy, the population size is increased with new colony members, which are biased by the best members of the colony, every g iterations until the population size reachs the predefined maximum population size [3–5,22,28].

SSEABC starts with a small population size. Then, at each g iteration, a new food source (solution) which is localized using the position of the global-best solution (food source) (X_G) is added to the population. The initialization rule of a new solution, $\acute{X}_{new}$, for each dimension j is the following:

$$\acute{x}_{new,j} = x_{new,j} + \varphi_{i,j}(x_{G,j} - x_{new,j}) \tag{7}$$

where X_{new} is a solution which is generated randomly by using Eq. 1, and $\acute{X}_{new}$ is the final initial position of the new food source. This strategy affects the search behavior of the algorithm. A small value of g encourages exploration while large values help exploitation [22,28].

4 Experiments and Results

4.1 Experimental Setup

To benchmarking SSEABC algorithm on large scale optimization, the test suite for Soft Computing Special Issue on "Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems" (SOCO) was used [23]. SOCO is composed of 19 problems. Seven of them are unimodal and twelve of them are multimodal. At the same time, four functions are separable, rest of them are nonseparable. Table 2 briefly presents the SOCO benchmark functions with their peculiarities. Detailed information about SOCO can be found their web site [14].

Table 2. Description of the SOCO benchmark function suite.

Fun.	Name	Range	Separable	Uni/Multimodal
f1	Shifted Sphere Function	[−100, 100]	Y	U
f2	Shifted Schwefel's Problem 2.21	[−100, 100]	N	U
f3	Shifted Rosenbrock's Function	[−100, 100]	N	M
f4	Shifted Rastrigin's Function	[−5, 5]	Y	M
f5	Shifted Griewank's Function	[−600, 600]	N	M
f6	Shifted Ackley's Function	[−32, 32]	Y	M
f7	Schwefel's Problem 2.22	[−10, 10]	Y	U
f8	Schwefel's Problem 1.2	[−65.536, 65.536]	N	U
f9	Extended f10	[−100, 100]	N	U
f10	Bohachevsky	[−15, 15]	N	U
f11	Schaffer	[−100, 100]	N	U
f12	Hybrid f9 & f1 (25%, 75%)	[−100, 100]	N	M
f13	Hybrid f9 & f3 (25%, 75%)	[−100, 100]	N	M
f14	Hybrid f9 & f4 (25%, 75%)	[−5, 5]	N	M
f15	Hybrid f10 & f7 (25%, 75%)	[−10, 10]	N	M
f16	Hybrid f9 & f1 (50%, 50%)	[−100, 100]	N	M
f17	Hybrid f9 & f3 (75%, 25%)	[−100, 100]	N	M
f18	Hybrid f9 & f4 (75%, 25%)	[−5, 5]	N	M
f19	Hybrid f10 & f7 (75%, 25%)	[−10, 10]	N	M

In the experiments, SSEABC and all compared ABC algorithms run independently 25 times for each functions. Stopping criterion of all experiments is $D \times 5000$ as stated in SOCO. Error value is defined as $f(x) - f(x^*)$. In here, x means the solution of the algorithm and x^* indicates the optimum value. The test results smaller than 10^{-14} have been assumed as 10^{-14}. All ABC algorithms have been run with the default parameters mentioned in their original papers. The default parameter values of SSEABC and ABC algorithms are listed in the Table 3. Moreover, in order to have statistically sound conclusions, the two-sided Wilcoxon rank sum test [32] at a 0.05 significance level is used to appraise the significance of the performance between SSEABC and each compared algorithms. All experiments were carried out on a computer which has 2 GB of RAM and Intel Xeon E5-2620 2 GHz processor.

4.2 Comparison of the SSEABC with the ABC Variants

In this study, we compared the performance of the SSEABC on the 1000-dimensional SOCO functions with five ABC variants. These ABC algorithms are the canonical Artificial Bee Colony (ABC) [17], Modified Artificial Bee Colony (MABC) [1], Improved Artificial Bee Colony (ImpABC) [10], Best-so-far

Table 3. Parameters which are used in experiments

Parameters	SSEABC	ABC	BABC	MABC	ImpABC	OPIABC	description
SN	10	62	100	10	25	62	The number of initial population size
lf	1.0	1.0	0.1	1.0	1.0	1.0	The limit factor formalized as $lf = limit/(SN{*}D)$
ps	1000	–	–	–	–	–	The size of search equations pool
g	10	–	–	–	–	–	The period of growth of population size
SN_{max}	40	–	–	–	–	–	The maximum number of population size
w_{min}	–	–	0.2	–	–	–	A parameter used in scout bee step of BABC
w_{max}	–	–	1.0	–	–	–	A parameter used in scout bee step of BABC
MR	–	–	–	0.4	1.0	–	Modificiation ratio which is used in employed bee and onlooker bee steps of MABC
SF	–	–	–	1.0	–	–	Scaling factor which is used in employed bee and onlooker bee steps of MABC
p	–	–	–	–	0.25	–	The selection probability ratio of a search equation in ImpABC

selection Artificial Bee Colony (BABC) [6] and One-Point Inheritance Artificial Bee Colony (OPIABC) [35].

The comparison results on 1000-dimensional SOCO functions over median results are presented in Table 4. As described in Table 4, SSEABC beats ABC, BABC and ImpABC on 17 functions, MABC for 19 functions and finally OPIABC for 9 functions. As a result, SSEABC is ranked first and significantly outperforms all ABC variants except OPIABC.

Table 4. SSEABC median values comparisons with ABC variants on SOCO. "Win", "Loss" and "Draw" part of the table shows the number of times SSEABC is better, equal or worse, respectively, than the algorithm

Functions	SSEABC	ABC	BABC	MABC	ImpABC	OPIABC
f1	**1.000E-14**	2.111E-06	2.411E+04	1.477E+00	1.545E+04	**1.000E-14**
f2	8.576E+01	**1.565E+02**	4.965E+01	1.411E+02	4.986E+01	1.417E+02
f3	1.026E+03	7.671E+02	1.667E+09	9.041E+05	1.730E+09	**2.414E+00**
f4	**1.000E-14**	1.481E+02	2.933E+03	4.121E+03	8.460E+03	**1.000E-14**
f5	**1.000E-14**	6.179E-07	2.121E+02	3.020E-01	1.197E+02	**1.000E-14**
f6	**1.000E-14**	2.006E-02	1.450E+01	1.982E+01	1.944E+01	9.499E-13
f7	**1.000E-14**	9.965E-04	1.375E+01	2.018E-02	4.253E+02	4.519E-14
f8	1.071E+06	2.514E+06	8.178E+05	6.756E+06	**4.785E+05**	2.297E+06
f9	**1.000E-14**	3.228E+02	4.351E+03	7.223E+03	6.388E+03	2.856E-02
f10	**1.000E-14**	7.370E-06	4.840E+02	6.399E+01	1.130E+03	**1.000E-14**
f11	**4.334E-07**	3.240E+02	4.195E+03	7.097E+03	6.489E+03	2.938E-02
f12	**1.308E-11**	4.967E+01	1.705E+04	2.146E+03	1.319E+04	3.769E-05
f13	8.083E+02	6.742E+02	1.159E+09	6.139E+03	5.692E+08	**6.150E+00**
f14	9.950E-01	1.263E+02	2.517E+03	3.262E+03	6.422E+03	**3.993E-06**
f15	**1.000E-14**	7.010E-04	4.725E+01	1.267E+01	1.137E+03	**1.000E-14**
f16	**7.462E-08**	1.170E+02	9.658E+03	4.174E+03	7.461E+03	2.826E-04
f17	2.135E+02	3.152E+02	8.625E+07	7.413E+03	7.105E+07	**2.219E-01**
f18	**8.937E-08**	8.753E+01	1.242E+03	1.944E+03	2.657E+03	1.516E-04
f19	**1.000E-14**	2.169E-04	9.801E+01	4.418E+01	8.553E+02	**1.000E-14**
rank	1	3	5	4	6	2
win		17	17	19	17	9
loss		2	2	0	2	4
draw		0	0	0	0	6
p-value		0.0088	0.00222	0.00014	0.00194	0.75656

4.3 Comparison of SSEABC with SOCO Competitors

SSEABC was also compared with 15 participant algorithms [7–9, 11–13, 19, 25, 27, 29, 31, 34, 36] in SOCO special issue. The median results of these algorithms is taken directly from their original papers. The comparison is given in Table 5. As seen in Table 5, SSEABC is significantly better than CHC, EvoPROpt and MASSW. Only MOS algorithm which is the winner of the competition and JDElscop outperform SSEABC algorithm significantly. Therefore, in the light of the comparison results, it can be clearly claimed that SSEABC is giving very competitive results on SOCO benchmark functions.

Table 5. SSEABC median values comparisons with 15 state-of-art algorithms on SOCO. “Win”, “Loss” and “Draw” part of the table shows the number of times SSEABC is better, equal or worse, respectively, than the algorithm

Functions	**SSEABC**	**CHC [9]**	**DE [29]**	**DEDM [11]**	**EM323 [13]**	**EvoPROpt [8]**	**GaDE [34]**	**GODE [30]**	**jDElscop [7]**	**MASSW [25]**	**MOS [19]**	**RPSOmv [12]**	**SaDEMMTS [36]**	**SOUPDE [31]**	**IPSOLS [26]**	**VXQR [27]**
f1	**1.000E-14**	1.296E-11	**1.000E-14**	**1.000E-14**	1.114E-11	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	2.820E-14	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**
f2	8.576E+01	1.442E+02	8.444E+01	7.056E+01	2.294E+01	3.163E+01	8.920E+01	8.910E+01	6.210E+01	1.391E+02	2.917E-01	4.290E+01	4.880E+01	9.250E+01	**6.679E-14**	9.640E+01
f3	1.026E+03	1.493E+03	9.685E+02	9.465E+02	4.906E+02	1.087E+03	9.450E+01	9.700E+02	8.440E+02	1.212E+03	5.790E+01	1.370E+02	1.000E-14	9.620E+02	**1.000E-14**	6.050E+02
f4	**1.000E-14**	4.731E+03	1.315E+00	9.950E-01	1.359E-11	1.214E+03	**1.000E-14**	1.080E+00	**1.000E-14**	1.611E+03	**1.000E-14**	2.780E-14	3.210E+01	2.000E-11	1.000E-14	5.684E-14
f5	**1.000E-14**	1.705E-13	**1.000E-14**	**1.000E-14**	6.651E-12	1.972E-02	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	1.120E-01	**1.000E-14**	**1.000E-14**	**1.000E-14**	1.086E-11
f6	**1.000E-14**	1.399E+01	3.297E-12	1.535E-12	1.299E-11	2.447E+00	1.420E-14	2.880E-13	2.610E-12	1.451E-09	**1.000E-14**	3.750E-12	**1.000E-14**	3.420E-13	1.000E-14	1.668E-11
f7	**1.000E-14**	1.876E-03	**1.000E-14**	**1.000E-14**	1.000E+200	**1.000E-14**	**1.000E-14**	1.000E+200	**1.000E-14**	6.173E-13	**1.000E-14**	**1.000E-14**	**1.000E-14**	3.570E-13	7.134E-11	**1.000E-14**
f8	1.071E+06	3.134E+05	2.459E+05	9.713E+10	1.541E+06	2.057E+05	1.730E+04	1.891E+05	3.150E+04	7.384E+04	1.896E+05	9.210E+05	1.460E+03	2.120E+05	1.197E+05	**1.000E-14**
f9	**1.000E-14**	6.109E+03	5.130E+03	**1.000E-14**	2.546E-08	3.656E+02	**1.000E-14**	1.707E-04	5.970E-08	5.988E+03	**1.000E-14**	9.840E+00	9.250E+01	7.390E-05	9.760E+00	9.994E+01
f10	**1.000E-14**	3.166E+02	**1.000E-14**	**1.000E-14**	**1.000E-14**	4.245E+02	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**	**1.000E-14**
f11	4.334E-07	4.832E+03	1.351E-03	**1.000E-14**	3.256E-07	3.503E+02	**1.000E-14**	1.718E-04	1.200E-07	5.270E+01	**1.000E-14**	9.610E+00	1.850E+02	7.440E-05	9.750E+00	1.116E+02
f12	1.308E-11	1.041E+03	1.698E-08	4.601E-12	2.206E-06	3.056E+01	3.580E-12	1.935E-09	**1.000E-14**	9.127E-02	**1.000E-14**	2.020E-12	**1.000E-14**	**1.000E-14**	1.374E+00	2.243E+02
f13	8.083E+02	1.803E+03	7.294E+02	7.167E+02	1.188E+03	9.990E+02	7.090E+02	7.314E+02	6.410E+02	9.974E+02	7.245E+01	2.270E+02	6.550E+02	7.260E+02	**1.280E+00**	5.388E+02
f14	9.950E-01	3.648E+03	9.950E-01	8.794E-10	2.182E-06	5.474E+02	8.240E-11	9.950E-01	**1.000E-14**	7.895E+02	**1.000E-14**	4.070E-08	1.400E+02	6.370E-12	1.778E+01	5.638E+00
f15	**1.000E-14**	7.761E+01	**1.000E-14**	**1.000E-14**	1.000E+200	8.119E+01	**1.000E-14**	1.000E+200	**1.000E-14**	1.154E-13	**1.000E-14**	**1.000E-14**	**1.000E-14**	2.690E-13	1.391E-10	2.695E-02
f16	7.462E-08	2.299E+03	4.193E-08	2.259E-11	5.102E-06	1.001E+02	2.320E-12	4.643E-09	**1.000E-14**	7.378E-01	**1.000E-14**	2.190E-06	**1.000E-14**	**1.000E-14**	2.408E+00	6.346E+01
f17	2.135E+02	3.813E+03	2.354E+02	2.240E+02	3.185E+02	7.859E+02	2.180E+02	2.358E+02	1.670E+02	3.157E+02	3.258E+00	4.020E+01	1.900E+02	2.310E+02	6.494E+00	2.974E+02
f18	8.937E-08	1.710E+03	2.367E-03	3.213E-07	5.719E-06	1.754E+02	1.180E-07	1.136E-05	2.840E-12	1.056E+01	**1.000E-14**	1.330E+00	8.200E+01	1.360E-12	2.462E+01	1.572E+01
f19	**1.000E-14**	3.446E+03	**1.000E-14**	**1.000E-14**	**1.000E-14**	1.736E+02	**1.000E-14**	**1.000E-14**	**1.000E-14**	4.526E-13	**1.000E-14**	**1.000E-14**	**1.000E-14**	9.500E-14	1.009E-10	1.373E-03
rank	6	16	10	5	13	15	3	11	2	14	1	9	4	7	8	12
win		18	8	5	13	15	4	10	2	15	0	8	5	9	9	13
loss		1	5	7	4	2	7	5	10	1	10	7	7	7	5	3
draw		0	6	7	2	2	8	4	7	3	9	4	7	3	5	3
p-value		0.00222	0.87288	0.58232	0.09296	0.00854	0.21498	0.29834	0.00596	0.00714	0.00512	0.36282	0.75656	0.6818	0.63836	0.23404

5 Conclusion

In this work, SSEABC algorithm have been proposed for large scale optimization problems. It dynamically determines the appropriate search equations and uses incremental population size strategy in order to performance improvement. The proposed algorithm was compared with the five ABC variants and 15 SOCO special issue on large scale optimization competitors. SSEABC can lead to superior performance over almost all considered ABC variants. In addition to this, MOS and JDElscop are the best algorithm for SOCO functions. Noteworthy is that SSEABC generally performed better than several competitor algorithms, which is widely acknowledged to be a state-of-the-art algorithm for large scale optimization problems. In this sense, the observed results with SSEABC are very promising.

In the future work, we plan to test our algorithm on other benchmark suites (such as CEC 2014 and CEC 2015 benchmark suites) and real-world optimization problems. Furthermore, a possible direction of a further work can be examined to improve algorithm performance. For instance, we may hybridize SSEABC with local search procedures; an alternative is to improve self-adaptive search equation selection strategy with considering more recently proposed components and adding new determination rules.

References

1. Akay, B., Karaboga, D.: A modified artificial bee colony algorithm for real-parameter optimization. Inf. Sci. **192**, 120–142 (2012)
2. Aydın, D.: Composite artificial bee colony algorithms: from component-based analysis to high-performing algorithms. Appl. Soft Comput. **32**, 266–285 (2015)
3. Aydın, D., Liao, T., Montes de Oca, M.A., Stützle, T.: Improving performance via population growth and local search: the case of the artificial bee colony algorithm. In: Hao, J.-K., Legrand, P., Collet, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) EA 2011. LNCS, vol. 7401, pp. 85–96. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35533-2_8
4. Aydın, D., Özyön, S.: Solution to non-convex economic dispatch problem with valve point effects by incremental artificial bee colony with local search. Appl. Soft Comput. **13**(5), 2456–2466 (2013)
5. Aydin, D., Stuetzle, T.: A configurable generalized artificial bee colony algorithm with local search strategies. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1067–1074. IEEE, May 2015
6. Banharnsakun, A., Achalakul, T., Sirinaovakul, B.: The best-so-far selection in artificial bee colony algorithm. Appl. Soft Comput. **11**(2), 2888–2901 (2011)
7. Brest, J., Maučec, M.S.: Self-adaptive differential evolution algorithm using population size reduction and three strategies. Soft. Comput. **15**(11), 2157–2174 (2010)
8. Duarte, A., Martí, R., Gortazar, F.: Path relinking for large-scale global optimization. Soft Comput. **15**(11), 2257–2273 (2011)
9. Eshelman, L.: The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In: Foundations of Genetic Algorithms, pp. 265–283 (1991)

10. Gao, W., Liu, S.: Improved artificial bee colony algorithm for global optimization. Inf. Process. Lett. **111**(17), 871–882 (2011)
11. García-Martínez, C., Rodríguez, F.J., Lozano, M.: Role differentiation and malleable mating for differential evolution: an analysis on large-scale optimisation. Soft Comput. **15**(11), 2109–2126 (2011)
12. García-Nieto, J., Alba, E.: Restart particle swarm optimization with velocity modulation: a scalability test. Soft Comput. **15**(11), 2221–2232 (2011)
13. Gardeux, V., Chelouah, R., Siarry, P., Glover, F.: EM323: a line search based algorithm for solving high-dimensional continuous non-linear optimization problems. Soft Comput. **15**(11), 2275–2285 (2011)
14. Herrera, F., Lozano, M., Molina, D.: Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems. http://sci2s.ugr.es/sites/default/files/files/TematicWebSites/EAMHCO/testfunctions-SOCO.pdf
15. Hsieh, S.T., Sun, T.Y., Liu, C.C., Tsai, S.T.: Solving large scale global optimization using improved particle swarm optimizer. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), vol. 1, pp. 1777–1784 (2008)
16. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
17. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**(3), 459–471 (2007)
18. Kazimipour, B., Xiaodong, L., Qin, A.K.: Effects of population initialization on differential evolution for large scale optimization. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2404–2411 (2014)
19. LaTorre, A., Muelas, S., Peña, J.M.: A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. Soft. Comput. **15**(11), 2187–2199 (2011)
20. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans. Evol. Comput. **16**(2), 210–224 (2012)
21. Liao, T., Aydın, D., Stützle, T.: Artificial bee colonies for continuous optimization: experimental analysis and improvements. Swarm Intell. **7**(4), 327–356 (2013)
22. Liao, T., Montes de Oca, M.A., Aydin, D., Stützle, T., Dorigo, M.: An incremental ant colony algorithm with local search for continuous optimization. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 125–132. ACM, New York (2011)
23. Lozano, M., Molina, D., Herrera, F.: Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. Soft. Comput. **15**(11), 2085–2087 (2011)
24. Mahdavi, S., Shiri, M.E., Rahnamayan, S.: Metaheuristics in large-scale global continues optimization: a survey. Inf. Sci. **295**, 407–428 (2015)
25. Molina, D., Lozano, M., Sánchez, A.M., Herrera, F.: Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SSW-Chains. Soft Comput. **15**(11), 2201–2220 (2011)
26. Montes de Oca, M.A., Aydin, D., Stützle, T.: An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re)design of optimization algorithms. Soft Comput. **15**(11), 2233–2255 (2011)

27. Neumaier, A., Fendl, H., Schilly, H., Leitner, T.: VXQR: Derivative-free unconstrained optimization based on QR factorizations. Soft. Comput. **15**(11), 2287–2298 (2011)
28. de Oca, M.A.M., Stutzle, T., Van den Enden, K., Dorigo, M.: Incremental social learning in particle swarms. IEEE Trans. Syst. Man, Cybern. Part B Cybern. **41**(2), 368–384 (2011)
29. Storn, R., Price, K.: Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
30. 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)
31. Weber, M., Neri, F., Tirronen, V.: Shuffle or update parallel differential evolution for large-scale optimization. Soft. Comput. **15**(11), 2089–2107 (2011)
32. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bull. **1**, 80–83 (1945)
33. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Inf. Sci. **178**(15), 2985–2999 (2008)
34. Yang, Z., Tang, K., Yao, X.: Scalability of generalized adaptive differential evolution for large-scale continuous optimization. Soft. Comput. **15**(11), 2141–2155 (2011)
35. Zhang, X., Yuen, S.Y.: Improving artificial bee colony with one-position inheritance mechanism. Memetic Comput. **5**(3), 187–211 (2013)
36. Zhao, S.Z., Suganthan, P.N., Das, S.: Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. Soft. Comput. **15**(11), 2175–2185 (2011). http://link.springer.com/10.1007/s00500-010-0645-4

Neural Networks, Self-organization, Machine Learning

Lie Algebra-Valued Bidirectional Associative Memories

Călin-Adrian Popa(✉)

Department of Computer and Software Engineering,
Polytechnic University Timişoara,
Blvd. V. Pârvan, No. 2, 300223 Timişoara, Romania
calin.popa@cs.upt.ro

Abstract. In recent years, complex-, quaternion-, and Clifford-valued neural networks have been intensively studied. This paper introduces Lie algebra-valued bidirectional associative memories, an alternative generalization of the real-valued neural networks, for which the states, outputs, and thresholds are all from a Lie algebra. The definition of these networks is given, together with an expression for an energy function, that is indeed proven to be an energy function for the proposed network.

Keywords: Clifford-valued neural networks · Bidirectional associative memories · Energy function · Lie algebra-valued neural networks

1 Introduction

Recently, there has been an increasing interest in the study of neural networks with values in multidimensional domains. Complex-valued neural networks, which were first introduced in the 1970s (see, for example, [23]), but have received more attention in the 1990s and in the past decade, are the first type of multidimensional neural networks that was proposed. They have a wide range of applications, starting from those in complex-valued signal processing and continuing with applications in telecommunications and image processing (see, for example, [4,11]).

Quaternion-valued neural networks were introduced in the 1990s also, first as a generalization of the complex-valued neural networks, see [1,2,14]. They are defined on the 4-dimensional algebra of quaternion numbers. Chaotic time series prediction, the 4-bit parity problem, and quaternion-valued signal processing are just a few areas of application for these networks. A promising area which remains to be explored is that of 3-dimensional and color image processing, in which data can be more naturally expressed in quaternion form.

Clifford-valued neural networks were defined in [18,19], and later discussed, for example, in [3,8,9]. Clifford algebras or geometric algebras have many applications in physics and engineering, making them appealing as novel types of data representation for the neural network domain, also. The complex and quaternion algebras are special types of Clifford algebras, which have dimension 2^n, $n \geq 1$.

Because of the strong connection between Clifford algebra and geometry, neural networks defined on this algebra will be able process different geometric objects and apply different geometric models to data, which may give them power to solve many problems arising in the design of intelligent systems.

Vector-valued neural networks represent a different approach besides the Clifford-valued neural networks, see [12,13,15]. Two variants of these networks exist: one based on the vector product, which has three dimensional vectors as weights, and one which has orthogonal matrices as weights (i.e. matrices that satisfy $AA^T = A^T A = I$). This last variant was further generalized to N-dimensional vectors, and thus the N-dimensional neural networks (see [16,17]), have N-dimensional vector inputs and outputs, but orthogonal matrix weights. Three-dimensional neural networks were used to learn geometric transformations and a single N-dimensional neuron was used for solving the N-bit parity problem.

We proposed a different generalization of real-valued neural networks in multidimensional domains, namely neural networks that have Lie algebraic inputs, outputs, weights and biases, in the form of Lie algebra-valued feedforward neural networks, see [20]. Because Lie algebras can have any dimension n, they also represent an alternative to the N-dimensional neural networks that we mentioned earlier. The definition of Lie algebras also comes from geometry, and they have numerous applications in physics and engineering (see [5,6,21]), and also in computer vision (for a survey, see [24], and the references thereof). Thus, we considered a promising idea to define neural networks with values in Lie algebras, also.

Since they were first introduced by Kosko in [7], bidirectional associative memories, an extension of the unidirectional auto-associative Hopfield neural networks, were intensely studied, and have many applications in pattern recognition and automatic control. Because of the fact that complex-valued bidirectional associative memories were introduced in [10], quaternion-valued bidirectional associative memories in [8], and Clifford-valued bidirectional associative memories in [22], we considered an interesting idea to introduce Lie algebra-valued bidirectional associative memories. These networks can be applied to store Lie-algebraic patterns and to solve difficult optimization problems defined on a Lie group or a Lie algebra.

The remainder of this paper is organized as follows: Sect. 2 gives the definition of Lie algebra-valued bidirectional associative memories, and the expression for the energy function, together with the proof that the given function is indeed an energy function for the defined network. Section 3 is dedicated to presenting the conclusions of the study.

2 Lie Algebra-Valued Bidirectional Associative Memories

A Lie algebra is a vector space $\mathfrak{g}$ over a field F together with an operation $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ called the Lie bracket, which satisfies the following axioms:

– It is bilinear: $[ax + by, z] = a[x, z] + b[y, z]$, $[x, ay + bz] = a[x, y] + b[x, z]$, $\forall a, b \in F$, $\forall x, y \in \mathfrak{g}$.
– It is skew symmetric: $[x, x] = 0$, which implies $[x, y] = -[y, x]$, $\forall x, y \in \mathfrak{g}$.
– It satisfies the Jacobi identity: $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$, $\forall x, y, z \in \mathfrak{g}$.

Consider the vector space $\mathfrak{so}(n)$ of skew-symmetric square matrices of order n, i.e. square matrices with real elements for which $A^T = -A$, $\forall A \in \mathfrak{so}(n)$. $\mathfrak{so}(n)$ is a Lie algebra, with the Lie bracket given by

$$[A, B] := AB - BA,\ \forall A, B \in \mathfrak{so}(n).$$

For the easiness of the exposition, we will work only with this Lie algebra, but the generalization to an arbitrary Lie algebra can be done in a simple way.

In what follows, we will define bidirectional associative memories for which the states, outputs, and thresholds are all from $\mathfrak{so}(n)$, which means that they are skew-symmetric square matrices. The network is described by the set of differential equations

$$\begin{cases} \tau_i \dfrac{dX_i(t)}{dt} = -X_i(t) + \displaystyle\sum_{j=1}^{P} W_{ij}^T f(Y_j(t)) W_{ij} + A_i, \\ \qquad\qquad\qquad\qquad\qquad\qquad \forall i \in \{1, \ldots, N\}, \\ \upsilon_j \dfrac{dY_j(t)}{dt} = -Y_j(t) + \displaystyle\sum_{i=1}^{N} W_{ji}^T f(X_i(t)) W_{ji} + B_j, \\ \qquad\qquad\qquad\qquad\qquad\qquad \forall j \in \{1, \ldots, P\}, \end{cases} \tag{1}$$

where $\tau_i \in \mathbb{R}$, $\tau_i > 0$ is the time constant of neuron X_i, $\upsilon_j \in \mathbb{R}$, $\upsilon_j > 0$ is the time constant of neuron Y_j, $X_i(t) \in \mathfrak{so}(n)$ is the state of neuron X_i at time t, $Y_j(t) \in \mathfrak{so}(n)$ is the state of neuron Y_j at time t, $W_{ij} \in SO(n)$ is the weight connecting neuron X_i to neuron Y_j, $f : \mathfrak{so}(n) \to \mathfrak{so}(n)$ is the nonlinear Lie algebra-valued activation function, A_i is the threshold of neuron X_i, and B_j is the threshold of neuron Y_j, $\forall i \in \{1, \ldots, N\}$, $\forall j \in \{1, \ldots, P\}$. $SO(n)$ is the group of orthogonal square matrices, i.e. square matrices with real elements for which $A^T A = AA^T = I_n$ and $\det A = 1$, $\forall A \in SO(n)$, and represents the Lie group associated with the Lie algebra $\mathfrak{so}(n)$. It is known that $W^T V W \in \mathfrak{so}(n)$, $\forall W \in SO(n)$, $\forall V \in \mathfrak{so}(n)$, and so the above expressions are well defined. The derivative is considered to be the matrix formed by the derivatives of each element $[X_i(t)]_{ab}$ of the matrix $X_i(t)$ with respect to t:

$$\frac{dX_i(t)}{dt} := \left(\frac{d([X_i]_{ab})}{dt} \right)_{1 \le a, b \le n}.$$

If we denote by $U_i(t) := f(X_i(t))$ the output of neuron X_i and by $V_j(t) := f(Y_j(t))$ the output of neuron Y_j, the above set of differential equations can be written as:

$$\begin{cases} \tau_i \dfrac{dX_i(t)}{dt} = -X_i(t) + \displaystyle\sum_{j=1}^{P} W_{ij}^T V_j(t) W_{ij} + A_i, \\ \qquad\qquad\qquad\qquad\qquad\qquad \forall i \in \{1, \ldots, N\}, \\ \upsilon_j \dfrac{dY_j(t)}{dt} = -Y_j(t) + \displaystyle\sum_{i=1}^{N} W_{ji}^T U_i(t) W_{ji} + B_j, \\ \qquad\qquad\qquad\qquad\qquad\qquad \forall j \in \{1, \ldots, P\}. \end{cases}$$

The activation function is formed of n^2 functions $f^{ab} : \mathfrak{so}(n) \to \mathbb{R}$, $1 \leq a, b \leq n$:

$$f(X) = \left(f^{ab}(X)\right)_{1 \leq a,b \leq n}.$$

In order to study the stability of the above defined network, we need to make a series of assumptions about the activation function.

The first assumption is that the functions f^{ab} are continuously differentiable with respect to each $[X]_{cd}$, $\forall 1 \leq c, d \leq n$, $\forall 1 \leq a, b \leq n$, and the function f is bounded: $\exists M > 0$, $||f(X)|| \leq M$, $\forall X \in \mathfrak{so}(n)$, where $||Y||$ is the Frobenius norm of matrix Y, defined by $||Y|| = \sqrt{\mathrm{Tr}(YY^T)}$, and $\mathrm{Tr}(Y)$ represents the trace of matrix Y. Now, the $n^2 \times n^2$ Jacobian matrix of the function f can be defined as

$$\mathbf{Jac}_f(X) = \left(\frac{\partial f^{ab}(X)}{\partial [X]_{cd}}\right)_{\substack{1 \leq a,b \leq n \\ 1 \leq c,d \leq n}}.$$

The second assumption that we have to make is that f is injective and $\mathbf{Jac}_f(X)$ is symmetric and positive definite, $\forall X \in \mathfrak{so}(n)$. This, together with the above assumption, assures the existence of the inverse function of f, $g : \mathfrak{so}(n) \to \mathfrak{so}(n)$, $g = f^{-1}$. We can thus write $g(U_i(t)) = X_i(t)$, $\forall i \in \{1, \ldots, N\}$, and $g(V_j(t)) = Y_i(t)$, $\forall j \in \{1, \ldots, P\}$. Now, there exists a function $G : \mathfrak{so}(n) \to \mathbb{R}$,

$$G(X) = \sum_{a,b=1}^{n} \int_0^{[X]_{ab}} g^{ab}(Y^{ab}) dy,$$

where $g^{ab} : \mathfrak{so}(n) \to \mathbb{R}$ are the component functions of g and the matrices Y^{ab} have the following form

$$[Y^{ab}]_{cd} = \begin{cases} y, & (c,d) = (a,b) \\ [X]_{cd}, & \text{else} \end{cases}, \ \forall 1 \leq a, b \leq n.$$

For example, for 2×2 matrices, we have that

$$G(X) = \int_0^{[X]_{11}} g^{11}\left(\begin{pmatrix} y & [X]_{12} \\ [X]_{21} & [X]_{22} \end{pmatrix}\right) dy + \int_0^{[X]_{12}} g^{12}\left(\begin{pmatrix} [X]_{11} & y \\ [X]_{21} & [X]_{22} \end{pmatrix}\right) dy$$
$$+ \int_0^{[X]_{21}} g^{21}\left(\begin{pmatrix} [X]_{11} & [X]_{12} \\ y & [X]_{22} \end{pmatrix}\right) dy + \int_0^{[X]_{22}} g^{22}\left(\begin{pmatrix} [X]_{11} & [X]_{12} \\ [X]_{21} & y \end{pmatrix}\right) dy.$$

This function satisfies

$$\frac{\partial G(X)}{\partial [X]_{ab}} = g^{ab}(X),\ \forall 1 \leq a, b \leq n.$$

The above condition can also be written in matrix form as

$$\frac{\partial G(X)}{\partial X} = g(X). \tag{2}$$

The last assumption concerns the weights of the network, which must satisfy:

$$W_{ji} = W_{ij}^T,\ \forall i \in \{1, \ldots, N\},\ \forall j \in \{1, \ldots, P\}.$$

Having made all the above assumptions, we can define the energy function $E : \mathfrak{so}(n)^{N+P} \to \mathbb{R}$ of the bidirectional associative memory (1) as:

$$\begin{aligned} E(\mathbf{U}(t), \mathbf{V}(t)) = &- \sum_{i=1}^{N} \sum_{j=1}^{P} \mathrm{Tr}(U_i(t)^T W_{ij}^T V_j(t) W_{ij}) \\ &+ \sum_{i=1}^{N} G(U_i(t)) - \sum_{i=1}^{N} \mathrm{Tr}(A_i^T U_i(t)) \\ &+ \sum_{j=1}^{P} G(V_j(t)) - \sum_{j=1}^{P} \mathrm{Tr}(B_j^T V_j(t)). \end{aligned} \tag{3}$$

A function E is an energy function for the network (1) if the derivative of E along the trajectories of network, denoted by $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt}$, satisfies the condition $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} \leq 0$ and $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} = 0 \Leftrightarrow \frac{dU_i(t)}{dt} = \frac{dV_j(t)}{dt} = 0$, $\forall i \in \{1, \ldots, N\}$, $\forall j \in \{1, \ldots, P\}$. We will show that the function E defined in (3) is indeed an energy function for the network (1).

For this, we start by applying the chain rule:

$$\begin{aligned} \frac{dE(\mathbf{U}(t), \mathbf{V}(t))}{dt} &= \sum_{i=1}^{N} \sum_{a,b=1}^{n} \frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial [U_i(t)]_{ab}} \frac{d[U_i(t)]_{ab}}{dt} \\ &+ \sum_{j=1}^{P} \sum_{a,b=1}^{n} \frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial [V_j(t)]_{ab}} \frac{d[V_j(t)]_{ab}}{dt} \\ &= \sum_{i=1}^{N} \mathrm{Tr}\left(\left(\frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial U_i(t)}\right)^T \frac{dU_i(t)}{dt}\right) \\ &+ \sum_{j=1}^{P} \mathrm{Tr}\left(\left(\frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial V_j(t)}\right)^T \frac{dV_j(t)}{dt}\right), \end{aligned} \tag{4}$$

where by $\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [U_i(t)]_{ab}}$ we denoted the partial derivative of the function E with respect to each element $[U_i(t)]_{ab}$ of the matrices $U_i(t)$, $\forall 1 \leq a, b \leq n$, $\forall i \in \{1, \ldots, N\}$, and analogously for $\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [V_j(t)]_{ab}}$.

For the partial derivatives

$$\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial U_i(t)} = \left(\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [U_i(t)]_{ab}}\right)_{1\leq a,b\leq n}$$

and

$$\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial V_j(t)} = \left(\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [V_j(t)]_{ab}}\right)_{1\leq a,b\leq n},$$

we have from (3) that

$$\begin{aligned}\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial U_i(t)} &= -\sum_{j=1}^{P} W_{ij}^T V_j(t) W_{ij} + g(U_i(t)) - A_i\\ &= -\left(\sum_{j=1}^{P} W_{ij}^T V_j(t) W_{ij} - X_i(t) + A_i\right)\\ &= -\tau_i \frac{dX_i(t)}{dt},\ \forall i \in \{1,\ldots,N\},\end{aligned}$$

$$\begin{aligned}\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial V_j(t)} &= -\sum_{i=1}^{N} W_{ji}^T U_i(t) W_{ji} + g(V_j(t)) - B_j\\ &= -\left(\sum_{i=1}^{N} W_{ji}^T U_j(t) W_{ji} - Y_j(t) + B_j\right)\\ &= -\upsilon_j \frac{dY_j(t)}{dt},\ \forall j \in \{1,\ldots,P\},\end{aligned}$$

where we used the fact that

$$\frac{d\mathrm{Tr}(X^T A)}{dX} = A,$$

$$\frac{d\mathrm{Tr}(AXB)}{dX} = A^T B^T,$$

relation (2), the assumption $W_{ji} = W_{ij}^T$, and also the set of equations given by (1). Now, Eq. (4) becomes:

$$\begin{aligned}\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} &= \sum_{i=1}^{N}\mathrm{Tr}\left(\left(-\tau_i\frac{dX_i(t)}{dt}\right)^T\frac{dU_i(t)}{dt}\right)\\ &+\sum_{j=1}^{P}\mathrm{Tr}\left(\left(-\upsilon_j\frac{dY_j(t)}{dt}\right)^T\frac{dV_j(t)}{dt}\right)\\ &=-\sum_{i=1}^{N}\tau_i\left[\mathrm{vec}\left(\frac{dX_i(t)}{dt}\right)\right]^T\mathrm{vec}\left(\frac{dU_i(t)}{dt}\right)\\ &\quad-\sum_{j=1}^{P}\upsilon_j\left[\mathrm{vec}\left(\frac{dY_j(t)}{dt}\right)\right]^T\mathrm{vec}\left(\frac{dV_j(t)}{dt}\right)\\ &=-\sum_{i=1}^{N}\left\{\tau_i\left[\mathrm{vec}\left(\frac{dU_i(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(U_i(t))]^T\mathrm{vec}\left(\frac{dU_i(t)}{dt}\right)\right\}\\ &\quad-\sum_{j=1}^{P}\left\{\upsilon_j\left[\mathrm{vec}\left(\frac{dV_j(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(V_j(t))]^T\mathrm{vec}\left(\frac{dV_j(t)}{dt}\right)\right\}\\ &\le 0, \end{aligned}\tag{5}$$

where we denoted by vec (X) the vectorization of matrix X. We also used the identity

$$\mathrm{Tr}(A^TB)=\mathrm{vec}(A)^T\mathrm{vec}(B),\ \forall A,B\in\mathfrak{so}(n),$$

and, from $g(U_i(t))=X_i(t)$ and $g(V_j(t))=Y_j(t)$, we obtained that

$$\mathrm{vec}\left(\frac{dg(U_i(t))}{dt}\right)=\mathbf{Jac}_g(U_i(t))\mathrm{vec}\left(\frac{dU_i(t)}{dt}\right),$$

$$\mathrm{vec}\left(\frac{dg(V_j(t))}{dt}\right)=\mathbf{Jac}_g(V_j(t))\mathrm{vec}\left(\frac{dV_j(t)}{dt}\right),$$

$\forall i\in\{1,\ldots,N\}$, $\forall j\in\{1,\ldots,P\}$. Because $\mathbf{Jac}_f(X)$ is symmetric and positive definite, we deduce that $\mathbf{Jac}_g(U)$ is also symmetric and positive definite, and thus

$$\left[\mathrm{vec}\left(\frac{dU_i(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(U_i(t))]^T\mathrm{vec}\left(\frac{dU_i(t)}{dt}\right)\ge 0,$$

$$\left[\mathrm{vec}\left(\frac{dV_j(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(V_j(t))]^T\mathrm{vec}\left(\frac{dV_j(t)}{dt}\right)\ge 0,$$

$\forall i\in\{1,\ldots,N\}$, $\forall j\in\{1,\ldots,P\}$, which allowed us to write the last inequality in relation (5). Equality is attained when $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt}=0\Leftrightarrow \mathrm{vec}\left(\frac{dU_i(t)}{dt}\right)=\mathrm{vec}\left(\frac{dV_j(t)}{dt}\right)=0\Leftrightarrow\frac{dU_i(t)}{dt}=\frac{dV_j(t)}{dt}=0$, $\forall i\in\{1,\ldots,N\}$, $\forall j\in\{1,\ldots,P\}$, thus ending the proof that E is indeed an energy function for the network (1).

We now give two examples of activation functions, inspired by the ones used in real-valued and complex-valued neural networks:

$$f(V)=\frac{V}{1+||V||},\ \forall V\in\mathfrak{so}(n),$$

$$f\left(([V]_{ab})_{1\leq a,b\leq n}\right) = (\tanh[V]_{ab})_{1\leq a,b\leq n}, \ \forall V \in \mathfrak{so}(n).$$

The first one corresponds to the fully complex activation functions, and the second one corresponds to the split complex activation functions from the complex-valued domain.

3 Conclusions

The definition of the Lie algebra-valued bidirectional associative memories was given. Lie algebra-valued neural networks are an alternative generalization of the real-valued neural networks, besides the complex-, quaternion-, and Clifford-valued neural networks.

The existence conditions for the energy function of the proposed network were detailed. Then, the expression of the energy function was given, showing that the proposed function is indeed an energy function for the defined Lie algebra-valued bidirectional associative memory.

It is natural to think that the future will bring even more applications for the complex- and quaternion-valued neural networks, and also for their direct generalization, namely the Clifford-valued neural networks. 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 Lie algebra-valued neural networks might come into play, because their dimension is only n, and thus it allows for more efficient memory use than in the case of Clifford-valued neural networks.

The present work represents another 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 being handled by the network.

References

1. Arena, P., Fortuna, L., Muscato, G., Xibilia, M.: Multilayer perceptrons to approximate quaternion valued functions. Neural Networks **10**(2), 335–342 (1997)
2. 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)
3. Buchholz, S., Sommer, G.: On clifford neurons and clifford multi-layer perceptrons. Neural Networks **21**(7), 925–935 (2008)
4. Hirose, A.: Complex-Valued Neural Networks. Studies in Computational Intelligence, vol. 400. Springer, Heidelberg (2012)
5. Iachello, F.: Lie Algebras and Applications. Lecture Notes in Physics, vol. 891. Springer, Heidelberg (2015)
6. de Kerf, E., Bauerle, G., ten Kroode, A.: Lie Algebras: Finite and Infinite Dimensional Lie Algebras and Applications in Physics. North Holland, Amsterdam (1997)

7. Kosko, B.: Bidirectional associative memories. IEEE Trans. Syst. Man Cybern. **18**(1), 49–60 (1988)
8. Kuroe, Y.: Models of clifford recurrent neural networks and their dynamics. In: International Joint Conference on Neural Networks (IJCNN), pp. 1035–1041. IEEE (2011)
9. Kuroe, Y., Tanigawa, S., Iima, H.: Models of Hopfield-type Clifford neural networks and their energy functions - hyperbolic and dual valued networks. In: Lu, B.-L., Zhang, L., Kwok, J. (eds.) ICONIP 2011. LNCS, vol. 7062, pp. 560–569. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24955-6_67
10. Lee, D., Wang, W.: A multivalued bidirectional associative memory operating on a complex domain. Neural Networks **11**(9), 1623–1635 (1998)
11. Mandic, D., Goh, S.: Complex Valued Nonlinear Adaptive Filters Noncircularity, Widely Linear and Neural Models. Wiley, Hoboken (2009)
12. 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)
13. Nitta, T.: Generalization ability of the three-dimensional back-propagation network. In: International Conference on Neural Networks, vol. 5, pp. 2895–2900. IEEE (1994)
14. Nitta, T.: A quaternary version of the back-propagation algorithm. In: International Conference on Neural Networks, no. 5, pp. 2753–2756. IEEE (1995)
15. Nitta, T.: Three-dimensional vector valued neural network and its generalization ability. Neural Inf. Process. Lett. Rev. **10**(10), 237–242 (2006)
16. 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)
17. Nitta, T.: N-dimensional vector neuron and its application to the N-bit parity problem. In: Complex-Valued Neural Networks: Advances and Applications, pp. 59–74. Wiley (2013)
18. Pearson, J., Bisset, D.: Back propagation in a clifford algebra. In: International Conference on Artificial Neural Networks, vol. 2, pp. 413–416 (1992)
19. Pearson, J., Bisset, D.: Neural networks in the clifford domain. In: International Conference on Neural Networks, vol. 3, pp. 1465–1469. IEEE (1994)
20. Popa, C.A.: Lie algebra-valued neural networks. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–6. IEEE, July 2015
21. Sattinger, D., Weaver, O.: Lie groups and algebras with applications to Physics, Geometry, and Mechanics. Applied Mathematical Sciences, vol. 61. Springer, New York (1986)
22. Vallejo, J., Bayro-Corrochano, E.: Clifford hopfield neural networks. In: International Joint Conference on Neural Networks (IJCNN), pp. 3609–3612. IEEE, June 2008
23. Widrow, B., McCool, J., Ball, M.: The complex lms algorithm. Proc. IEEE **63**(4), 719–720 (1975)
24. Xu, Q., Ma, D.: Applications of lie groups and lie algebra to computer vision a brief survey. In: International Conference on Systems and Informatics (ICSAI), pp. 2024–2029. IEEE, May 2012

Guaranteed Training Set for Associative Networks

Eva Volna(✉) and Martin Kotyrba

Department of Informatics and Computers, University of Ostrava, 30. dubna 22, 70103 Ostrava, Czech Republic
{eva.volna,martin.kotyrba}@osu.cz

Abstract. The focus in this paper is on the proposal of guaranteed patterns in the training set for associative networks. All proposed patterns are pseudoortogonal and they also fulfil stability condition. Patterns were stored into the matrix using Hebb rules for associative networks. In the experimental study, we tested which from the heteroassociative Bidirectional Associative Memory (BAM) and autoassociative Hopfield network is more effective when working with the proposed patterns and what are the possibilities for Hopfield networks when working with real patterns. The comparison was made in order to recognize various damaged images using both types of associative networks. All obtained results are presented in tables or in graphs.

Keywords: Heteroassociative memory · Autoassociative memory · Hopfield network · Bidirectional Associative Memory (BAM)

1 Introduction

Associative memories have the ability to learn patterns from inputs, store a large number of patterns, and retrieve them reliably from noisy or corrupted patterns [4]. A pattern can be stored in memory through a learning process. For an imperfect input pattern, associative memory has the capability to recall the stored pattern correctly by performing a collective relaxation search. Associative memories can be either heteroassociative or autoassociative. For heteroassociation, the input and output vectors range over different vector spaces, while for autoassociation, both the input and output vectors range over the same vector space. Many associative-memory models have been proposed, among which linear associative memories [13], the brain-states-in-a-box (BSB) [3], and bidirectional associative memories (BAMs) [7] can be used as both autoassociative and heteroassociative memories, while the Hopfield model [4], the Hamming network [10], and the Boltzmann machine [1] can only be used as autoassociative models.

Classical associative memory models could only store a linear number of patterns [13]. A primary reason is classical models require memorizing a randomly chosen set of patterns. Advances in associative memory design allow storage of an exponential number of patterns [6], just like in communication systems.

The Hopfield recurrent neural network [4] is a classical autoassociative model of memory, in which collections of symmetrically-coupled neurons interact to perform emergent computation. It is a nonlinear dynamical system, where information retrieval is realized as an evolution of the system state. It can retrieve a pattern stored in memory in response to the presentation of the damaged pattern. This is done by mapping a fundamental memory onto a stable point of a dynamical system. The states of the neurons can be considered as short-term memories while the synaptic weights can be treated as long-term memories. Hopfield networks have potential to solve combinatorial optimization problems [5], store memories as attractors of its deterministic dynamics [2], etc.

2 Associative Memory

Associative memory models are known as content-addressable memories. The function of an associative memory is to recognize previously learned input vectors, even in the case where some noise has been added. A learning algorithm, which can be used to train associative networks, is called Hebbian learning. It is derived from biological neurons [11]. The goal of learning is to associate known input vectors with given output vectors. Contrary to continuous mappings, the neighborhood of a known input vector ξ^{μ} should also be mapped to the image ξ^{ν} of ξ^{μ}, that is if $B(\xi^{\mu})$ denotes all vectors whose distance from ξ^{μ} (using a suitable metric) is lower than some positive constant ε, then we expect the network to map $B(\xi^{\mu})$ to ξ^{ν}. Noisy input vectors can then be associated with the correct output. We can distinguish three overlapping kinds of associative networks [11].

- Heteroassociative networks map m input vectors $\xi^{\mu 1}, \xi^{\mu 2}, \cdots, \xi^{\mu m}$ in n-dimensional space to m output vectors $\xi^{\nu 1}, \xi^{\nu 2}, \cdots, \xi^{\nu m}$ in k-dimensional space, so that $\xi^{\mu i} \mapsto \xi^{\nu i}$. If $\left\| \widetilde{\xi} \mapsto \xi^{\nu i} \right\| < \varepsilon$ then $\widetilde{\xi} \mapsto \xi^{\nu i}$. This should be achieved by the learning algorithm, but becomes very hard when the number m of vectors to be learned is too high.
- Autoassociative networks are a special subset of the heteroassociative networks, in which each vector is associated with itself, i.e., $\xi^{\mu i} = \xi^{\nu i}$ for $i = 1, \cdots, m$. The function of such networks is to correct noisy input vectors.
- Pattern recognition networks are also a special type of heteroassociative networks. Each vector $\xi^{\mu i}$ is associated with the scalar i. The goal of such a network is to identify the 'name' of the input pattern.

3 The Training Set

3.1 Training Set Principles

The training set of associative networks cannot include any patterns, because they have to satisfy certain rule. The networks are not able to learn the whole training set correctly if the patterns are incorrectly completed. When comparing patterns, so-called Hamming distance [10] is often used. The metric determines how many common neurons differing in their behavior (excitatory/inhibitory neurons) are in individual patterns.

If the Hamming distance is small, the error occurs while recalling, if the distance is large (e.g. a shift or rotation of the pattern), errors occur too. The number of patterns should not exceed 15% of the number of neurons [13] (in practice, it has shown that this percentage has to be a little smaller). In [9] is stated a slightly more precise capacity of Hopfield networks, which is 13.8% out of the total number of neurons in the network.

Each pattern should form a local minimum of the energy function in the Hopfield network. All inputs close to the pattern are located in its neighborhood. The training set includes stable patterns if they fulfil the requirement of stability in the process of synaptic weights updating [9]:

$$W_{ij} = \begin{cases} \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^{\mu}\xi_j^{\mu} & \text{if } i \neq j, \\ 0 & \text{if } i = j, \end{cases} \tag{1}$$

where $\xi^{\mu} = (\xi_1^{\mu}, \xi_2^{\mu}, \ldots, \xi_N^{\mu})$ are N-dimensional binary vectors (μ = 1, 2, …, p) that represent selected memory configurations and their total number is p. The condition of a stability is the following:

$$\xi_i^{v} h_i^{v} > 0 \text{ for all } i, \tag{2}$$

where ξ^{v} is a chosen pattern, h^{v} a threshold of an excitation. This means that the local area of the i-th neuron should have the same polarity as the state of the i-th neuron. We get to a limit of large N (in the formula) by sequential adjustment [12]:

$$\xi_i^{v} h_i^{v} = 1 + \xi_i^{v} \sum_{\mu \neq v}^{p} \xi_i^{\mu} \left(\frac{1}{N} \sum_{j=1}^{N} \xi_j^{\mu} \xi_j^{v} \right) = 1 + C_i^{v} > 0, \tag{3}$$

where C_i^{v} is a crosstalk. If $C_i^{v} = 0$ then the pattern ξ^{v} satisfies the condition of stability. $C_i^{v} = 0$ when the patterns are orthogonal to each other. It follows the formula:

$$\xi^{\mu} \cdot \xi^{v} = \frac{1}{N} \sum_{j=1}^{N} \xi_i^{\mu} \xi_j^{v} = 0\, pro\, \mu \neq v. \tag{4}$$

If the patterns are orthogonal, their maximal number is $p_{max} \rightarrow N$. If $\left|C_i^{v}\right| < 1$, stability condition is fulfilled. This state occurs when those patterns are pseudo-orthogonal, i.e. when the mean value of their scalar product over all pairs of patterns $\xi^{\mu} \cdot \xi^{v}$. is approximately equal to zero and when $p \ll N$. In both cases, all samples are stable and the network is able to correct a certain proportion of damaged neurons [12].

Another important variable is crosstalk $m^{\mu}(t)$, which means the degree of similarity between two configurations. A crosstalk has a certain value in a certain time which should be increased up to m^{v} = 1, which means that the network reached an attractor. Using a crosstalk, we can construct a graph that shows the gradual development of recalling during time t [12].

If the patterns are created with equal probability, then we can determine the probability P_{error}. It means that any neutron is unstable and the relationship can be mathematically expressed as:

$$P_{error} = P\left(C_v^i > 1\right). \tag{5}$$

The number of neurons N as well as number of patterns P is important concerning P_{error}. If $N \gg 1$ and $P \gg 1$, then C_i^v. is *1/N* multiply by the sum of approximately N_p random numbers. The value has the binomial probability distribution with zero mean value and with a variance $\sigma^2 = p/N$. If N_p is large enough, we can approximate the binomial distribution via Gaussian distribution with zero mean value and a certain variance. Dependence of P_{error} on a Gaussian distribution can be mathematically expressed as follows:

$$P_{error} = \frac{1}{\sigma\sqrt{2\pi}} \int_1^\infty e^{-x^2/\sigma^2} dx = \frac{1}{2}\left[1 - \operatorname{erf}\left(1/\sqrt{2}\sigma^2\right)\right] = \frac{1}{2}\left[1 - \operatorname{erf}\left(\sqrt{N/2p}\right)\right] \tag{6}$$

where the error function *erf(x)* is:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\left(-u^2\right) du. \tag{7}$$

The dependence of P_{erro} on P_{max}/N is shown in Table 1. For example, when we need that P_{error} equals 0.001, then the maximum number of patterns is less than or equals to $0.105 \times N$. Hopfield memory works reliably, if $P_{max} = 0.138 \times N$. When exceeding this value, a situation can arise in which the network does not remember any of the learned patterns [9].

Table 1. The dependence of a probability P_{error} to the memory capacity of the network

P_{error}	P_{max}/N
0.001	0.105
0.0036	0.138
0.01	0.185
0.05	0.37
0.1	0.61

3.2 Guaranteed Patterns

As associative memory has a small capacity, the proposed patterns had to fulfil the following restrictions:

- to avoid over-fitting a network,
- to take into account the Hamming distance patterns,

- the number of patterns has to be less than 15% of the total number of neurons in the network,
- patterns have to fulfil a stability condition,
- patterns should be orthogonal to each other, or at least pseudo-orthogonal, Eq. (5).

Patterns were designed to represent characters of letters, namely X, I, Z, H, L, P, O, A, M, W (Fig. 1). Single patterns were represented by 15 × 15 pixels in bipolar representation, where "−1" corresponds to white color and "+1" corresponds to black color.

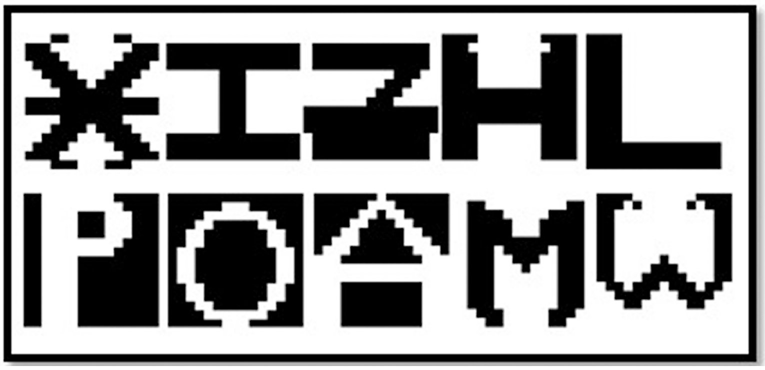


Fig. 1. The proposed patterns

From Table 2, we can see that all pairs of the proposed patterns are pseudo-orthogonal because they fulfil the condition $\left|\xi^x \cdot \xi^i\right| \ll 1$ and they fulfil a stability condition.

Table 2. Values C_i^v between each pair of patterns are shown in absolute value and rounded to three decimal places

	X	I	Z	H	L	P	O	A	M	W
X	#	0.013	0.156	0.307	0.093	0.120	0.138	0.049	0.289	0.067
I	0.013	#	0.413	0.182	0.004	0.031	0.004	0.200	0.040	0.031
Z	0.156	0.413	#	0.013	0.138	0.031	0.049	0.084	0.084	0.031
H	0.307	0.182	0.013	#	0.067	0.253	0.120	0.218	0.307	0.227
L	0.093	0.004	0.138	0.067	#	0.040	0.191	0.067	0.084	0.022
P	0.120	0.031	0.031	0.253	0.040	#	0.191	0.004	0.227	0.209
O	0.138	0.004	0.049	0.120	0.191	0.191	#	0.253	0.271	0.084
A	0.049	0.200	0.084	0.218	0.067	0.004	0.253	#	0.129	0.040
M	0.289	0.040	0.084	0.307	0.084	0.227	0.271	0.129	#	0.076
W	0.067	0.031	0.031	0.227	0.022	0.209	0.084	0.040	0.076	#

Patterns were designed so that the number of excitatory neurons (value 1) and the inhibitory neurons (value −1) was approximately the same. The samples were stored in a matrix (255 × 255) using the Hebb rule for associative networks, Eq. (1).

3.3 Processing of Real Patterns

The basis for processing real patterns is transferring them to grayscale. The color image is displayed on a monitor using the RGB mode. Each pixel has three channels: Red (R), Green (G) and Blue (B). If the image is just a grayscale image, it has to have all color components of equal values. Therefore, we have to convert each pixel using the formula:

$$I = 0,299R + 0,587G + 0,114B, \tag{8}$$

where I is the intensity of gray. Formula (8) is more preferred than only to make average of individual components, since the human eye perceives different colors differently. Although the picture loses color information, the user still clearly recognizes what the picture is. Since each color component (R/G/B) is represented by the value from 0 to 255, the image is represented by the 8-bit depths with 256 shades of gray. After converting the pixel intensity values into binary values, we can separate all eight single levels of gray (Fig. 2). If we do it for all the pixels, we get eight monochrome images, where each pixel can be evaluated either 0 or 1. After that, Hopfield networks is adapted by these particular patterns. Active dynamics runs inversely to the adaptive dynamics, where patterns consist of separate networks in the reverse order to their decomposition [3].

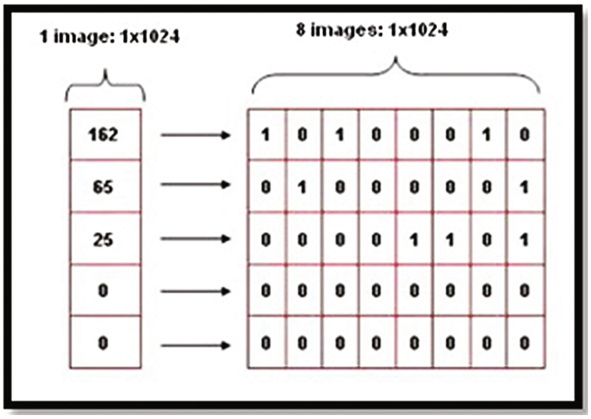


Fig. 2. Illustration of the process of splitting the image into eight binary vectors [3]

Figure 3 shows the process of splitting the image into eight monochrome images. Particular networks are adapted with these patterns. After the recognition process, these monochrome patterns are composed back into a grayscale pattern. It is important to note that half of monochromatic patterns is black. It is caused by the lower bits in the binary representation of the intensity of gray that are full.

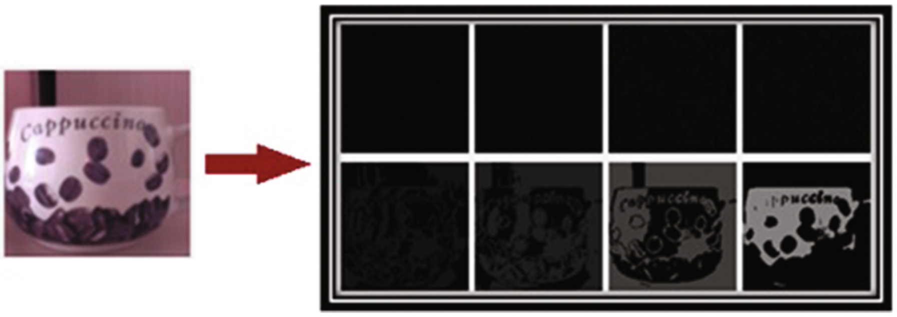


Fig. 3. Distribution of a grayscale pattern into eight monochrome of patterns

The proposed real patterns are shown in Fig. 4. The problem is that these images are colorful, and therefore it is difficult to adapt them by Hopfield network. We chose the possibility to create their monochrome versions for each gray level. The weighting matrix for each gray level is created separately during adaptation.

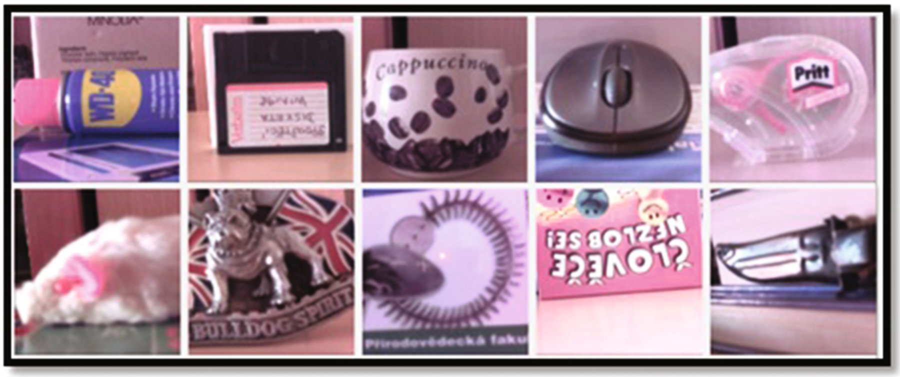


Fig. 4. The proposed real pattern (1 to 10 are shown sequentially by rows)

Active dynamics also runs a distribution of the input image into grayscale. Each gray level is processed separately as a single Hopfield network using its own weighting matrix. The resulting image is then displayed sequentially for each gray level until the resulting grayscale image is complete.

4 Experimental Outcomes

In this chapter, we will test which of the networks with the proposed patterns (BAM or Hopfield network) is more efficient and what possibilities are for Hopfield networks with real patterns. Both networks were adapted by all patterns from Fig. 1. The training

Table 3. Comparison of success rates between recognition of Hopfield network and BAM (values are presented in percentages).

Pattern	HOP_10	BAM_10	HOP_20	BAM_20	HOP_30	BAM_30	HOP_40	BAM_40	HOP_50	BAM_50	HOP_60	BAM_60	HOP_70	BAM_70	HOP_75	BAM_75
X	100	100	100	100	90	100	100	95	70	85	80	60	60	60	60	45
I	100	100	100	85	40	60	80	75	70	60	40	50	40	30	10	25
Z	100	100	100	100	90	100	100	90	50	90	80	75	30	90	30	65
H	100	100	100	100	100	100	100	100	100	100	100	100	80	95	100	90
L	100	100	100	100	100	100	90	100	100	100	80	90	80	80	30	65
P	100	100	100	100	100	100	100	100	100	100	100	90	90	85	80	85
O	100	100	100	85	100	100	100	95	100	100	100	95	70	90	80	75
A	100	85	100	70	100	65	90	50	100	60	80	50	50	30	50	40
M	100	100	100	90	80	80	70	70	80	60	50	100	30	70	10	40
W	100	100	100	100	100	100	100	100	90	100	90	100	70	90	60	65

set of BAM forms ten pairs of patterns, where the associated input and output vectors are the same, therefore both layers have the same number of neurons. The results are shown in Table 3. It is evident that it does not matter to which input of BAM's network the patterns is applied. The patterns were represented with 15 $\times$ 15 pixels (225 neurons). The number of damaged neurons gives the string "xx" after the network name (HOP_xx, BAM_xx). The values are given in per cent and represent the average values from 30 measurements. According to Table 3 [8], we can state that the results of both networks are comparable with regard to the recognition of damaged patterns.

In the next part of the experiment, we adapted Hopfield networks with real patterns (Fig. 4). The experimental procedure was as follows.. The network was adapted by all real patterns and consequently the recognition process with particular undamaged patterns was run. After writing the results of an individual recognition, we performed an evaluation whether the patterns are stable. The results are given in Table 4 [8]. Each pattern is decomposed into eight Hopfield networks according to the depth of gray, so the number of iterations in individual networks and results of recognition are recorded. As it is not possible to guarantee conditions for the stability of the real patterns, it was necessary to determine whether the network adapted by all patterns are stable.

The results indicate that the patterns from Fig. 4 are not stable. The recognition process failed most often in the last bit layer of grayscale image or there were a greater number of iterations needed to detect the image. As the network adaptation by all patterns was not stable, it was necessary to find a stable configuration of the patterns. We found a stable configuration by gradually adapting the network by individual patterns. After adaptation of the first pattern, we checked whether the network was stable, and if so, the network was adapted by the next pattern. We were performing this procedure until the patterns were stable. Stability testing was carried out by verifying whether the adapted patterns were correctly recognized after one iteration. After

Table 4. Recognition of real patterns by Hopfield networks

Pattern	1-st net	2-nd net	3-rd net	4-th net	5-th net	6-th net	7-th net	8-th net	Pattern recognition
1	1	1	1	1	1	1	1	1	A
2	1	1	1	1	1	5	3	1	N
3	1	1	1	1	1	1	1	5	N
4	1	1	1	1	1	1	1	5	N
5	1	1	1	1	1	1	1	4	N
6	1	1	1	1	1	1	14	5	N
7	1	1	1	1	1	1	1	5	N
8	1	1	1	1	1	1	1	1	A
9	1	1	1	1	1	1	1	3	N
10	1	1	1	1	1	1	1	6	N

performing this experiment, we found out that after adaptation by any five patterns, the configuration was stable.

One of the options that we tested was adaptation of the first five patterns from Fig. 4. Figure 5 [8] shows the results of pattern recognition that has different degrees of damage. There were gradually 1024, 2048, 3072, 4096, 5120 and 6144 neurons damaged out of the total of 16384 neurons. Y axis shows the pattern number and the x axis shows the percentage of each pattern recognition.

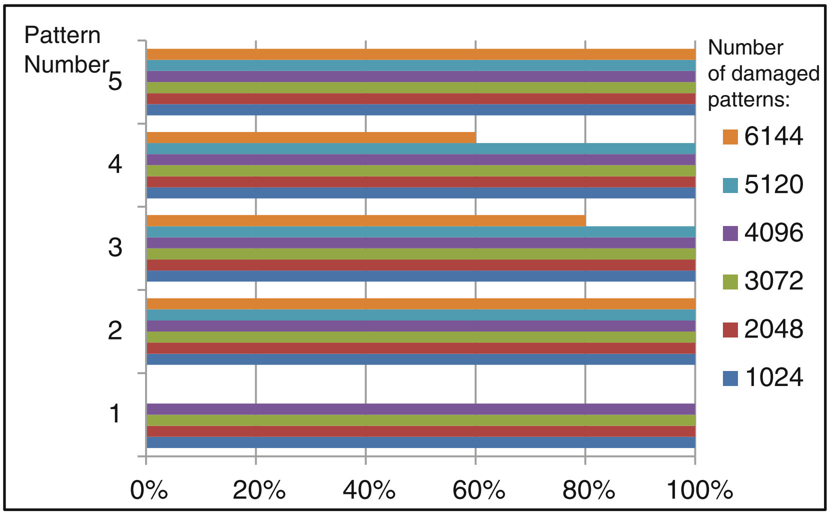


Fig. 5. Recognition of stable patterns 1–5 from Fig. 4

Pattern 2 and 5 belong to the most stable patterns as we can see in Fig. 5. Even if the damage is 37.5% (6144 neurons), the network was always able to recognize the proper pattern. Patterns 3 and 4 also are stable up to the damage 31.25% (5120 neurons). Pattern 1 is the least stable pattern. It remains stable up to 25% of damage (4096 neurons), then it was unrecognizable. If patterns were correctly recognized, only two iterations were needed. The exception is the last layer, where we recorded a higher number of iterations. It is due to the fact that the network which is created by the last bit layer of the grayscale image (pixels with intensity value from 128 up to the 255 are located here) did not properly distribute the neurons according to the rules for stable patterns, Eq. (5).

5 Conclusions

The focus of this paper is on a proposal of guaranteed patterns in the training set for associative networks. Comparative experimental studies were conducted with the proposed patterns in order to detect damaged patterns using heteroassociative Bidirectional Associative Memory (BAM) and autoassociative Hopfield network. According to Table 3, it can be stated that the results of both networks are comparable with regard to the recognition of damaged patterns. Hopfield network also was tested for the purpose of recognition of real patterns (photos). Stability testing of the proposed real patterns is shown in Table 4. As the network adapted by all patterns was not stable, it was necessary to find a stable configuration of the patterns. In conclusion of the experimental verification, we can say that after adaptation of any five patterns from Fig. 4 we received a stable configuration. The results of testing the stable configurations are shown in Fig. 5.

Acknowledgments. The research described here has been financially supported by University of Ostrava grant SGS14/PřF/2016. 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. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cogn. Sci. **9**, 147–169 (1985)
2. Amari, S.I.: Characteristics of sparsely encoded associative memory. Neural Netw. **2**(6), 451–457 (1989)
3. Gonzaga, A., Marin, A., Silva, E., Bertoni, F., Costa, F., Albuquerque, L.: Neutral facial image recognition using parallel Hopfield neural networks. Universidade de São Paulo, São Paulo. http://iris.sel.eesc.usp.br/lavi/pdf/Fabiana_SIBGRAPI.pdf. Accessed 2 Mar 2015
4. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. **79**(8), 2554–2558 (1982)
5. Hopfield, J.J., Tank, D.W.: Neural computation of decisions in optimization problems. Biol. Cybern. **52**(3), 141–152 (1985)
6. Karbasi, A., Salavati, A.H., Shokrollahi, A.: Iterative learning and denoising in convolutional neural associative memories. In: Proceedings of 30th International Conference on Machine Learning, pp. 445–453 (2013)

7. Kosko, B.: Adaptive bidirectional associative memories. Appl. Opt. **26**, 4947–4960 (1987)
8. Kutacek, S.: Hopfield networks and their applications. Master thesis. University of Ostrava, Ostrava (2015). (in Czech)
9. Kvasnička, V., Beňušková, Ľ., Pospíchal, J., Farkaš, I., Tiňo, P., Kráľ, A.: Introduction to the Theory of Neural Networks. IRIS, Bratislava (1997). (in Slovak)
10. Lippmann, R.P.: An introduction to computing with neural nets. IEEE ASSP Mag. **4**(2), 4–22 (1987)
11. Rojas, R.: Neural Networks: A Systematic Introduction. Springer Science & Business Media, New York (2013)
12. Šíma, J., Neruda, R.: Theoretical Questions of Neural Networks. MATFYZPRESS, Praha (1996) (in Czech)
13. Wu, Y., Hu, J., Wu, W., Zhou, Y., Du, K.L.: Storage capacity of the Hopfield network associative memory. In: 2012 Fifth International Conference on Intelligent Computation Technology and Automation, pp. 330–336. IEEE (2012)

Markov Chain for Author Writing Style Profile Construction

Pavels Osipovs(✉), Andrejs Rinkevičs, Galina Kuleshova, and Arkady Borisov

Department of Modelling and Simulation, Riga Technical University, Riga, Latvia
pavels.osipovs@gmail.com, andrejs.rinkevics@rtu.lv, {galina.kulesova, arkadijs.borisovs}@cs.rtu.lv

Abstract. In this paper, the latest results of research in the area of author's personal style profile construction are reviewed. The main goal is to explore the ability to use Markov chain graph, educated based on original author texts to store specifics of his personal writing features. Having such personal profile enables text comparison for authorship confirmation. The ability to do it will be in demand in lot of different areas, for example, authorship detection of scientific articles, or artistic literature texts. This paper describes the main idea offered, the proposed algorithm for two graphs similarity level calculation, the structure of the experimental system created and results of the experiments conducted.

Keywords: Formalization of author writing style · Graphs similarity level · Markov chain · Texts similarity

1 Introduction

Recent studies show that the task of text authorship determination is not solved with the accuracy required to guarantee correct identification yet. Some researchers reported accuracy up to 80% correct results, but mostly in some specific cases only, and no common algorithm that could work with different types of texts is proposed. However, an interest in this kind of task is constantly growing, and comparatively many new studies in the field are being conducted [2].

There are a lot of different people who are interested in finding a solution to the above task in their own areas of interest. They are: *literary critics*, *mathematicians*, *historians*, *philologists*, *lawyers*, *criminalists* etc. [9]. To identify the authorship of some text, typically it is sent to some human experts, literary critics or historians who are able to determine authorship by characteristic language peculiarities and stylistic techniques or identify the author of the unknown text. The ability to automate identification of the authorship makes it is possible to get rid of some essential drawbacks observed in identifying authorship by experts such as time consumption and biases of expert points of view. Still, on the current level of detection accuracy, automated authorship detection may only be used as an additional tool for expert.

Various researchers use different approaches to solve the task of author style detection, e.g.:

- top-k elements approach is discussed in [4];
- the ability to use classical statistical approaches in the area under consideration has been studied in [5];
- support vector machine approach has been used in text classification for authorship attribution analysis [6];
- fuzzy area based methods have shown good results [3] on test data sets;
- probabilistic approach has been used by Microsoft researchers to unite text classifiers under common meta-class for better effectivity [7].

This research is focused on studying the possibility of using Markov chains in the task of text authorship identification [8]. The technique used in the study is based on the ideas described in paper 1, which discusses the system of anomalous action detection using Markov chains, but applied to author's style profile creation. By using this kind of approach, a profile of author writing style is created that uses a Markov chain as the main data structure to store it. Nodes of the chain graph are an unchanged sequence of words of the specified quantity that is taken directly from the text. The weights of the edges characterize the probability of prolonging the sequence of words of the node.

2 Description of the Approach Used

The main purpose of the research described is to check the possibility of constructing a correct profile of author writing style using Markov chains. General scheme of the implemented technique is shown in Fig. 1.

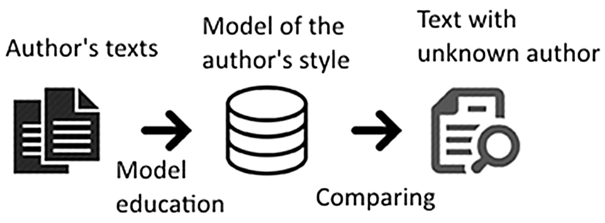


Fig. 1. Common structure of approach

A profile of author writing style is constructed using the texts for which the authorship of the targeted author is 100% known. As a next step, the profile is used to analyze another text, whose authorship is unknown. As a result of analysis, Boolean value will be returned: ***True*** or ***False***, depending, respectively, whether the text under consideration belongs to the target author or not.

2.1 Specifics of Subject Domain

It should be noted that in some cases an author changes his style as a specific literary technique. In addition, modern authors make use of assistance of other people who write some pieces of their works. The volume of the analyzed text and texts used to train the profile are also important. It has to be large enough to contain plenty of information about the specifics of style of its author. It must be stated that as of today texts of small volume are difficult to classify. There exists empirically discovered lower limit of text size that contains enough information – it is 26,000 words. It is intuitively clear that this value will differ depending on the author.

Another special feature of the domain of the task under consideration is translated texts. Commonly, it is highly not recommendable to analyze a text in the non-original language. Maximum that can be done is to compare texts translated by a single translator because literary translation is a complicated task and in the process of translation, the author contributes to the text parts of his own author style.

2.2 Profile Training

Profile training process is conducted based on all the texts collected for which there is a confidence in their authorship. An analyzer goes through each text, and extracts all data needed to update the resulting graph of the author style.

2.3 Profile Using

The use of the profile consists in applying already trained profile to the analysis of the new text. In addition, an iterative procedure is explored that calculates a sequence of values of level metrics of the authorship for a large number of text pieces. The overall result will be the summary mean value of metrics for the whole text. In addition, it is possible to use characteristics that differ from a simple mean value. The procedure of using the existing profile to analyze a new text is described in more detail below.

3 Profile Training Procedure

Profile training is an iterative procedure that builds a graph based on the texts used for education. The construction of the profile of author style includes the following steps:

1. Collection of original texts used to train the profile;
2. Pre-treatment of the collected texts;
3. Pre-treatment settings configuration:
   (a) Should punctuation be removed?
   (b) Specify the size of words that will be named as "*short*".
   (c) Should "*short*" words be deleted?
   (d) Should the word form be normalized?

4. Specification of the parameter of profile learning, *w* - window size (number of words per node of the graph).
5. Direct the learning process based on the application requirements.

When choosing a window size *w*, a dilemma appears: its small values contain too little information, but a big size too precisely adapts to the training set (the classic problem of over-education).

The process of navigating a text consists in an iterative application of several operations.

At the beginning of the creation of the profile, a window is initialized using an empty set of symbols - *Ø*. The initial value of the window *w* is then specified.

Two operations are defined on paths:

- *shift(σ,x)* - which shifts the trace ***σ*** to the left and adds atomic element ***x*** at the end of the track, for example:
  shift(aba,c) = bac.
- *next(σ)* - returns the first character of a trace ***σ*** and moves one position to the left, for example, *next(abcd) = a* and updates path to the state *bcd.*

The initial state of the Markov chain is defined as the trace with length = *w*, consisting of null characters. For example, if the window size is *w* = 3, the initial state will have the form [*Ø,Ø,Ø*].

The process of building the Markov chain includes several steps that are iteratively repeated. To each track of the current set the following operations are applied:

- Let c = *next(σ)*;
- Setting *next_state = shisf(current_state, c)*;
- Increase counter for state *current state* and arc between *current state* and *next state*;
- Update *current state* to having value *next state.*

That is at each iteration two counters values are formed: the value for the current state and the value for the transition from the current state to the next state. There is also set or updated the value of the transition probability for a transition between the current state and next state nodes.

As a result, when the operations described above are applied to all of the analyzed text, a graph will be created containing all present in it combinations of words of a given length, their frequency of use and connection with other combinations.

4 Profile Using Procedure

By the term "*profile using*" we mean its application in the analysis of a new text. The final result will be the use of the vector identity values for each level of the anomalous part of the target text size *w*. The process of calculating each metric anomaly consists in performing a sequence of operations described below.

At the beginning of the analysis, additional global variables ***X*** and ***Y*** are introduced, which are used throughout all analysis iterations. Initially, the variables ***X*** and ***Y*** are equal to zero, the next step changes current state by adding atomic element to the end and removing the initial description of the current state.

On the basis of the previously developed Markov chain that contains an author's style template for each transition between the atomic elements of the text written by an unknown author, metric *$\mu(a)$* is calculated, which denotes the status of the current value of the metric and new values for variables ***X*** and ***Y***.

At each step, there are two ways to calculate the value of the metric by the following algorithm. If the current profile graph contains transition arc from the *previous state* to the *new state* $\beta_i \rightarrow \beta_{i+1}$, then ***X*** and ***Y*** are updated using the following function - parameters:

$$\begin{aligned} \mathrm{Y} &= \mathrm{Y} + \mathrm{F}(\mathrm{s}, (\mathrm{s}, \mathrm{s}')); \\ \mathrm{X} &= \mathrm{X} + \mathrm{G}(\mathrm{s}, (\mathrm{s}, \mathrm{s}')); \end{aligned}$$

Else, if in current profile graph an arc from the *previous state* to the *current state* is not presented, then these functions-parameters are used:

$$\begin{aligned} \mathrm{Y} &= \mathrm{Y} + \mathrm{Z}; \\ \mathrm{X} &= \mathrm{X} + 1. \end{aligned}$$

Finally, the value of metric ***μ(α)*** is calculated, which is equal to Y/X.

Metric ***μ(α)*** shows how well the Markov chain predicts trace α, that is, the smaller its value is, the better the profile predicts the author's style.

Since ***μ*** is parameterized by functions ***F***, ***G*** and the number ***Z***, a different selection of ***F*** and ***G*** will affect the final value of the classifier, which provides the ability to customize the fine specifics of the problem domain.

Metrics of the Difference

Functions ***F*** and ***G*** can be implemented in different ways, depending on the characteristics of the analyzed text. At this time there may be used the following approaches:

- Frequency-based metric;
- Local minimal entropy metric;
- Probabilistic metric.

The summary results do not differ greatly, but depending on the characteristics of the analyzed text, the best results can be shown by different metrics. More details about functions ***F*** and ***G*** calculation can be found on [1].

5 Algorithm for Calculating Similarity Level of Two Markov Chains

For the numerical evaluation of the similarity of two Markov chains, an algorithm is proposed that is based on the analysis of the characteristics of their transition matrices. The value of absolute difference between these two matrices is used to obtain the numerical value of their similarity level. The resulting value is normalized to one, i.e., *zero* is obtained for identical graphs and *one* is obtained for 100% different graphs.

The proposed algorithm for *G1* and *G2* graphs comparison includes several steps:

1. For each of the compared graphs built, its matrix of transition probabilities is constructed: *G1_TM* and *G2_TM* respectively, each of which contains transition probabilities between all nodes.
2. Based on *G1_TM* and *G2_TM* - a summary matrix of absolute transitions differences (*MTA)* is built. For that purpose, the following formula is used:

$$MTA_{ij} = \left|G1\_TM_{ij} - G2\_TM_{ij}\right|, \forall i, j \in N \tag{1}$$

3. Then another matrix (*MTE)* is built with *1* comprising on the intersection of two nodes if there is a transition between these nodes at least in one of the graphs compared.

$$MTE_{ij} = \begin{cases} 1, & G1\_TM_{ij} \neq 0 \vee G2\_TM_{ij} \neq 0 \\ 0, & otherwise \end{cases} \tag{2}$$

4. The final level of the similarity value *S* - is calculated as the sum of all elements of the *sum(MTA)* divided into *sum(MTE)*.

The latter can be formally written as follows (with *n* and *k* is width and height of *MTE* matrix respectively):

$$K = \sum_{i=0}^{n} \sum_{j=0}^{k} MTE_{ij} \tag{3}$$

$$S = \frac{\sum_{i=0}^{n} \sum_{j=0}^{k} MTA_{ij}}{K} \tag{4}$$

As a result, the difference value is equal to *one* for completely different matrices and is *zero* for identical matrices.

An example of similarity level calculation for graphs *G1* and *G2* is given in Fig. 2. It shows two graphs with similar, but not the same transition probabilities.

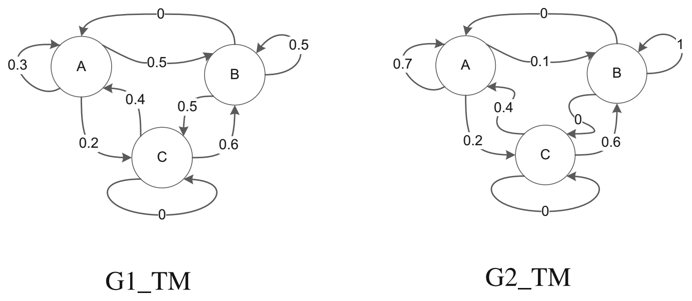


Fig. 2. Graphs compared

0.3	0.5	0.2
0	0.5	0.5
0.4	0.6	0

G1_TM

0.7	0.1	0.2
0	1	0
0.4	0.6	0

G2_TM

0.4	0.4	0
0	0.5	0.5
0	0	0

MTA

1	1	1
0	1	1
1	1	0

MTE

Fig. 3. Transition probabilities matrixes *MTA* and *MTE*, and the resulting similarity level

The matrices of transition probabilities corresponding to graphs *G1* and *G2* graphs are presented in Fig. 3, matrices *MTA* and *MTE*, as well as the final level of the graphs difference are shown in Fig. 3. The resulting value of similarity in this example is ***0,26***.

At present, the main limitation of the algorithm is its ability to compare only those graphs that contain the same number of nodes. However, if necessary, the algorithm can be later modified to work properly to compare graphs with different numbers of nodes.

6 Experimental Software Created

To check how the theoretical approach developed deals with real data, a software system for experiments management was created. As the main programming language, Python [10, 11] was selected, as a powerful and universal tool that may cover all requirements by graphical user interface building, internal data processing and data manipulation and representation. In case of profile education based on a lot of different texts, its size may require too much RAM, because typically one 32-bit process has a limitation of 4 GB memory available. To avoid this limitation, a 64-bit platform was used that has upper limit in 128 GB of RAM per process, which is more than enough for modern PC of researcher.

To build reach and powerful interface, special approach was used, when Python PyQT UI library was used only to show WebUI component scaled to 100% of application area. When HTML5, CSS3 and JavaScript libraries were used to build useful interface, it allowed us to comfortably conduct all the experiments required.

As a bridge between internal Python and interface scripts, special binding functions were used, which allowed easy transfer of JSON data between Python process and his hosted WebUI component internal scripts.

7 Experiments and Results

The main purpose of all experiments was to confirm the ability of author's profile to distinguish texts belonging to him and to another people. As a common way of comparison, similarity level was used. Experiments were carried out on the basis of the texts from 10 authors, each of which presented from 10 and up to 40 texts. The minimal number of words in each of the texts used was set 26,000.

7.1 Comparison of Classification Accuracy Depending on the Window Size Used

One of the interesting results obtained is the comparing classification level at different values of the window size. If the lowest possible window length equal to one atomic element is applied, modeling takes a lot of time. When the dimension of the window is increased, there is observed a growth in the rate of profile construction. This is due to the lessening of linkages between nodes, as there are rarely observed the same windows in the construction.

According to the results of the experiment, the method of step-by-step metric was able to determine the authorship with different sizes of windows in 60–80% of the total work. The accuracy of author detection with different values of window size is illustrated in Fig. 4.

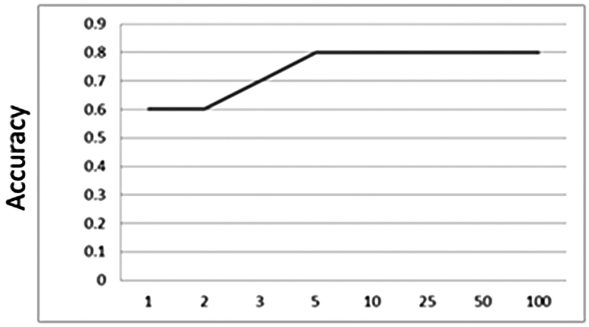


Fig. 4. The success rate for different window sizes

7.2 Impact of Different Profile Settings on the Final Quality of Classification

A large part of experiments was aimed to detect the impact of different types of pre-processing on the accuracy of text classification. The main idea of this set of experiments was to investigate the recognition quality using different dimensions of the window size in the process of the profile education. The continuation of the study foresees elucidation of the effect of text pre-processing on final classification results. The main objective of this experiment was to increase the detection of stylistic factors, accumulating only the information required in the profile.

One of main terms used on this step is *short* and *long* words. We have following possible approaches to mark word as *short* (or *long*):

1. Based on typical Letter Frequency Counts (LFC) statistics for target language;
2. Based on list of mostly used short words for each language. For example on [13] presented list of 50 such words for English. First 5 words (the, of, and, to, in) is mostly often used in all texts analyzed;
3. Just use predefined constant of letters count manually setted in source code.

First and second approaches are less flexible in case of working with texts on different languages, because statistical properties may differs a much [12].

Third approach can be not accurate enough for some exotic languages, but good in case of initial experimentation process. For English, as basic language on the stage of experiments, based on research presented on [13], value of **3** was used as length of *short* words and **13** for *long*.

All experimental results united on a single chart are shown in Fig. 5. According to the results of the experiment, it can be assumed that "*short*" words and punctuation marks do not affect the final quality of classification. In addition, when using the prior removal of "*short*" words, together with the removal of punctuation mark, or separately, the final recognition level does not change and is held at the level of eight recognized works out of 10.

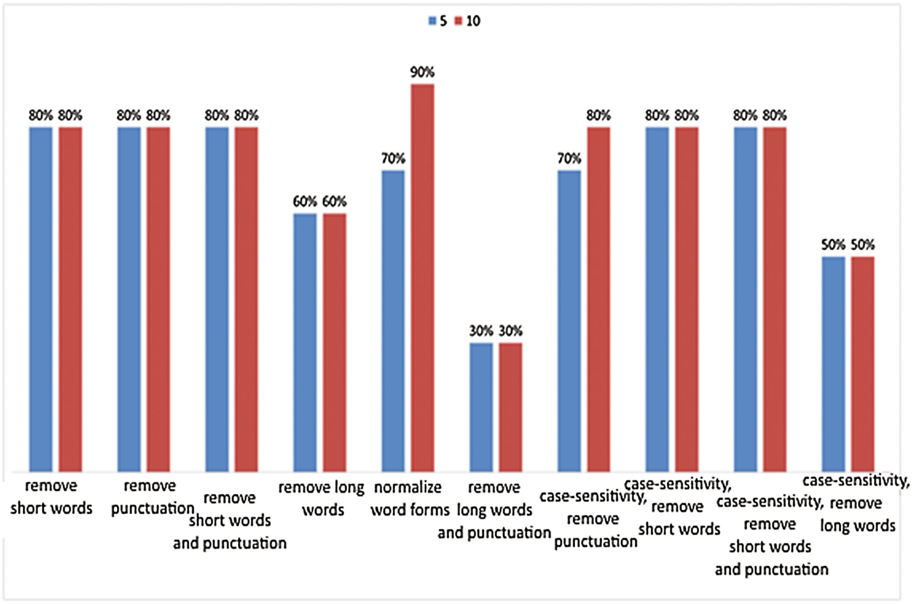


Fig. 5. Results of experimental recognition of texts using different settings for initial profile education and target texts preparation

The applied variants of text pre-processing are as follows:

1. Remove "*short*" words;
2. Remove punctuation;
3. Remove "*short*" words and punctuation marks together;
4. Remove "*long*" words;
5. Normalize word forms across whole text before analyzing it;
6. Remove "*long*" words and punctuation marks;
7. Case-sensitivity, remove punctuation;

8. Case-sensitivity, remove "*short*" words;
9. Case-sensitivity, remove "*short*" words and punctuation marks;
10. Case-sensitivity, remove "*long*" words.

In turn, in the case of prior removal of "*long*" words, experimental results showed that a lot of information about the author style is lost in this case. The accuracy of determination falls by 20–30%. This observation confirms the results of experiments 4, 6 and 10. Together with the removal of punctuation, the recognisability drops to 3 words out of 10 which is the worst indicator of all the experimentally verifiable changes. It looks as if "*long*" words and punctuation accumulate most of the specific information about the author's style.

As a result, we can conclude that the pre-removal of "*short*" words and punctuation marks is a good way to speedup analysis without losing significant level of final classification quality. Classification speed will grow because the resulting graph of the profile will consist of less count of arcs and nodes, which will also decrease the amount of RAM used.

The use of normalization of the word forms to improve recognisability can be a good idea. However, this approach has a weakness in the transformation of the forms of words. Current realization of the conversion algorithm is based on the grammar rules of the language studied, which can distort the author invented speech turns, losing the important elements of the author's style.

8 Conclusions

To summarise the results of this research, it can be declared that the approach described makes it possible to formalize the specifics of the author's style as a software profile - object. To educate such profile, it is enough to have texts written by target author himself. It is assumed that the writer in his work adheres to a certain manner of writing, which makes it possible to use different methods for determining the authorship of his texts.

The experiments conducted have shown the principal possibility of determining the authorship of the text with 60–80% accuracy. In the process of setting up an experimental system, some new ideas how to increase the accuracy of the detection, were obtained, which is the purpose for future research.

References

1. Osipov, P.A., Borisov, A.N.: Abnormal action detection based on Markov models. Autom. Control Comput. Sci. **45**(2), 94–105 (2011)
2. Elayidom, M.S., Jose, C., et al.: Text classification for authorship attribution analysis. Adv. Comput. Int. J. (ACIJ) **4**(5), 1–10 (2013)
3. Homem, N., Carvalho, J.P.: Authorship identification and author fuzzy fingerprints. In: 2011 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS), pp. 1–6. IEEE (2011). 978-1-61284-968-3/11/2011

4. Metwally, A., Agrawal, D., Abbadi, A.: Efficient computation of frequent and top-k elements in data streams, University of California, Santa Barbara, USA, Technical report 2005-23, September 2005
5. Dabagh, R.M.: Authorship attribution and statistical text analysis. Metodološki zvezki **4**(2), 149–163 (2007)
6. Zheng, R., Qin, Y., Huang, Z., Chen, H.: Authorship analysis in cybercrime investigation. In: Chen, H., Miranda, R., Zeng, D.D., Demchak, C., Schroeder, J., Madhusudan, T. (eds.) ISI 2003. LNCS, vol. 2665, pp. 59–73. Springer, Heidelberg (2003). doi:10.1007/3-540-44853-5_5
7. Bennett, P.N., Dumais, S.T., Horvitz, E.: The combination of text classifiers using reliability indicators. Inf. Retrieval **8**(1), 67–100 (2005)
8. Sanderson, C., Guenter, S.: On authorship attribution via Markov chains and sequence kernels. Presented at 18th International Conference on Pattern Recognition (ICPR 2006), Hong Kong, China, 20–24 August 2006
9. Stamatatos, E., Daelemans, W., et al.: Overview of the author identification task at PAN 2014. Presented at CLEF Conference, PAN part, Sheffield, UK, 15–18 September 2014
10. Langtangen, H.P.: A Primer on Scientific Programming with Python. Texts in Computational Science and Engineering, vol. 6, 4th edn., XXXI, 872 p. Springer, Heidelberg (2014). ISBN: 978-3-642-54959-5
11. Johansson, J.R., Nation, P.D., Nori, F.: QuTiP: an open-source Python framework for the dynamics of open quantum systems. Comput. Phys. Commun. **183**(8), 1760–1772 (2012)
12. Smith, R.: Distinct word length frequencies: distributions and symbol entropies. Glottometrics **23**, 7–22 (2012). ISBN: 978-3-942303-17-0
13. Norvig, P.: English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU. http://norvig.com/mayzner.html. Accessed 25 April 2016

Predicting Dust Storm Occurrences with Local Linear Neuro Fuzzy Model: A Case Study in Ahvaz City, Iran

Hossein Iranmanesh[1](✉), Mehdi Keshavarz[1], and Majid Abdollahzade[2]

[1] Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
{hiranmanesh,mehdikeshavarz}@ut.ac.ir

[2] Research Institute for Energy Planning and Management (RIEMP), Tehran, Iran
m.abmdollahzade@pardisiau.ac.ir

Abstract. Dust storm phenomena have vital effects on human life and are significant threat on ecosystem, climate and environmental conditions. Therefore, it may be of vital importance to develop an effective prediction system and mechanism to prevent it and/or reduce its devastating effects. This paper focuses on predicting meteorological conditions associated with dust-storms in the city of Ahvaz, south-western of Iran utilizing local linear neuro fuzzy model with LOLIMOT learning algorithm. For this purpose two different cases are considered. The first case aims to predict the next storm day occurrence and the second case focuses to calculate the number of storm days in next 15 days. The results show that findings under both cases are very close to reality and efficient for predicting dust storm occurrences in Ahvaz city and thus, the methodology could be useful for predicting this event for similar cities as well.

Keywords: Dust storms · LOLIMOT · Ahvaz · Forecasting

1 Introduction

At present environmental issues and related crisis are considered as one of the most important and critical concerns for majority of the world countries. The scope, field and magnitude of the crisis may differ from one place to another and from one country to another one. For example one of the biggest problems for the regions located in the dry and semi dry parts of the world is the phenomenon of dust [1]. Sand and dust storms are common phenomena in the North-China, North-Africa and Australia Middle-East during early summer, spring and winter [2].

Dust storms have adverse effect on people health. The dust particles in the air can cause respiratory diseases, asthma especially. Also, viruses and dust mites carried in the floating particles can lead to various diseases in humans as well as breathing problems. Dust storms can destroy agriculture, crops and livestock, trade and general well-being of the people in the affected area, and even on the nature's ecosystem.

Iran is located in the dry and semi dry belt of the world so that more than 30% of this country is in this belt [3]. Also the west part of Iran is adjacent to neighboring countries' deserts like Iraq, Syria and Saudi Arabia whereas Arabia Peninsula is a major desert source in middle east [4] and one out of five most important regions in the world producing the dust [5]. So Iran is exposed to dust both regionally and locally.

In the recent years with respect to density, continuance, floating particles, dust spreading time and the areas of coverage, the dust phenomenon has been more than what it used to be in the past. In these years continuation of drought, reduction in raining and relative environmental dryness along with other ecological factors aggravation caused by human being such as anomalous usage of desert water resources, canebrakes elimination and wars have caused some of the lakes and lagoons in Iraq and Syria to desiccate. Such events have resulted in severe spread of dust in the region. This phenomenon has had more destructive impact in west and south west part of Iran and in short term has had unfavorable effects on environment, economy and inhabitants health of people living in border cities in west and south west such as Kermanshah, Ilam and Khuzestan. The formal statistics of meteorology suggest that the average days of dusty days in the recent 50 years in cities like Ahwaz, Abadan, Bushehr, Kermanshah have been 68, 76, 75 and 27 days per annum respectively.

Based on the amount of ability to sight the dust and also its severity, the world meteorological organization (WMO) [6] have categorized this phenomenon into 4 groups, severe dust, dust storm, blowing dust and floating dust. As the name implies, the severe dust storm is the most severe one out of these groups somehow it completely covers the sky and decreases horizontal vision ability to 200 m and even sometimes to zero meter. The dust storm decreases horizontal vision ability to 1000 m. Storm of this type carries a large amount of sands and dust upward by strong winds. Another type of dust spreading is blowing dust which is considered as medium ones from severity point of view and decreases horizontal vision from 1000 to 10 km. This phenomenon is generated by winds blowing in high altitude which carry some dust and sand. The weakest type of these storms is the floating dust which decreases the horizontal vision to less than 10 km [7].

The challenging impacts of dust are very clear from economic, social, environmental, political and hygienic point of view. So the successful and on time prediction of this event can remarkably help damage reduction. For example, in the case of prediction of right time dust storm occurrence, it would be possible to notify and warn the pulmonary patients, old people and children not to leave home. Furthermore, it can be possible to provide a suitable preparation for air and ground transportation companies to cancel or find an alternative means for their services in case of dust pollution. It is obvious that the storm decreases sight vision so it can discomfit air plane landing or take off. Therefore, either journeys can be rescheduled or alternative measures can be planned to mobilize neighboring cities or means of transport to prevent undue impedance of necessary movements. The dust storm forecast can have other applications like making preparation for hospitals, fire stations, etc.

To predict the occurrence of dust storms, artificial neural networks (ANN) are one of the most suitable means of prediction systems which have been designed to train and learn complex systems. To date, these networks have been used in several contexts like business, industries, biology, dust spreading occurrence, etc.

ANNs are designed to simulate the way in which the human brain processes information. Human brain consists of a number of simple processing factors called neurons each of which receives a signal that contains some complete information about another neuron or an external stimuli and processes and converts it and generates a processed output using an activating function. Next it sends the related output to the more internal neurons or other neurons. This feature named information processing, changes the artificial neuron system to a powerful computational for learning from examples and generalizing these learning to examples which have not been observed before. The effectiveness and merits of using neural network in monitoring the Earth's atmosphere and environment has highlighted in many studies [8, 9]. Rivas-Perea et al. [10] combined the maximum likelihood classifier with a probabilistic neural network for detecting automatic dust storm. Kaboodvandpour et al. [11] utilized adaptive neuro-fuzzy inference system for predicting dust storm in Sanandaj city of Iran.

A relatively new class of numeral networks is local linear model tree (LOLIMOT) [12]. LOLIMOT is a well-known technique for nonlinear system. The most remarkable advantages of LOLIMOT over ANN are simpler training algorithm, has easier training structure, it will not require initial weighing elements, it shows less sensitivity to noise, it can efficiently create a nonlinear behavior of a system using less neuron compared with ANN [12–14]. LOLIMOT is completely automatic model with very few adjustable parameters.

The effect of dust phenomena in Ahvaz city has been studied in several researches [15–17]. Despite the importance of dust storm prediction in Ahvaz city, the researches devoted to predict dust storm utilizing neural network is limited. The main goal of this paper is to predict the next occurrence dust days in Ahvaz city utilizing local linear neuro fuzzy (LLNF) model with LOLIMOT learning algorithm from 2005 until 2009. For this purpose two different cases are considered. In first case, a time series is generated in which the difference of two following numbers shows the time interval of storm occurrence from the day before. In the second case, years are divided into groups in which each group consists of 15 days. Then the frequencies of dust occurrence in each group were obtained. This method is useful for understanding how many dust storms will happen in the next 15 days.

2 Local Linear Neuro Fuzzy Model

2.1 Model Description

The main approach of the LLNF model is dividing space into small linear subspaces with fuzzy validity function. Each linear subspace with its validity function is described a fuzzy neuron.

Therefore, LLNF is a neuron fuzzy network with one hidden layer and a linear neuron in the output layer which computes the weighted sum of the outputs of locally linear models as

$$y_i = w_{i0} + w_{i1}u_1 + w_{i2}u_2 + \ldots + w_{ip}u_p \tag{1}$$

$$\hat{y} = \sum_{i=1}^{M} y_i \varphi_i(u) \tag{2}$$

The Network structure is shown in Fig. 1, where $u = [u_{1,}u_{2,}\ldots, u_p]^T$ is the model input; M is the number of Locally Linear Model (LMM) neurons and W_{ij} is LLM parameters associated with neuron i. For having appropriate interpretation of validity functions, normalization is necessary. The validity functions are typically chosen as normalized Gaussian function:

$$\varphi_i(u) = \frac{\mu_i(u)}{\sum_{i=1}^{M} \mu_i(u)} \tag{3}$$

$$\mu_i(u) = exp\left(-0.5\left(\frac{(u_1 - c_{i1})^2}{\sigma_{i1}^2} + \ldots + \frac{(u_p - c_{ip})^2}{\sigma_{ip}^2}\right)\right) \tag{4}$$

where c_{ij} and σ_{ij} represent center and standard deviation of normalized Gaussian validity function, respectively.

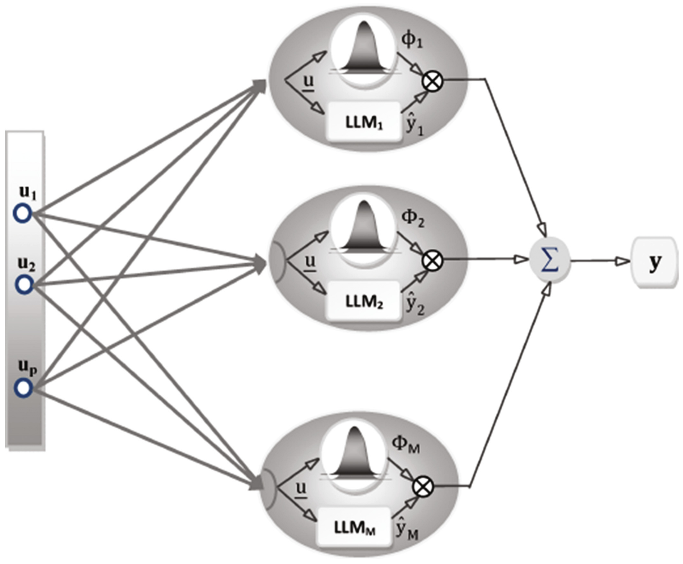


Fig. 1. Structure of LLNF model [18]

The LOLIMOT algorithm according to can be summarized as follows:

1. Initial model: Set M = 1 and $\varphi_i(u) = 1$. Therefore, validity function is covering the whole input space.
2. Finding the worst neuron: Calculate the Mean Square Error for all M neurons and find the neuron with minimum efficiency.
3. Checking all divisions: The worst neuron must be divided into two equal halves in all of p-dimensions. Then, for every two new neurons create fitness functions and estimate secondary parameters for two new neurons. Calculate the overall cost function for the model.
4. Finding the best deviation: Select the direction corresponding to the minimum error. Then, increment the number of neurons from $\mathrm{M} \rightarrow \mathrm{M} + 1$. Stop if the termination criterion is met, otherwise comes back to the step 2.

2.2 Accuracy Criteria

The numerical performance index of prediction is as follows:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}}{Y_i}\right| \times 100 \tag{5}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} \tag{6}$$

3 Case Study and Data

3.1 Case Study Region

Khuzestan is located in southwestern of Iran. This province consists of 16 divisions and 26 cities. The capital of Khuzestan is Ahwaz with geographical location of

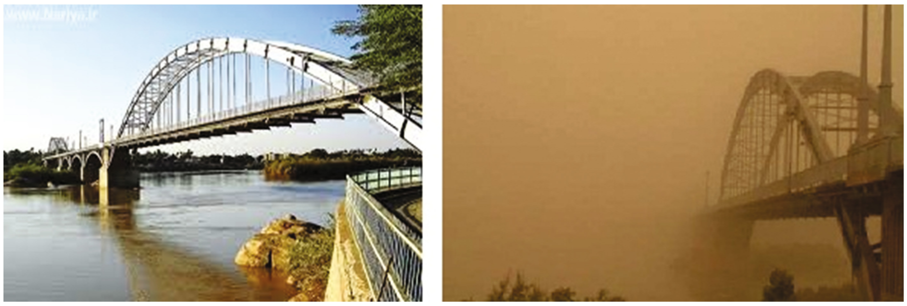


Fig. 2. A bridge on Karun river in Ahvaz on a clear and a dust day

31°20′N, 48°40′E. Ahvaz has a total population of 1.3 million and a total surface area of 220 km^2. Ahvaz is bounded by the major sources of dust storms in the Middle East (Saudi Arabia, Iraq and Kuwait) [19, 20]. Figure 2 compares a clear and a dusty day of Ahvaz city.

3.2 Data

To predict the dust storm in this research, the data from Ahwaz synoptic station are used. These data were analyzed by meteorology organization and the vicious data were omitted from statistics community. In this research the word dust means dust storm that decreases the horizontal vision less than 1000 ms. So those days in which the vision

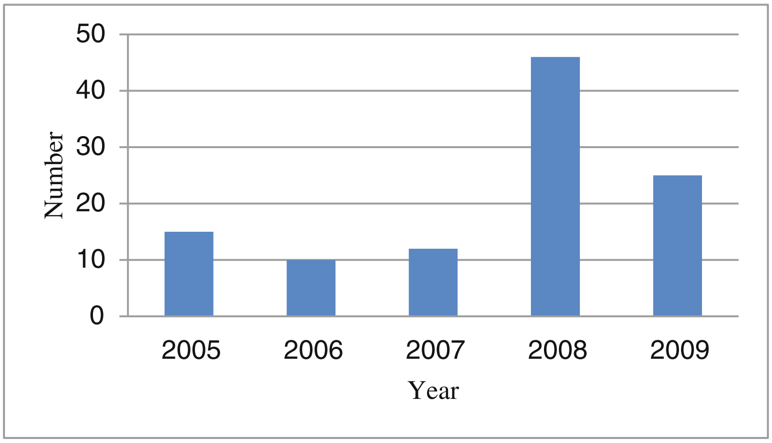


Fig. 3. Number of dust storms in each year

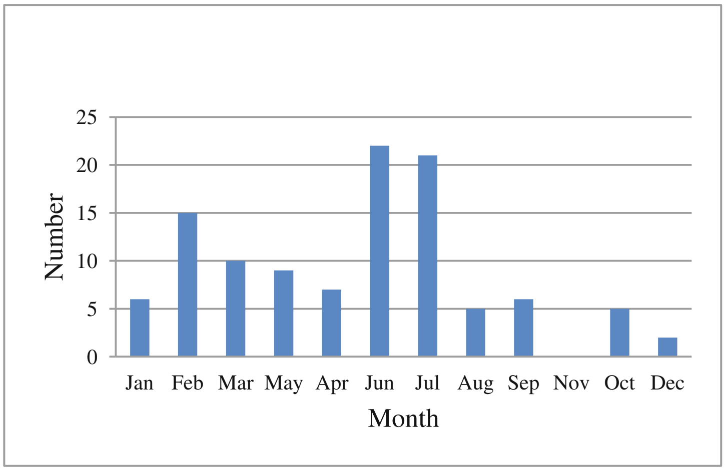


Fig. 4. The aggregated number of dust storm days in each month from 2005 up to 2009

ability was less than 1000 m was extracted from data. After revising it was observed that before 2005 the number of days with dust storm was very limited. Therefore, the data from 2005 to 2009 are used in this research for predicting the dust storms. Figure 3 shows the frequency of dust storm occurrence in each year. As it is observed from the figure, most of dust storms happened in 2008. Also from the figure it is noticed that the frequency of dust storms after 2008 is more than the years 2005, 2006 and 2007.

Figure 4 shows the total dusty days in the terms of months during 2005 to 2009. As it is observed from this figure, in these years the dustiest days were in Jun and July with average 22 and 21 dusty days respectively. The notable point in this figure is November in which there was no dust storm at all. Also from seasonal point of view, winter had the least dust storm.

4 Result

4.1 First Case: Predicting Dust Storm Days

There is no general consensus about the way of dividing data between training and validation. But usually 70 to 90% of data are allocated to training and the remainder to model validation. This paper uses 70% of data for training and the remainder for validation.

In this case, the first calendar day of dust storm occurrence in our data is considered as number 1. The interval between first and second calendar dusty day is added to former number. Thus, a time series is generated in which the difference of two following numbers shows the time interval of storm occurrence from the day before.

The autocorrelation analysis is utilized for proper input selection. Since dust storm day data is the only input of this case, the dust storm day lags leading to the lowest validation error should be selected. For the series generated in this case lags 1 and 2 yielded the better performance. Figure 5 shows the result of predicted and actual dust

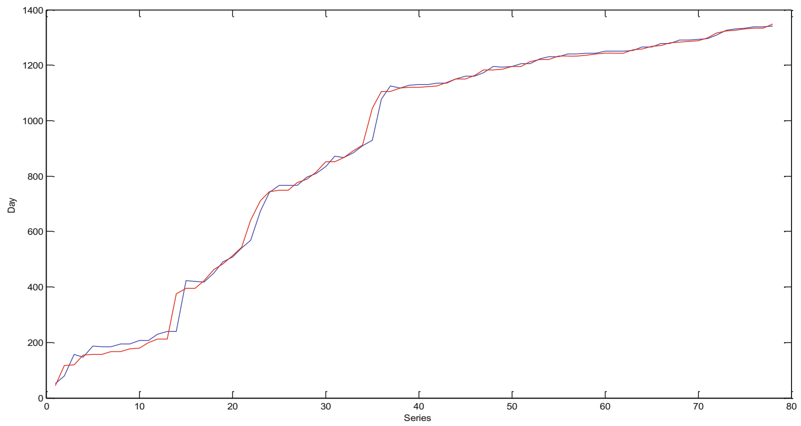


Fig. 5. Actual and predicted dust storm occurrence days for first case

storm day for test data. In Fig. 5, the red color indicates actual data and blue color shows predicted data.

The value of MAPE and R^2 are 0.63 and 0.9602 respectively. The smaller MAPE value and also nearness of R^2 to 1 means a better network from function point of view. Thus we can claim that a pretty good predict has been formed.

4.2 Second Case: Dividing Days into 15 Days Grouping

The years are bundled into groups in which each groups consists of 15 days. Then the frequencies of dust occurrence in each group are obtained. This method helps predict how many dust storms will happen in the next 15 days. For the series generated in this case lags 1 and 2 yielded the better performance. Figure 6 shows the result of predicted and actual dust storm day for test data. In Fig. 6, the red color indicates actual data and blue color shows predicted data.

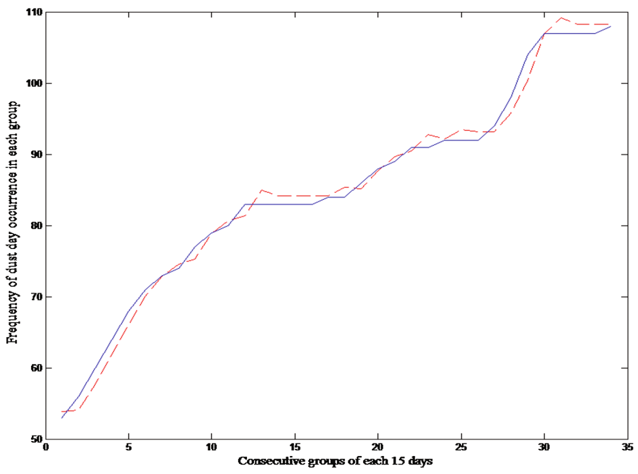


Fig. 6. Actual and predicted dust storm occurrence days for second case

The value of MAPE and R^2 are 1.46 and 0.9602 respectively. The smaller MAPE value and also nearness of R^2 to 1 means a better network from function point of view. Thus, it may be claimed that a pretty good prediction has been formed.

5 Conclusion

Iranian city of Ahvaz is bounded by the major sources of dust storms in the Middle East such as Saudi Arabia, Iraq and Kuwait. Dust storms phenomena have very adverse effects on weather, health, environment and climate of Ahvaz city. In this research, the word dust means dust storm that decreases the horizontal vision less than 1000 meters. It was observed from 2005 to 2009 dustiest days were in Jun and July with average 22 and

21 dusty days respectively. These numbers imply that in these two months there should be more preparation for facing dust phenomenon and there was no dust storm at all in November. This paper focuses on predicting meteorological conditions associated with dust-storms in the Ahvaz city utilizing LLNF model with LOLIMOT learning algorithm.

For this purpose two different cases are considered. The result of first case shows that the value of MAPE and R^2 are 0.63 and 0.9602 respectively; and the result of second case shows that the value of MAPE and R^2 are 1.46 and 0.9602 respectively. Therefore, the performance of both cases has high potential for predicting dust storm occurrences in the city of Ahvaz.

References

1. Orlovsky, L., Orlovsky, N., Durdyev, A.: Dust storms in Turkmenistan. J. Arid Environ. **60**, 83–97 (2005)
2. Tsolmon, R., Ochirkhuyag, L., Sternberg, T.: Monitoring the source of trans-national dust storms in north east Asia. Int. J. Digital Earth **1**, 119–129 (2008)
3. Modarres, R.: Regional maximum wind speed frequency analysis for the arid and semi-arid regions of Iran. J. Arid Environ. **72**, 1329–1342 (2008)
4. Barkan, J., Kutiel, H., Alpert, P.: Climatology of dust sources in North Africa and the Arabian Peninsula, based on TOMS data. Indoor Built Environ. **13**, 407–419 (2004)
5. Rezazadeh, M., Irannejad, P., Shao, Y.: Climatology of the Middle East dust events. Aeol. Res. **10**, 103–109 (2013)
6. WMO, Climate and land degradation. World Meteorological Organization, Switzerland (2005)
7. Wang, S., Yuan, W., Shang, K.: The impacts of different kinds of dust events on PM 10 pollution in Northern China. Atmos. Environ. **40**, 7975–7982 (2006)
8. Knutti, R., Stocker, T., Joos, F., Plattner, G.-K.: Probabilistic climate change projections using neural networks. Clim. Dyn. **21**, 257–272 (2003)
9. Gardner, M.W., Dorling, S.: Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmos. Environ. **32**, 2627–2636 (1998)
10. Rivas-Perea, P., Rosiles, J.G., Murguia, M.I.C., Tilton, J.J.: Automatic dust storm detection based on supervised classification of multispectral data. In: Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Soft Computing for Recognition Based on Biometrics, pp. 443–454. Springer, Heidelberg (2010)
11. Kaboodvandpour, S., Amanollahi, J., Qhavami, S., Mohammadi, B.: Assessing the accuracy of multiple regressions, ANFIS, and ANN models in predicting dust storm occurrences in Sanandaj, Iran. Nat. Hazards **78**, 879–893 (2015)
12. Nelles, O., Fink, A., Isermann, R. Local linear model trees (LOLIMOT) toolbox for nonlinear system identification. In: 12th IFAC Symposium on System Identification (SYSID), Santa Barbara, USA, pp. 845–850 (2000)
13. Nelles, O.: Nonlinear local optimization. In: Nelles, O. (ed.) Nonlinear System Identification, pp. 79–112. Springer, Heidelberg (2001)
14. Nelles, O., Hecker, O., Isermann, R.: Automatic model selection in local linear model trees (LOLIMOT) for nonlinear system identification of a transport delay process. In: Proceedings of IFAC Symposium on System Identification, Kitakyushu, Fukuoka, Japan (1997)

15. Shahsavani, A., Naddafi, K., Haghighifard, N.J., Mesdaghinia, A., Yunesian, M., Nabizadeh, R., Arahami, M., Sowlat, M., Yarahmadi, M., Saki, H.: The evaluation of PM 10, PM 2.5, and PM 1 concentrations during the Middle Eastern Dust (MED) events in Ahvaz, Iran, from April through September 2010. J. Arid Environ. **77**, 72–83 (2012)
16. Shahsavani, A., Naddafi, K., Haghighifard, N.J., Mesdaghinia, A., Yunesian, M., Nabizadeh, R., Arhami, M., Yarahmadi, M., Sowlat, M.H., Ghani, M.: Characterization of ionic composition of TSP and PM10 during the Middle Eastern Dust (MED) storms in Ahvaz, Iran. Environ. Monit. Assess. **184**, 6683–6692 (2012)
17. Goudarzi, G., Shirmardi, M., Khodarahmi, F., Hashemi-Shahraki, A., Alavi, N., Ankali, K.A., Babaei, A.A., Soleimani, Z., Marzouni, M.B.: Particulate matter and bacteria characteristics of the Middle East Dust (MED) storms over Ahvaz, Iran. Aerobiologia **30**, 345–356 (2014)
18. Iranmanesh, H., Abdollahzade, M., Miranian, A., Hassani, H.: A developed wavelet-based local linear neuro fuzzy model for the forecasting of crude oil price. Int. J. Energy Stat. **1**, 171–193 (2013)
19. Léon, J.F., Legrand, M., Mineral dust sources in the surroundings of the North Indian Ocean. Geophys. Res. Lett. **30** (2003)
20. Goudie, A., Middleton, N.: Saharan dust storms: nature and consequences. Earth Sci. Rev. **56**, 179–204 (2001)

Maximum Traveling Salesman Problem by Adapted Neural Gas

Iveta Dirgová Luptáková(✉) and Jiří Pospíchal

Faculty of Natural Sciences, University of SS. Cyril and Methodius in Trnava, 917 01 Trnava, Slovak Republic
iveta.dirgova@ucm.sk, jiri.pospichal@gmail.com

Abstract. This paper considers the problem of solving the Maximum Traveling Salesman Problem (otherwise known as "taxicab ripoff problem") in a plane, using an adapted Neural Gas algorithm with some features of Kohonen's Self-Organizing Map. Maximum Traveling Salesman Problem is similar to classical Traveling Salesmen Problem (TSP), but instead of a search for a Hamiltonian tour visiting all vertices and returning to the initial vertex, which has a minimum sum of lengths of visited edges, we are looking for a maximum sum of lengths. This problem is less popular than classical TSP, nevertheless, in recent years also received a great amount of attention, leading to many heuristics and theoretical results. In this paper, we propose a new heuristic, which is certainly not as efficient as the already existing methods. We are not trying to enter the fierce competition to find the most effective algorithm, but we are trying to experimentally examine a possibility to use a special type of neural network to solve such a problem. Experiments show, that elements of neural gas approach together with SOM are applicable to this kind of problem and provide reasonable results.

Keywords: Longest Hamiltonian cycle · Heuristic · Optimization · Neural gas · Kohonen's SOM · Shortest superstring problem

1 Introduction

Unlike classical Traveling Salesman Problem, the objective of the Maximum Traveling Salesman Problem (MTSP) is to find a tour, in other words, Hamiltonian cycle going through all the vertices, where the total length, i.e. sum of the lengths of the edges, is maximized. We shall require solving the problem in a plane, so the edge weights (i.e. lengths) are positive and the problem shall be symmetric, i.e. the distance between two cities does not depend on the direction of the travel, and the triangle inequality should be satisfied. In the MTSP, the salesman wants to visit each city exactly once and return to the initial city with a maximum possible distance traveled. The problem was firstly comprehensively described as the longest traveling salesman (LTS) by [5, 13], and more recently in [4, 11]. In general, the MTSP could be easily reduced to classical minimum TSP, if we multiplied the weights of edges by −1. However, since TSP with negative lengths or weights of edges is rarely considered in algorithms, we cannot solve the MTSP that easily. Similarly to TSP, there exist boundary conditions when the

MTSP can be solved in polynomial time, but for the standard two-dimensional Euclidean space it is not clear, whether the problem is NP-hard or not (for 3 dimensions it is NP-hard).

Apart from the Metric Maximum TSP, there exist other variants of the problem, like semi-metric Maximum TSP in a directed graph or a symmetric Maximum TSP, where the triangle inequality may not be satisfied. There exists also a similar TSP problem, the Maximum Scatter Traveling Salesman Problem [2] which searches for a tour in which the smallest edge is as large as possible.

The most popular application connected with the Maximum TSP is the shortest superstring problem [12], coming from DNA sequencing in bioinformatics and in data compression. In DNA sequencing the problem is to obtain the sequence of its basis (i.e. the 4 letter string). It is not possible directly, but biochemists can obtain various shorter fragments (substrings) of the molecule and sequence them. Since the fragments come from different copies of the molecule, they can overlap. The goal is to reconstruct the original DNA molecule (superstring) from the overlapping fragments (substrings). It is assumed, that similar to Occam's razor, the shortest superstring (simplest satisfactory explanation) is the right one bioinformatics looks for. The substrings can be equated with vertices and the overlap between two substrings with an edge where the size of the overlap is the length of the edge. The longest Hamiltonian path, therefore, provides the shortest superstring that contains every fragment as a substring. The longest Hamiltonian path is a close problem to the longest Hamiltonian tour, just the first and last vertices are not connected, and moreover, there exists also a circular DNA, where the Maximum TSP solves the sequencing problem exactly. However, a positioning of vertices in a space so that their distances would at least roughly correspond to the length of edges would be very problematic.

A popular heuristic for MTSP is, that a neighboring vertex should be as far as possible [7, 8], and we use it in our adaptation of neural network approach to MTSP.

Neural networks have not been yet used for any variant of MTSP, although for classical variants there are hundreds of papers, more recently e.g. by Kohonen's self-organizing map (SOM) [3, 17] or Hopfield network [9].

For the algorithmic solution of MTSP, mostly variants of classical heuristics for TSP have been used, but a hybrid genetic algorithm was already tried as well [1].

Even though the MTSP variants are generally likely NP-hard, various heuristics, analogously to Traveling Salesman Problem, are very fast and very good [6, 15, 16, 18]. Similarly to the Traveling Salesman Problem, where Hopfield used his neural network to solve a 10-city problem [10], the present paper tries to combine approaches from neural gas and Kohonen's self-organizing map to solve MTSP. The effectiveness of the proposed algorithm is substantially lover and results suboptimal in comparison with the dedicated MTSP algorithms, although the complexity of a stochastic based optimization cannot be directly measured, since unlike deterministic algorithms, more computational time should provide better results. The aim of the paper is to prove workability of the idea, not its dominance over sophisticated heuristics. The further presentation of the new approach will, therefore, provide no direct comparison of its effectiveness against other algorithms. Only empirical statistical results of decrease in quality of results with the increase of the size of the problem for simple cases shall be provided.

2 Neural Gas Approach Combined with SOM Features

Neural gas is an artificial neural network approach, introduced in 1991 [14]. It is typically used for robust clustering, e.g. for DDoS attack classification [19, 20] or other applications. Similarly to other neural network learning approaches, it switches during learning between mapping and training. Mapping finds neurons with weight vectors closest to current input vector (they are of the same dimensions), and training moves the weight vectors of "winning" neurons even closer to the current input vector. Classical neural gas does not change the number of its neurons.

Unlike SOM, the classical neural gas does not contain any predefined "structure" or connections between neurons, and the neurons to be trained are defined only on the basis of their distance from the current input. Neural gas got its name from abrupt moves of the winning neurons, flitting around like gas molecules. These moves are relatively large at the beginning, thus allowing the algorithm to search through the search space properly. This is presumably the reason for the robustness of the algorithm.

However, representation of the solution of the traveling salesman problem by neural network needs some predefined relation between neurons, which would then define the permutation – sequence of visited cities. The simplest solution used in SOM representation of TSP is to connect neurons artificially by edges into a cycle. N neurons would correspond to N cities and, initially, the position of each neuron is set at a random point close to the centroid of the position of cities. The coordinates of the cities would define the input vectors, and the cycle between neurons mapped to cities would define the Hamiltonian cycle. The difference in MTSP is, that the winning neuron should move closer to the city currently selected as the input, but the neighbors of the winning neuron should move towards a city placed as far as possible from the currently selected city. This approach can be applied alternatively to further neurons along the cycle, where neurons with 2 edge distance from the winning neuron should move slightly towards the current input city while neurons 3 edges removed from the winning neuron should move towards the city most distant from the input one, but the move should be even smaller. These moves can be defined by a formula adapted from neural gas

$$\boldsymbol{w}_i^{new} = \boldsymbol{w}_i^{old} + \varepsilon_1 \cdot e^{-k_i/\lambda}\left(\boldsymbol{x} - \boldsymbol{w}_i^{old}\right) \tag{1}$$

$$\boldsymbol{w}_i^{new} = \boldsymbol{w}_i^{old} + \varepsilon_2 \cdot e^{-k_i/\lambda}\left(\boldsymbol{y}_{\boldsymbol{max\_distant\_from\_x}} - \boldsymbol{w}_i^{old}\right) \tag{2}$$

where i is the index of either winning neuron or neurons distant from the winning neuron by an even number of edges for Eq. (1) or neurons distant from the winning neuron by an odd number of edges for Eq. (2); k_i is the number of edges. The position of the city currently presented as input is $\boldsymbol{x}$ and position or weight of a neuron with index i is $\boldsymbol{w_i}$. A position of city most distant from the input city placed in $\boldsymbol{x}$ is $\boldsymbol{y}_{\boldsymbol{max\_distant\_from\_x}}$.

The moves described by Eqs. (1) and (2) are shown in Fig. 1. As the outside circle are given the vertices (cities), for which the Maximum TSP problem should be solved (the dotted edges are shown just to show the circle, they do not have any meaning for the algorithm). Inside, by red circles, are shown positions of neurons in adapted neural gas. Currently, positions of neuron 3 and its neighbors are being adapted. The neuron indexed 3 moves towards the nearest city 9. The move is described by Eq. (1). The neighboring neurons of neuron 3 move towards the farthest city from city 9, in this case towards city 4. These moves are described by Eq. (2), directions of all three moves are shown by black arrows. If there are more of farthest cities, as in this case are cities 3 and 4, one of them is always randomly selected anew. This accentuates the jerky movements of neural gas algorithm. In the standard neural gas, the parameter ε is an adaptation step size and λ is neighborhood range. In the standard neural gas the Eq. (2) would be missing and parameters ε and λ would be reduced with increasing number of presented points by elaborate equations. In our simplified version of the algorithm, we set $e^{-k_i/\lambda} = 1$ for both equations, $\varepsilon = 0.7$ for the winning neuron in Eq. (1) and $\varepsilon = 0.8$ for its direct neighbors in the predefined circle. The cities are presented as input always in different permutation sequence. In case the winning neuron closest to the currently presented city had already been the closest to another city in the currently presented permutation, the currently unselected neuron closest to the city is selected as winning, so that each city would attract only one neuron as winning in each permutation. After a random permutation of all cities was presented as inputs and the position of winning neuron and its neighbors in the cycle were changed by Eqs. (1) and (2) after a presentation of each city, the ε_2 was simply adapted by $\varepsilon_2 = \varepsilon_2 * 0.95$. The values of parameters were selected by a local hill climbing optimization procedure. The adaptation was stopped, after the sum of differences between positions of neurons and positions of cities closest to them was smaller than a given threshold 0.1.

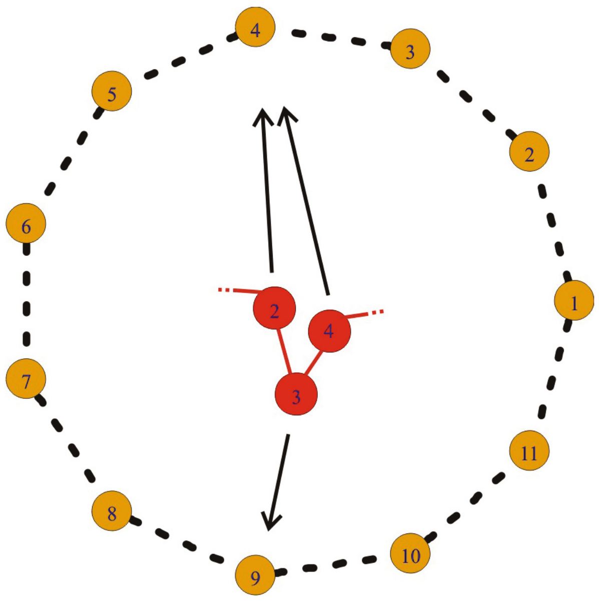


Fig. 1. The principle of adaptation of neural gas for Maximum Traveling Salesman Problem shown for 11 cities positioned on a circle.

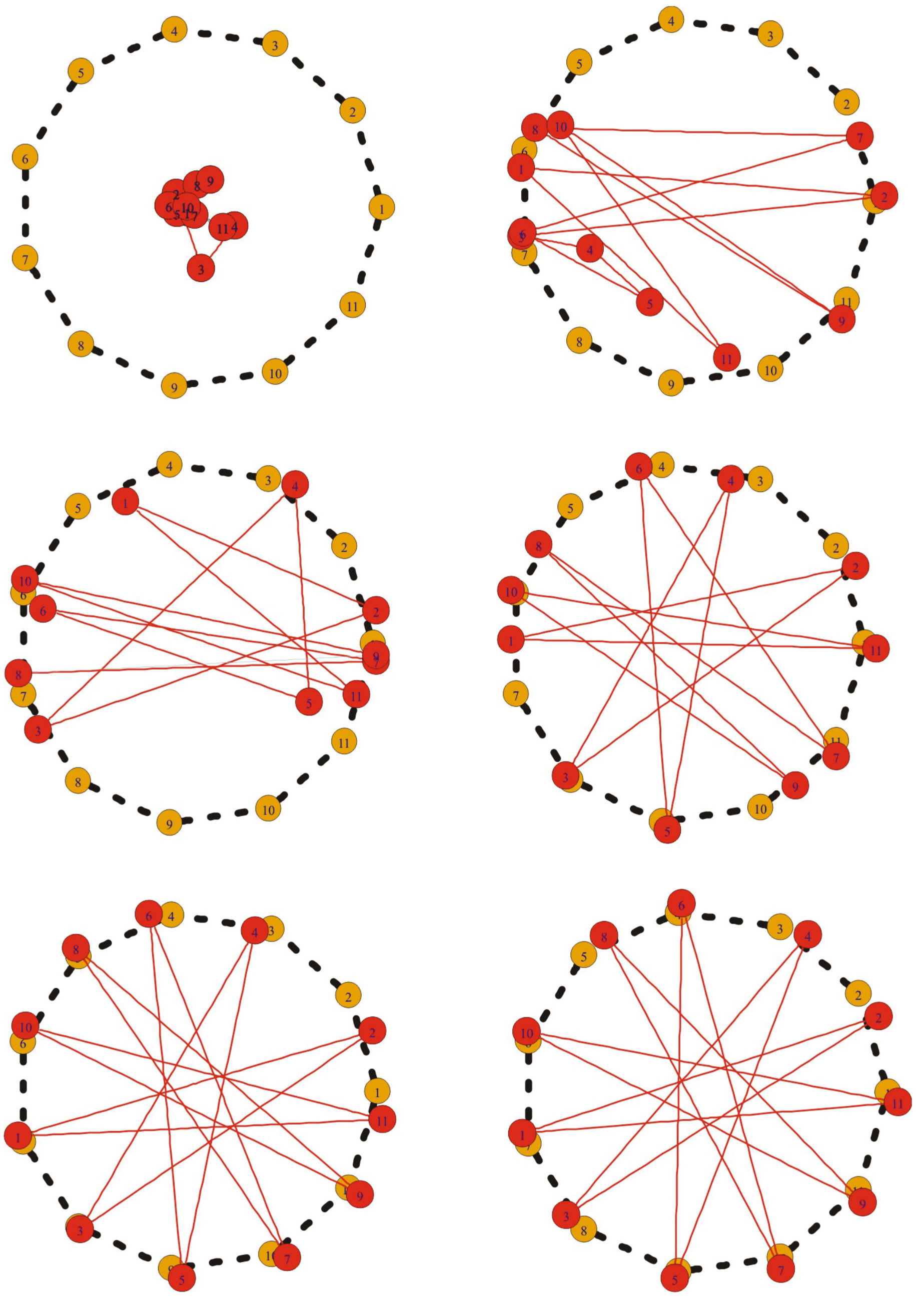


Fig. 2. Example of 11 cities positioned in a circle and of initial positions of neurons and by rows for the 1st, 5th, 10th, 15th, and 25th iteration of adapted neural gas algorithm for MTSP.

The adaptation steps in Eqs. (1) and (2) are in fact equivalent to gradient descent method in classical neural 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 position, but its neighbors change their positions as well.

3 Experimental Results

The adapted neural gas was used to find the tour of maximum length for cities positioned regularly on a circle of diameter 2. An example of iterations of the adapted neural gas algorithm for Maximum Traveling Salesman Problem on 11 cities in a circle can be seen in Fig. 2. From the randomly placed initial positions of neurons near the center of the figure, the first iteration results in positions of the neurons near the circumference but in totally wrong places. It is evident from difference in positions between 1st, 5th and 10th iteration, that in the beginning the algorithm flits neurons wildly around. Only the 10th iteration starts to look acceptable, and from 15th to 25th iteration already there are not many changes.

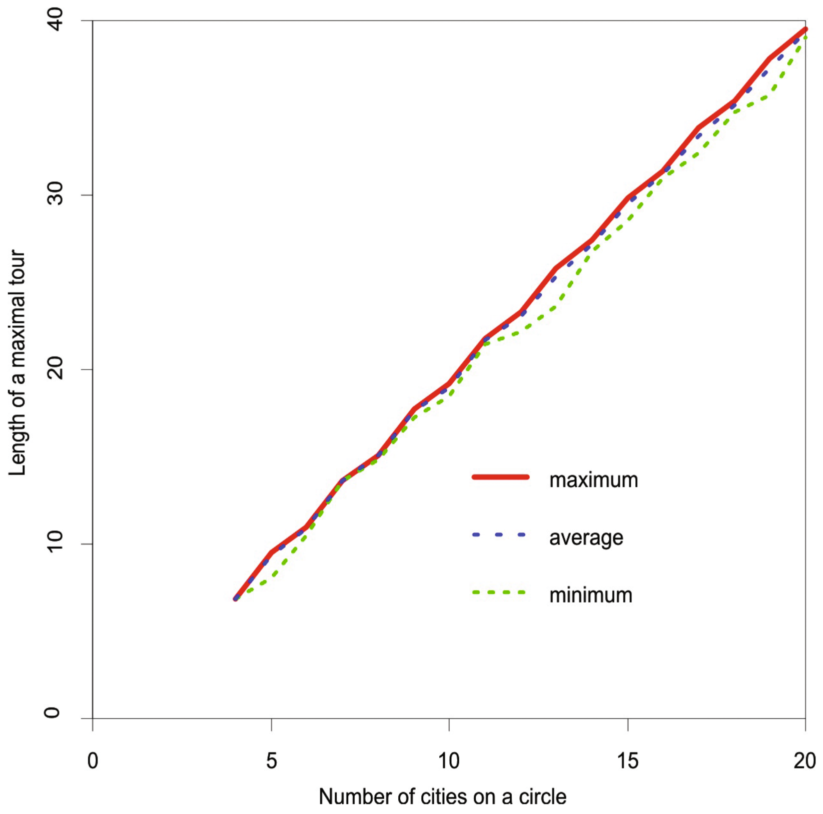


Fig. 3. Maximum, minimum and average lengths of Maximum Traveling Salesman Problem for 4–20 cities positioned on a circle.

The Fig. 3 shows the results from ten runs for each number of cities on a circle. The number of cities ranged from 4 up to 20. While the minimal length of a maximal tour from 10 runs sometimes slightly differs from the global optimum, the average and maximum lengths of the maximal tours are pretty similar, showing that suboptimal results do not appear often. It should be noted, that while in the case of odd number of cities the neighbors in MTSP Hamiltonian cycle should be the cities opposite of the current city across the circles diameter, in the case of even number of cities the ideal case is not that regular and therefore the adapted neural gas algorithm for these cases has more local optima.

4 Conclusions

The adapted neural gas provided acceptable solutions for the Maximum Traveling Salesman Problem, even though the method was not designed or ever before used for such a purpose and had to be substantially adapted. Although only one type of positioning of the cities was tested, other types and positions in multidimensional cases 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 dedicated heuristics. Future improvements of this type of algorithm might employ more sophisticated adaptation rate changes.

References

1. Ahmed, Z.H.: An experimental study of a hybrid genetic algorithm for the maximum traveling salesman problem. Math. Sci. **7**(1), 1–7 (2013)
2. Arkin, E.M., Chiang, Y.-J., Mitchell, J.S.B., Skiena, S.S., Yang, T.-C.: On the maximum scatter traveling salesperson problem. SIAM J. Comput. **29**(2), 515–544 (1999)
3. Avşar, B., Aliabadi, D.E.: Parallelized neural network system for solving Euclidean traveling salesman problem. Appl. Soft Comput. **34**, 862–873 (2015)
4. Barvinok, A., Gimadi, E.K., Serdyukov, A.I.: The maximum traveling salesman problem. In: Gutin, G., Punnen, A. (eds.) The Traveling Salesman Problem and Its Variations, pp. 585–607. Kluwer Academic Publishers, Dordrecht (2002)
5. Blokh, D., Gutin, G.: Maximizing traveling salesman problem for special matrices. Discrete Appl. Math. **56**, 83–86 (1995)
6. Dudycz, S., Marcinkowski, J., Paluch, K., Rybicki, B.: A 4/5-Approximation Algorithm for the Maximum Traveling Salesman Problem (2015). arXiv preprint arXiv:1512.09236
7. Fekete, S.P., Meijer, H., Rohe, A., Tietze, W.: Solving a Hard problem to approximate an Easy one: heuristics for maximum matchings and maximum traveling salesman problems. J. Exp. Algorithmics (JEA) **7**, 11 (2002)
8. Fekete, S.P.: Finding longest geometric tours. In: Schulz, A.S., Skutella, M., Stiller, S., Wagner, D. (eds.) Gems of Combinatorial Optimization and Graph Algorithms, pp. 29–36. Springer, Cham (2015)
9. García Rodriguez L., Talaván P.M., Yañez Gestoso, F.J.: Improving the Hopfield model performance when applied to the traveling salesman problem: a divide-and-conquer scheme. Soft Comput., 1–15 (2016)

10. Hopfield, J.J., Tank, D.W.: “Neural” computation of decisions in optimization problems. Biol. Cybern. **52**(3), 141–152 (1985)
11. Ilavarasi, K., Joseph, K.S.: Variants of travelling salesman problem: a survey. In: Information Communication and Embedded Systems (ICICES), pp. 1–7. IEEE (2014)
12. Karpinski M., Schmied R.: Improved inapproximability results for the shortest superstring and related problems. In: Wirth, A. (ed.) Proceedings of the Nineteenth Computing: The Australasian Theory Symposium, (CATS 2013), vol. 141, pp. 27–36. Australian Computer Society, Inc., Darlinghurst, Australia (2013)
13. Lewenstein M., Sviridenko, M.: Approximating asymmetric maximum TSP. In: Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 646–654 (2003)
14. Martinetz, T., Schulten, K.: A “neural gas” network learns topologies. In: Kohonen, T., et al. (ed.) Artificial Neural Networks, pp. 397–402. Elsevier (1991)
15. Niel, B.I.: Longest Hamiltonian in Nodd-Gon. Open J. Discrete Math. **3**(02), 75 (2013)
16. Niel, B.I.: Every longest Hamiltonian path in even N-Gons. Discrete Math. Algorithms Appl. **4**(04), 1250057 (2012)
17. Ryter R., Stauffer, M., Hanne, T., Dornberger, R.: Analysis of chaotic maps applied to self-organizing maps for the Traveling Salesman Problem. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1717–1724 (2015)
18. Shenmaier, V.V.: Asymptotically optimal algorithms for geometric Max TSP and Max m-PSP. Discrete Appl. Math. **163**, 214–219 (2014)
19. Šimon, M., Huraj, L., Čerňanský, M.: Performance evaluations of IPTables firewall solutions under DDoS attacks. J. Appl. Math. Stat. Inform. **11**(2), 35–45 (2015)
20. Zhong, S., Khoshgoftaar, T.M., Seliya, N.: Clustering-based network intrusion detection. Int. J. Rel. Qual. Saf. Eng. **14**, 169–187 (2007)

Conjugate Gradient Algorithms for Quaternion-Valued Neural Networks

Călin-Adrian Popa(✉)

Department of Computer and Software Engineering,
Polytechnic University Timişoara,
Blvd. V. Pârvan, No. 2, 300223 Timişoara, Romania
calin.popa@cs.upt.ro

Abstract. This paper introduces conjugate gradient algorithms for training quaternion-valued feedforward neural networks. Because these algorithms had better performance than the gradient descent algorithm in the real- and complex-valued cases, the extension to the quaternion-valued case was a natural idea. The classical variants of the conjugate gradient algorithm are deduced starting from their real-valued variants, and using the framework of the HR calculus. The resulting quaternion-valued training methods are exemplified on time series prediction applications, showing a significant improvement over the quaternion gradient descent algorithm.

Keywords: Quaternion-valued neural networks · Conjugate gradient algorithm · Time series prediction

1 Introduction

Over the last few years, the domain of quaternion-valued neural networks has received an increasing interest. Some popular applications of these networks include chaotic time-series prediction [2], color image compression [10], color night vision [13], polarized signal classification [5], and 3D wind forecasting [11,23,24].

In the 3D and 4D domains, where some signals are naturally expressed in quaternion-valued form, these networks appear as a natural choice for solving problems such as time series prediction. Several methods have been proposed to increase the efficiency of learning in quaternion-valued neural networks. These methods include different network architectures and different learning algorithms, some of which are specially designed for this type of networks, while others are extended from the real-valued case.

One method that has been proven to be very efficient in the real-valued and complex-valued [17] cases is the conjugate gradient learning algorithm. First proposed, among others, by [6,12], this algorithm has become today one of the most known and used methods to train feedforward neural networks. Having this idea in mind, it seems natural to extend this learning algorithm to quaternion-valued neural networks, also.

In this paper, we deduce the quaternion-valued conjugate gradient algorithm starting from the real-valued case, by using the framework of the $\mathbb{HR}$ calculus. We test the proposed conjugate gradient algorithms on linear and chaotic time series prediction problems.

The remainder of this paper is organized as follows: Sect. 2 is an introduction to the $\mathbb{HR}$ calculus, which is a type of calculus used for extending real-valued algorithms to the quaternion-valued domain. Section 3 is concerned with the derivation of the quaternion-valued conjugate gradient algorithms. The experimental results of the three applications of the proposed algorithms are shown and discussed in Sect. 4, along with a description of each problem. Section 5 is dedicated to presenting the conclusions of the study.

2 The $\mathbb{HR}$ Calculus

We will first present the basics of the $\mathbb{HR}$ calculus [26], which will be later used to deduce the conjugate gradient algorithms for a quaternion-valued error function.

Let $\mathbb{H} = \{q_a + iq_b + jq_c + kq_d | q_a, q_b, q_c, q_d \in \mathbb{R}\}$ be the algebra of quaternions, where i, j, k are the imaginary units which satisfy $i^2 = j^2 = k^2 = ijk = -1$. For any $\mu \in \mathbb{H}$, we define the operation $q^\mu := \mu q \mu^{-1}$. Then, for any $q = q_a + iq_b + jq_c + kq_d \in \mathbb{H}$, we have $q^i = iqi^{-1} = q_a + iq_b - jq_c - kq_d$, $q^j = jqj^{-1} = q_a - iq_b + jq_c - kq_d$, $q^k = kqk^{-1} = q_a - iq_b - jq_c + kq_d$. If $f : \mathbb{H} \to \mathbb{H}$, we can define the $\mathbb{HR}$ derivatives of f by

$$\begin{pmatrix} \frac{\partial f}{\partial q} \\ \frac{\partial f}{\partial q^i} \\ \frac{\partial f}{\partial q^j} \\ \frac{\partial f}{\partial q^k} \end{pmatrix} := \frac{1}{4} \begin{pmatrix} 1 & -i & -j & -k \\ 1 & -i & j & k \\ 1 & i & -j & k \\ 1 & i & j & -k \end{pmatrix} \begin{pmatrix} \frac{\partial f}{\partial q_a} \\ \frac{\partial f}{\partial q_b} \\ \frac{\partial f}{\partial q_c} \\ \frac{\partial f}{\partial q_d} \end{pmatrix}.$$

Now, consider a quaternion vector $\mathbf{q} = (q_1, q_2, \ldots, q_N)^T \in \mathbb{H}^N$, which can be written as $\mathbf{q} = \mathbf{q}_a + i\mathbf{q}_b + j\mathbf{q}_c + k\mathbf{q}_d \in \mathbb{H}^N$, where $\mathbf{q}_a$, $\mathbf{q}_b$, $\mathbf{q}_c$, $\mathbf{q}_d \in \mathbb{R}^N$. We have that $\mathbf{q}^i = -i\mathbf{q}i = \mathbf{q}_a + i\mathbf{q}_b - j\mathbf{q}_c - k\mathbf{q}_d$, $\mathbf{q}^j = -j\mathbf{q}j = \mathbf{q}_a - i\mathbf{q}_b + j\mathbf{q}_c - k\mathbf{q}_d$, $\mathbf{q}^k = -k\mathbf{q}k = \mathbf{q}_a - i\mathbf{q}_b - j\mathbf{q}_c + k\mathbf{q}_d \in \mathbb{H}^N$, which can be written compactly as

$$\begin{pmatrix} \mathbf{q} \\ \mathbf{q}^i \\ \mathbf{q}^j \\ \mathbf{q}^k \end{pmatrix} = \begin{pmatrix} \mathbf{I}_N & i\mathbf{I}_N & j\mathbf{I}_N & k\mathbf{I}_N \\ \mathbf{I}_N & i\mathbf{I}_N & -j\mathbf{I}_N & -k\mathbf{I}_N \\ \mathbf{I}_N & -i\mathbf{I}_N & j\mathbf{I}_N & -k\mathbf{I}_N \\ \mathbf{I}_N & -i\mathbf{I}_N & -j\mathbf{I}_N & k\mathbf{I}_N \end{pmatrix} \begin{pmatrix} \mathbf{q}_a \\ \mathbf{q}_b \\ \mathbf{q}_c \\ \mathbf{q}_d \end{pmatrix},$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix. We denote

$$\overset{\mathcal{H}}{\mathbf{q}} := \begin{pmatrix} \mathbf{q} \\ \mathbf{q}^i \\ \mathbf{q}^j \\ \mathbf{q}^k \end{pmatrix} \in \mathbb{H}^{4N},\ \overset{\mathcal{R}}{\mathbf{q}} := \begin{pmatrix} \mathbf{q}_a \\ \mathbf{q}_b \\ \mathbf{q}_c \\ \mathbf{q}_d \end{pmatrix} \in \mathbb{R}^{4N},\ \mathbf{J} := \begin{pmatrix} \mathbf{I}_N & i\mathbf{I}_N & j\mathbf{I}_N & k\mathbf{I}_N \\ \mathbf{I}_N & i\mathbf{I}_N & -j\mathbf{I}_N & -k\mathbf{I}_N \\ \mathbf{I}_N & -i\mathbf{I}_N & j\mathbf{I}_N & -k\mathbf{I}_N \\ \mathbf{I}_N & -i\mathbf{I}_N & -j\mathbf{I}_N & k\mathbf{I}_N \end{pmatrix}.$$

With these notations, the above relation becomes

$$\overset{\mathcal{H}}{\mathbf{q}} = \mathbf{J}\overset{\mathcal{R}}{\mathbf{q}}.$$

It can be verified that $\mathbf{J}^H\mathbf{J} = \mathbf{J}\mathbf{J}^H = 4\mathbf{I}_{4N}$, and so we also have that

$$\overset{\mathcal{R}}{\mathbf{q}} = \frac{1}{4}\mathbf{J}^H\overset{\mathcal{H}}{\mathbf{q}}. \tag{1}$$

A function $f : \mathbb{H}^N \to \mathbb{R}$ can now be seen in three equivalent forms

$$f(\mathbf{q}) \Leftrightarrow f(\overset{\mathcal{H}}{\mathbf{q}}) := f(\mathbf{q}, \mathbf{q}^i, \mathbf{q}^j, \mathbf{q}^k) \Leftrightarrow f(\overset{\mathcal{R}}{\mathbf{q}}) := f(\mathbf{q}_a, \mathbf{q}_b, \mathbf{q}_c, \mathbf{q}_d).$$

If we define

$$\frac{\partial f}{\partial \mathbf{q}} := \left(\frac{\partial f}{\partial q_1}, \ldots, \frac{\partial f}{\partial q_N}\right),$$
$$\frac{\partial f}{\partial \overset{\mathcal{H}}{\mathbf{q}}} := \left(\frac{\partial f}{\partial \mathbf{q}}, \frac{\partial f}{\partial \mathbf{q}^i}, \frac{\partial f}{\partial \mathbf{q}^j}, \frac{\partial f}{\partial \mathbf{q}^k}\right),$$
$$\frac{\partial f}{\partial \overset{\mathcal{R}}{\mathbf{q}}} := \left(\frac{\partial f}{\partial \mathbf{q}_a}, \frac{\partial f}{\partial \mathbf{q}_b}, \frac{\partial f}{\partial \mathbf{q}_c}, \frac{\partial f}{\partial \mathbf{q}_d}\right),$$

we have, from the chain rule, that

$$\frac{\partial f}{\partial \overset{\mathcal{H}}{\mathbf{q}}} = \frac{1}{4}\frac{\partial f}{\partial \overset{\mathcal{R}}{\mathbf{q}}}\mathbf{J}^H \Leftrightarrow \frac{\partial f}{\partial \overset{\mathcal{R}}{\mathbf{q}}} = \frac{\partial f}{\partial \overset{\mathcal{H}}{\mathbf{q}}}\mathbf{J}.$$

Now, if we define $\nabla_{\mathbf{q}} f := \left(\frac{\partial f}{\partial \mathbf{q}}\right)^H$, $\nabla_{\overset{\mathcal{H}}{\mathbf{q}}} f := \left(\frac{\partial f}{\partial \overset{\mathcal{H}}{\mathbf{q}}}\right)^H$, $\nabla_{\overset{\mathcal{R}}{\mathbf{q}}} f := \left(\frac{\partial f}{\partial \overset{\mathcal{R}}{\mathbf{q}}}\right)^T$, where $(\cdot)^T$ and $(\cdot)^H$ represent the transpose and the Hermitian transpose, respectively, the above relations can be written as

$$\nabla_{\overset{\mathcal{H}}{\mathbf{q}}} f = \frac{1}{4}\mathbf{J}\nabla_{\overset{\mathcal{R}}{\mathbf{q}}} f \Leftrightarrow \nabla_{\overset{\mathcal{R}}{\mathbf{q}}} f = \mathbf{J}^H\nabla_{\overset{\mathcal{H}}{\mathbf{q}}} f. \tag{2}$$

3 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 perform better than the classical gradient descent. For the full deduction of conjugate gradient algorithms in the real-valued case, see [4,12].

Let's assume that we have a quaternion-valued neural network with an error function denoted by $E : \mathbb{H}^N \to \mathbb{R}$, and an N-dimensional weight vector denoted by $\mathbf{w} \in \mathbb{H}^N$. We start with the conjugate gradient algorithm for the real-valued case, in which the function $E(\mathbf{w})$ can be viewed as $E(\overset{\mathcal{R}}{\mathbf{w}})$. The iteration for calculating the value $\overset{\mathcal{R}}{\mathbf{w}}^*$, for which the minimum of the function $E(\overset{\mathcal{R}}{\mathbf{w}})$ is attained, is

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k, \tag{3}$$

where $\overset{\mathcal{R}}{\mathbf{p}}_k \in \mathbb{R}^{4N}$ represents the search direction. The value of $\alpha_k \in \mathbb{R}$ can be determined exactly, but because of the computational burden, we can replace the explicit calculation of α_k with an inexact line search that minimizes $E(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}) = E(\overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k)$, i.e. a line minimization along the search direction $\overset{\mathcal{R}}{\mathbf{p}}_k$, starting at the point $\overset{\mathcal{R}}{\mathbf{w}}_k$. In our experiments, we used the golden section search, which is guaranteed to have linear convergence, see [14].

Using (1), Eq. (3) becomes

$$\frac{1}{4}\mathbf{J}^H \overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \frac{1}{4}\mathbf{J}^H \overset{\mathcal{H}}{\mathbf{w}}_k + \alpha_k \frac{1}{4}\mathbf{J}^H \overset{\mathcal{H}}{\mathbf{p}}_k,$$

or, equivalently,

$$\overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \overset{\mathcal{H}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{H}}{\mathbf{p}}_k. \tag{4}$$

Now, the iteration for the next search direction is given by

$$\overset{\mathcal{R}}{\mathbf{p}}_{k+1} = -\overset{\mathcal{R}}{\mathbf{g}}_{k+1} + \beta_k \overset{\mathcal{R}}{\mathbf{p}}_k, \tag{5}$$

where $\overset{\mathcal{R}}{\mathbf{g}}_k := \nabla_{\overset{\mathcal{R}}{\mathbf{w}}_k} E$, and $\beta_k \in \mathbb{R}$ has different expressions, depending on the type of the conjugate gradient algorithm. For example, the *Hestenes-Stiefel* update expression (see [9]) for β_k is:

$$\beta_k = \frac{\overset{\mathcal{R}}{\mathbf{g}}_{k+1}^T (\overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k)}{\overset{\mathcal{R}}{\mathbf{p}}_k^T (\overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k)}. \tag{6}$$

From (1) and (2), we observe that (5) can be written as

$$\frac{1}{4}\mathbf{J}^H \overset{\mathcal{H}}{\mathbf{p}}_{k+1} = -\mathbf{J}^H \overset{\mathcal{H}}{\mathbf{g}}_{k+1} + \beta_k \frac{1}{4}\mathbf{J}^H \overset{\mathcal{H}}{\mathbf{p}}_k,$$

or, equivalently, as

$$\overset{\mathcal{H}}{\mathbf{p}}_{k+1} = -\overset{\mathcal{H}}{\mathbf{g}}_k + \beta_k \overset{\mathcal{H}}{\mathbf{p}}_k, \tag{7}$$

because the $\frac{1}{4}$ factor can be absorbed into $\overset{\mathcal{H}}{\mathbf{p}}_k$.

Taking into account the fact that

$$\overset{\mathcal{R}}{\mathbf{q}}_1^T \overset{\mathcal{R}}{\mathbf{q}}_2 = \overset{\mathcal{R}}{\mathbf{q}}_1^H \overset{\mathcal{R}}{\mathbf{q}}_2 = \left(\frac{1}{4}\mathbf{J}^H \overset{\mathcal{H}}{\mathbf{q}}_1\right)^H \frac{1}{4}\mathbf{J}^H \overset{\mathcal{H}}{\mathbf{q}}_2 = \frac{1}{16}\overset{\mathcal{H}}{\mathbf{q}}_1^H \mathbf{J}\mathbf{J}^H \overset{\mathcal{H}}{\mathbf{q}}_2 = \frac{1}{4}\overset{\mathcal{H}}{\mathbf{q}}_1^H \overset{\mathcal{H}}{\mathbf{q}}_2,$$

the update expression (6) can be written as

$$\beta_k = \frac{\overset{\mathcal{H}}{\mathbf{g}}{}_{k+1}^{H}(\overset{\mathcal{H}}{\mathbf{g}}_{k+1} - \overset{\mathcal{H}}{\mathbf{g}}_{k})}{\overset{\mathcal{H}}{\mathbf{p}}{}_{k}^{H}(\overset{\mathcal{H}}{\mathbf{g}}_{k+1} - \overset{\mathcal{H}}{\mathbf{g}}_{k})}. \tag{8}$$

Up until now we have worked with vectors from $\mathbb{H}^{4N}$. Ideally, we would like to work with vectors directly in $\mathbb{H}^N$. Considering the definition of $\overset{\mathcal{H}}{\mathbf{q}}$ for $\mathbf{q} \in \mathbb{H}^N$, this is done by taking the first N elements of the vector $\overset{\mathcal{H}}{\mathbf{q}}$. Thus, Eqs. (4) and (7) become, respectively,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{p}_k, \tag{9}$$

$$\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{p}_k, \tag{10}$$

where $\mathbf{w}_k$, $\mathbf{p}_k$, $\mathbf{g}_k := \nabla_{\mathbf{w}_k} E \in \mathbb{H}^N$. A simple calculation shows that

$$\frac{1}{4}\overset{\mathcal{H}}{\mathbf{q}}{}_1^{H}\overset{\mathcal{H}}{\mathbf{q}}_2 = \mathrm{Re}\left(\mathbf{q}_1^H \mathbf{q}_2\right),$$

where $\mathrm{Re}(q)$ represents the real part of the quaternion q, i.e. $\mathrm{Re}(q) = q_a$, if $q = q_a + iq_b + jq_c + kq_d \in \mathbb{H}$, and so (8) becomes

$$\beta_k = \frac{\mathrm{Re}\left(\mathbf{g}_{k+1}^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)}{\mathrm{Re}\left(\mathbf{p}_k^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)}. \tag{11}$$

Relations (9), (10), and (11) now define the quaternion-valued *Hestenes-Stiefel* algorithm.

Similar calculations give the quaternion-valued *Polak-Ribiere* update expression (see [16]):

$$\beta_k = \frac{\mathrm{Re}\left(\mathbf{g}_{k+1}^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)}{\mathbf{g}_k^H \mathbf{g}_k},$$

and the quaternion-valued *Fletcher-Reeves* update formula (see [19]):

$$\beta_k = \frac{\mathbf{g}_{k+1}^H \mathbf{g}_{k+1}}{\mathbf{g}_k^H \mathbf{g}_k}.$$

If the error function E is quadratic, then the conjugate gradient algorithm is guaranteed to find its minimum in at most N steps. But, in general, the error function may be far from quadratic, and so the algorithm will need more than N steps to approach the minimum. Over these steps, the conjugacy of the search directions tends to deteriorate, so it is a common practice to restart the algorithm with the negative gradient $\mathbf{p}_k = -\mathbf{g}_k$, after N steps.

A more sophisticated restart algorithm, proposed by Powell in [18], following an idea by Beale in [3], is to restart if there is little orthogonality left between the current gradient and the previous gradient. To test this, we verify that the following inequality holds:

$$\left|\mathrm{Re}\left(\mathbf{g}_k^H \mathbf{g}_{k+1}\right)\right| \geq 0.2\mathbf{g}_{k+1}^H \mathbf{g}_{k+1}.$$

The update rule (10) for the search direction $\mathbf{p}_k$ is also changed to

$$\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{p}_k + \gamma_k \mathbf{p}_t,$$

where

$$\beta_k = \frac{\mathrm{Re}\left(\mathbf{g}_{k+1}^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)}{\mathrm{Re}\left(\mathbf{p}_k^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)},\ \gamma_k = \frac{\mathrm{Re}\left(\mathbf{g}_{k+1}^H(\mathbf{g}_{t+1} - \mathbf{g}_t)\right)}{\mathrm{Re}\left(\mathbf{p}_t^H(\mathbf{g}_{t+1} - \mathbf{g}_t)\right)},$$

and $\mathbf{p}_t$ is an arbitrary downhill restarting direction. The conjugate gradient algorithm with these characteristics is called the *Powell-Beale* algorithm.

In order to apply the conjugate gradient algorithms to quaternion-valued feedforward neural networks, we only need to calculate the gradient $\mathbf{g}_k$ of the error function E at different steps, which we do by using the backpropagation algorithm.

4 Experimental Results

4.1 Linear Autoregressive Process with Circular Noise

An important application of quaternion-valued neural networks is at signal prediction. A known benchmark proposed in [15], and used for testing quaternion-valued neural networks in [8,22,25], is the prediction of the quaternion-valued circular white noise $n(k) = n_a(k) + i n_b(k) + j n_c(k) + k n_d(k)$, where $n_a(k)$, $n_b(k)$, $n_c(k)$, $n_d(k) \sim \mathcal{N}(0,1)$ passed through the stable autoregressive filter given by $y(k) = 1.79y(k-1) - 1.85y(k-2) + 1.27y(k-3) - 0.41y(k-4) + n(k)$.

We trained quaternion-valued feedforward neural networks using the general gradient descent algorithm (abbreviated GD), 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 conjugate gradient algorithm with Powell-Beale restarts (CGPB).

The tap input of the filter was 4, so the networks had 4 inputs, 4 hidden neurons on a single hidden layer, and one output. The activation function for the hidden layer was the fully quaternion hyperbolic tangent function, given by $G^2(q) = \tanh q = \frac{e^q - e^{-q}}{e^q + e^{-q}}$, and the activation function for the output layer was the identity $G^3(q) = q$. Training was done for 5000 epochs with 5000 training samples.

After running each algorithm 50 times, the averaged results are given in Table 1. 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.

In this case, CGHS and CGPR gave approximately the same results, with CGFR performing better and CGPB worse.

Table 1. Experimental results for linear autoregressive process with circular noise

Algorithm	Prediction gain
GD	4.51
CGHS	5.17
CGPR	5.19
CGFR	**6.91**
CGPB	5.00

4.2 3D Lorenz System

The Lorenz system is given by the ordinary differential equations

$$\begin{aligned}\frac{dx}{dt} &= \alpha(y-x)\\ \frac{dy}{dt} &= -xz+\rho x-y\\ \frac{dz}{dt} &= xy-\beta z,\end{aligned}$$

where $\alpha = 10$, $\rho = 28$ and $\beta = 2/3$. This represents a chaotic time series prediction problem, and was used to test quaternion-valued neural networks in [2,20,25].

The tap input of the filter was 4, and so the networks had 4 inputs, 4 hidden neurons, and one output neuron. The networks were trained for 5000 epochs with 1337 training samples.

The average results after 50 runs of each algorithm are given in Table 2. The measure of performance was the prediction gain, like in the above experiment.

In this experiment, CGHS and CGPB performed approximately in the same way, CGFR slightly better, and the best was CGPR.

Table 2. Experimental results for the 3D Lorenz system

Algorithm	Prediction gain
GD	7.56
CGHS	10.04
CGPR	**11.31**
CGFR	10.69
CGPB	10.12

4.3 4D Saito Chaotic Circuit

The last experiment involves the 4D Saito chaotic circuit given by

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dy_1}{dt} \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -\alpha_1 & \alpha_1\beta_1 \end{bmatrix} \begin{bmatrix} x_1 - \eta\rho_1 h(z) \\ y_1 - \eta\frac{\rho_1}{\beta_1} h(z) \end{bmatrix}$$
$$\begin{bmatrix} \frac{dx_2}{dt} \\ \frac{dy_2}{dt} \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -\alpha_2 & \alpha_2\beta_2 \end{bmatrix} \begin{bmatrix} x_2 - \eta\rho_2 h(z) \\ y_2 - \eta\frac{\rho_2}{\beta_2} h(z) \end{bmatrix},$$

where $h(z) = \begin{cases} 1, & z \geq -1 \\ -1, & z \leq 1 \end{cases}$, is the normalized hysteresis value, and $z = x_1 + x_2$, $\rho_1 = \frac{\beta_1}{1-\beta_1}$, $\rho_2 = \frac{\beta_2}{1-\beta_2}$. The values of the parameters are $(\alpha_1, \beta_1, \alpha_2, \beta_2, \eta) = (7.5, 0.16, 15, 0.097, 1.3)$. This is also a chaotic time series prediction problem, and was used as a benchmark for quaternion-valued neural networks in [1,2,7,8,21,22].

Like in the above experiments, the tap input of the filter was 4, and so the networks had the same architectures as the ones described earlier. We trained the networks for 5000 epochs with 5249 training samples.

The average prediction gains after 50 runs of each algorithm are given in Table 3.

In this last experiment, CGPR had the best performance, followed closely by CGPB, and lastly by CGFR and CGHS.

Table 3. Experimental results for the 4D Saito chaotic circuit

Algorithm	Prediction gain
GD	5.76
CGHS	11.59
CGPR	**13.64**
CGFR	12.08
CGPB	13.02

5 Conclusions

The deduction of the quaternion-valued conjugate gradient algorithm was presented, starting from the real-valued case and using the framework of $\mathbb{HR}$ calculus for the extension to the quaternion-valued case. The four variants of the conjugate gradient algorithm with different updates and restart procedures were applied for training networks used on three time series applications.

The experimental results shown that all the conjugate gradient algorithms performed better on the proposed problems than the classical gradient descent algorithm, in some cases as much as 8 dB better in terms of prediction gain.

Conjugate gradient with Polak-Ribiere and Fletcher-Reeves updates generally had the best performances, followed by conjugate gradient with Hestenes-Stiefel updates and conjugate gradient with Powell-Beale restarts.

As a conclusion, it can be said that conjugate gradient algorithms represent an efficient method of training feedforward quaternion-valued neural networks, as it was shown by their performance in solving different time series prediction problems.

References

1. Arena, P., Fortuna, L., Muscato, G., Xibilia, M.: Multilayer perceptrons to approximate quaternion valued functions. Neural Netw. **10**(2), 335–342 (1997)
2. Arena, P., Fortuna, L., Muscato, G., Xibilia, M.: Neural Networks in Multidimensional Domains Fundamentals and New Trends in Modelling and Control. Lecture Notes in Control and Information Sciences, vol. 234. Springer, London (1998)
3. Beale, E.: A derivation of conjugate gradients. In: Lootsma, F.A. (ed.) Numerical Methods for Nonlinear Optimization, pp. 39–43. Academic Press, London (1972)
4. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press Inc., New York (1995)
5. Buchholz, S., Le Bihan, N.: Polarized signal classification by complex and quaternionic multi-layer perceptrons. Int. J. Neural Syst. **18**(2), 75–85 (2008)
6. Charalambous, C.: Conjugate gradient algorithm for efficient training of artificial neural networks. IEE Proc. G Circuits Devices Syst. **139**(3), 301–310 (1992)
7. Ujang, C.B., Took, C., Mandic, D.: Split quaternion nonlinear adaptive filtering. Neural Netw. **23**(3), 426–434 (2010)
8. Ujang, C.B., Took, C., Mandic, D.: Quaternion-valued nonlinear adaptive filtering. IEEE Trans. Neural Netw. **22**(8), 1193–1206 (2011)
9. Hestenes, M., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**(6), 409–436 (1952)
10. Isokawa, T., Kusakabe, T., Matsui, N., Peper, F.: Quaternion neural network and its application. In: Palade, V., Howlett, R.J., Jain, L. (eds.) KES 2003. LNCS (LNAI), vol. 2774, pp. 318–324. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45226-3_44
11. Jahanchahi, C., Took, C., Mandic, D.: On HR calculus, quaternion valued stochastic gradient, and adaptive three dimensional wind forecasting. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–5. IEEE, July 2010
12. 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)
13. Kusamichi, H., Isokawa, T., Matsui, N., Ogawa, Y., Maeda, K.: A new scheme for color night vision by quaternion neural network. In: International Conference on Autonomous Robots and Agents, pp. 101–106, December 2004
14. Luenberger, D., Ye, Y.: Linear and Nonlinear Programming. International Series in Operations Research & Management Science, vol. 116. Springer, Heidelberg (2008)
15. Mandic, D., Chambers, J.: Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability. Wiley, New York (2001)
16. Polak, E., Ribiere, G.: Note sur la convergence de méthodes de directions conjuguées. Revue Française d'Informatique et de Recherche Opérationnelle **3**(16), 35–43 (1969)

17. Popa, C.-A.: Conjugate gradient algorithms for complex-valued neural networks. In: Arik, S., Huang, T., Lai, W.K., Liu, Q. (eds.) ICONIP 2015. LNCS, vol. 9490, pp. 412–422. Springer, Cham (2015). doi:10.1007/978-3-319-26535-3_47
18. Powell, M.: Restart procedures for the conjugate gradient method. Math. Program. **12**(1), 241–254 (1977)
19. Reeves, C., Fletcher, R.: Function minimization by conjugate gradients. Comput. J. **7**(2), 149–154 (1964)
20. Took, C., Mandic, D.: The quaternion lms algorithm for adaptive filtering of hypercomplex processes. IEEE Trans. Sig. Process. **57**(4), 1316–1327 (2009)
21. Took, C., Mandic, D.: Quaternion-valued stochastic gradient-based adaptive IIR filtering. IEEE Trans. Sig. Process. **58**(7), 3895–3901 (2010)
22. Took, C., Mandic, D.: A quaternion widely linear adaptive filter. IEEE Trans. Sig. Process. **58**(8), 4427–4431 (2010)
23. Took, C., Mandic, D., Aihara, K.: Quaternion-valued short term forecasting of wind profile. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–6. IEEE, July 2010
24. Took, C., Strbac, G., Aihara, K., Mandic, D.: Quaternion-valued short-term joint forecasting of three-dimensional wind and atmospheric parameters. Renewable Energy **36**(6), 1754–1760 (2011)
25. Xia, Y., Jahanchahi, C., Mandic, D.: Quaternion-valued echo state networks. IEEE Trans. Neural Netw. Learn. Syst. **26**(4), 663–673 (2015)
26. Xu, D., Xia, Y., Mandic, D.: Optimization in quaternion dynamic systems: gradient, Hessian, and learning algorithms. IEEE Trans. Neural Netw. Learn. Syst. **27**(2), 249–261 (2016)

Evaluating Suitable Job Applicants Using Expert System

Bogdan Walek(✉), Ondrej Pektor, and Radim Farana

Department of Informatics and Computers, University of Ostrava, 30. Dubna 22, 701 03 Ostrava, Czech Republic
{bogdan.walek, radim.farana}@osu.cz, ondrej@pektor.cz

Abstract. This paper proposes a fuzzy system for selection of suitable job applicants using an expert system. We propose the evaluation of job applicants using evaluation number and the evaluation of a job interview by a prepared questionnaire and expert system – EXS1. Based on this information and knowledge base, the expert system (EXS2) suggests the most suitable candidates for the position. The proposed fuzzy system is verified on a selected job position and a few job applicants.

Keywords: Fuzzy system · Job applicants · Job position · Job interview · Expert system · Fuzzy · Evaluation number · Questionnaire

1 Introduction

Every company sometimes recruits new employees. Hiring of new staff and selecting suitable job applicants is often a very complicated process, because there are a lot of criteria which enter this process. There are also different hard and soft skills of each job applicant [1].

Currently, selecting suitable job applicants is usually solved by a Human Resources (HR) department. If there is no HR department or HR manager in a company, this task is solved by the director or the owner of the company. But the selection process is the same or very similar. This paper focuses mainly on medium and big companies where the HR manager is responsible for the hiring process [1].

Currently, there are several approaches related to personnel selection [10, 11].

This paper will propose a fuzzy system for selection of suitable job applicants using an expert system. The paper is continuation of paper [1–3].

2 Problem Formulation

As said above, hiring of new staff is a complicated process and an HR manager has a difficult task to select the most suitable job applicant. In medium and big companies, information about job applicants is stored in a database. Such a database typically contains CV of job applicants and their hard skills [1].

One of difficult tasks is how to store soft skills and subsequently process them appropriately. It is difficult to measure the level of each soft skill of a job applicant and compare it with soft skill levels of other job applicants. During the hiring process, the HR manager can recognize strong or weak soft skills but it might be difficult to describe and store all information about job applicant's soft skills in a database, including subjective impressions and evaluation of the HR manager [1].

Another problem is that there are many characteristics and properties of candidates for a specific job position. Moreover, each characteristic may affect the decision-making process during selecting the most suitable job applicant.

Another problem is that the HR manager´s subjective evaluation of each job applicant, and a specific candidate can be prioritised because of his impression on the HR manager. Thus, it is not his hard and soft skills which are more important than job applicant´s impression on the manager. The HR manager also can make mistakes, for instance: forgetting important characteristics during a multi-round selection process, time pressure, misunderstanding of the importance of the requirements for the job position, etc. [1].

Currently, there are open source systems for human resource management, such as Opencats or OrangeHRM. They are mainly focused on supporting the process of hiring new employees, but not on evaluating the most suitable employees or on decision-support of selecting the most suitable candidates for the job [1].

3 Problem Solution

Based on the above-mentioned reasons, we propose a fuzzy system with an expert system for evaluating and proposing the most suitable candidates for a given position.

Main parts of proposed fuzzy system are visually shown in the Fig. 1.

3.1 Job Position Modelling

In the first step, it is essential to define requirements for the job position. This part is typically defined by an expert (project manager, team leader, head of division, etc.). Usually, the job position consists of two major categories of requirements – hard skills and soft skills.

For job position modelling, we propose the following universal structure for all requirements [1]:

- Category (possible values: mandatory-M, desired-D, is an advantage-A)
- Type (possible values: hard skill-H, soft skill-S)
- Quantification/Level
- Skill (expertise) description (name)
- Detailed description – optional
- Final description for requirement for job advert – consists of quantification, skill description and optionally detailed description, final description should be slightly modified for publishing in a job advert

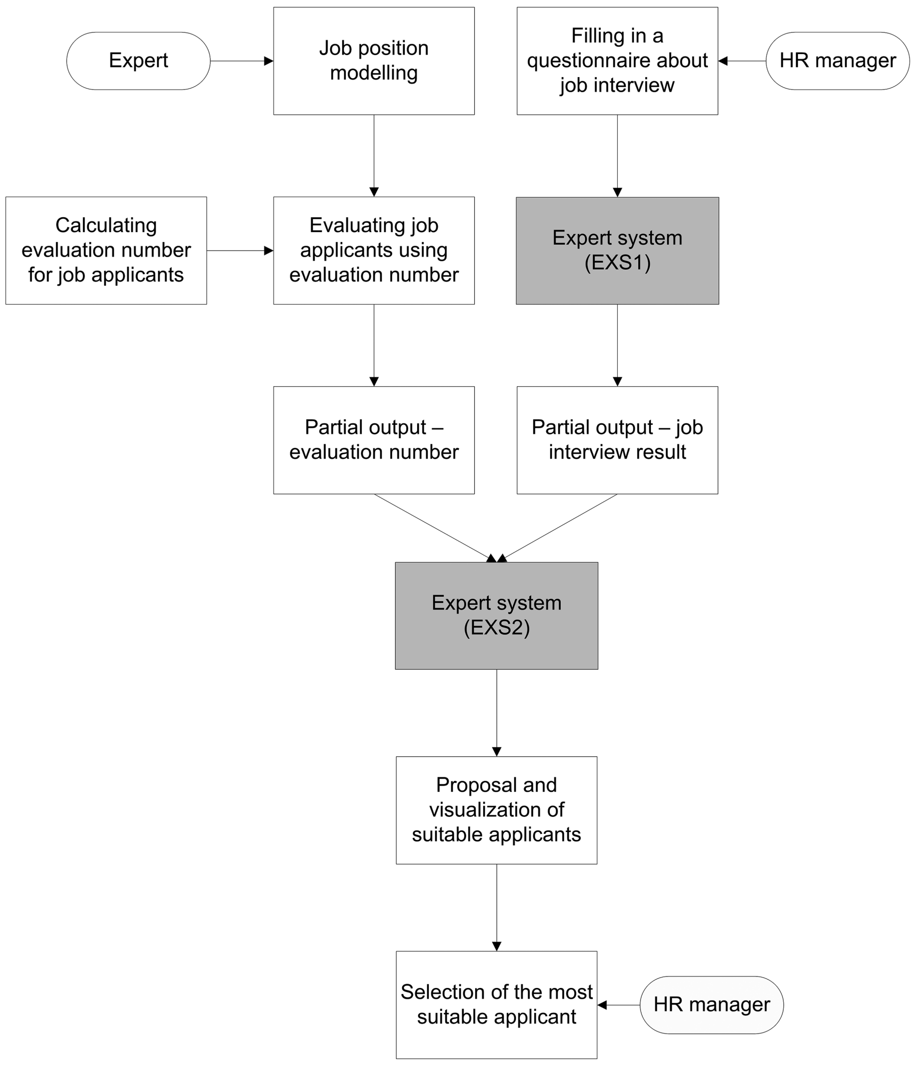

Expert
Job position modelling
Filling in a questionnaire about job interview
HR manager
Calculating evaluation number for job applicants
Evaluating job applicants using evaluation number
Expert system (EXS1)
Partial output – evaluation number
Partial output – job interview result
Expert system (EXS2)
Proposal and visualization of suitable applicants
Selection of the most suitable applicant
HR manager

Fig. 1. Proposed fuzzy system

Determining scales and possible values

For each job position requirement (criterion) which is used to evaluate the candidates the scale of vague terms which can be used for this criterion must be defined. Within the final web application of the proposed system, there will be available general pre-defined scales which should be connected to user-defined scales.

Each scale includes the word “Unknown”, which can be set as default, and the value is 0. Other vague expressions have higher values, depending on how the expression is positive. These values can also be used for the calculation.

The only exception is the criterion “practice length”, which will be evaluated using numbers which indicate the real length of practice, for example 2 (years), 3 (years), 4 (years), etc.

Determining importance of criteria
For each criterion (job position requirement), the importance of criterion can be defined. The importance of criterion determines how important the requirement is for a specific job position. For simplicity, from the user’s perspective the importance of criterion is determined using scale one to five points. These points will have values from 1/5 to 5/5 in calculating of the evaluation number.

Importance can be determined depending on the place occupied by expert group using the methods of multi-criteria analysis like Fuller’s triangle or Saaty’s method [5–7] or using fuzzy approach [8, 9].

3.2 Evaluating Job Applicants Using Evaluation Number

In this step, information about job applicants is loaded. The HR manager has a CV (curriculum vitae) and other documents for each job applicant. We propose storing information about job applicants in a database of applicants.

A profile of each entry in the job applicant database consists of [1]:

- Personal information
- Relevant hard skills
- Relevant soft skills
- Other skills – for instance, a job applicant seeks for a job position Java programmer and he puts into his CV - skill “driver license for buses”. This is an interesting skill but irrelevant to this job position. Though a hard skill, the HR manager stores it in category Other skills

Assigning relevant information
For each job position requirement, the HR manager assigns relevant information from the job applicant profile. If the HR manager is not able to find relevant information for the job position requirement, the value of the job position requirement will be “Unknown”.

Calculation base
The first step in the calculation is to determine a base. The base is determined as a fictitious job applicant with whom other job applicants are compared. To determine the base, two strategies were tested. The first strategy is to determine the base as the average of all job applicants. This approach evaluates the candidates independently of other competitions and a comparison with other competition is impossible in this case. The calculation is defined below:

$$BaseCriterion_i = \sum\nolimits_{j=1}^{n} \frac{JobApplicantRequirement_j}{n} \quad (1)$$

The second strategy is to determine a base as the best possible candidate. This is done by assigning the greatest possible value of the corresponding scale for each criterion. If the scales are maintained for several competitions for the same position, the evaluation of the candidates can be compared to each other. This strategy seems to be more appropriate.

Calculation of the evaluation for one criterion

After determining the base candidate, the calculation for evaluation of individual criteria for each candidate is performed. This calculation is carried out by dividing the value assigned by HR manager to the corresponding value of the base candidate, all multiplied by the importance of the criterion.

$$CriterionEvaluation_i JobApplicant_j = \frac{CriterionValue_i JobApplicant_j}{BaseCriterion_i} * \frac{CriterionImportance_i}{ImportancePointsCount} \tag{2}$$

Evaluating suitability of job applicant based on criteria

The final step of this evaluation is to calculate the overall suitability of each candidate. The result is the sum of all evaluation criteria for the specific candidate.

$$JobApplicantEvaluation_j = \sum\nolimits_{j=1}^{n} CriterionEvaluation_i JobApplicant_j \tag{3}$$

The evaluation (validation) number value depends on the criteria count. Normalisation is possible using the base (maximum) or manually set the reference base (evaluation by current employer).

3.3 Filling in a Questionnaire About Job Interview

In this step, the HR manager fills in the prepared questionnaire about the job interview. The questionnaire consists of several categories of questions. Each category consists of several questions and the result of each category is an input linguistic variable for the first expert system which evaluates the job applicant suitability based on the job interview (EXS1). The knowledge base of the expert system consists of IF-THEN rules.

Table 1. Structure of proposed questionnaire

Category	Question	Answer
C1	Q1-1	A1-1-1
C1	Q1-1	A1-1-2
C1	Q1-1	A1-1-3
C1	Q1-2	A1-2-1
C1	Q1-2	A1-2-2
C1	Q1-2	A1-2-3
C2	Q2-1	A2-1-1
C2	Q2-1	A2-1-2
C2	Q2-1	A2-1-3

The structure of the proposed questionnaire is shown in Table 1:
The structure of expert system EXS1 is shown below:

Input linguistic variables:

- R1 – result of C1 category
- R2 – result of C2 category
- R3 – result of C3 category
- R4 – result of C4 category

Output linguistic variable:

- Job interview evaluation

3.4 Defining Expert System for Proposing Suitable Job Applicants

In this step, the expert system for proposing suitable job applicants based on previous partial outputs is defined (EXS2). The knowledge base of the expert system consists of IF-THEN rules.

Input linguistic variables are:

- VALIDATION – evaluation (validation) number – output of evaluating job applicant suitability based on his skills, defined in Sect. 3.2, possible values: very low, low, medium, high, very high
- INTERVIEW – job interview evaluation – output of evaluating job applicant after job interview, output of EXS1, possible values: very low, low, medium, high, very high
- WEIGHT – weight of inputs, determines which partial result is more important for final job applicant evaluation. HR manager can determine that validation number is more or less important for final job applicant evaluation, possible values (validation – validation number is more important, medium – validation number and job interview evaluation has the same weight, interview – job interview evaluation is more important)

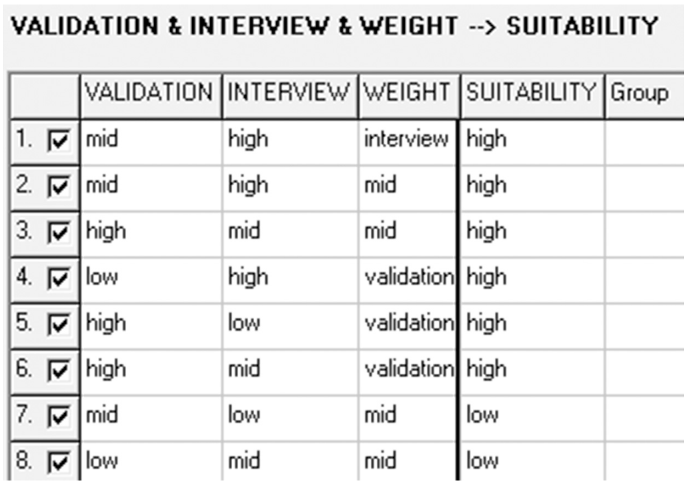
Output linguistic variable is:

- SUITABILITY – level of job applicant suitability, determines the level of job applicant suitability, possible values: very high, high, medium, low, very low

The creation of the expert system knowledge base was performed in the LFL Controller. Linguistic Fuzzy Logic Controller is more described in [4]. Part of the knowledge base of expert system EXS2 is shown in Fig. 2.

3.5 Proposal and Visualization of Suitable Applicants

In this step, all job applicants are evaluated by EXS2 and for each job applicant the level of suitability is assigned. The job applicants are ordered from the most suitable job applicant and visualised for the HR manager.



VALIDATION & INTERVIEW & WEIGHT --> SUITABILITY

	VALIDATION	INTERVIEW	WEIGHT	SUITABILITY	Group
1. ☑	mid	high	interview	high	
2. ☑	mid	high	mid	high	
3. ☑	high	mid	mid	high	
4. ☑	low	high	validation	high	
5. ☑	high	low	validation	high	
6. ☑	high	mid	validation	high	
7. ☑	mid	low	mid	low	
8. ☑	low	mid	mid	low	

Fig. 2. Part of knowledge base of EXS2

3.6 Selection of the Most Suitable Applicant

In the last step, the HR manager selects the most suitable candidate for the particular position. The expert system, evaluation numbers of job applicants and questionnaire results are only helpful tools for the human resources manager who will make the final selection of the most suitable candidate.

4 Verification

In cooperation with company Raynet s.r.o (company that develops CRM systems), we prepared a verification of the proposed fuzzy system for job positions of this company. The verification will be performed on a selected job position.

4.1 Job Position Modelling

The selected job position is called SW developer. The requirements (criteria) for this job position are shown in Table 2.

4.2 Evaluating Job Applicants Using Evaluation Number

In this step, three experimental job applicants are evaluated. Their suitability is determined by the evaluation (validation) number. Evaluation of job applicants using the evaluation number is shown in Table 3.

Table 2. Requirements for job position SW developer

SW developer		
Hard skills	**Scale**	**Importance**
HTML5	Scale 1	4/5
CSS	Scale 1	4/5
JavaScript	Scale 1	4/5
Java	Scale 1	3/5
Spring	Scale 1	2/5
SQL	Scale 1	4/5
Postgre	Scale 1	3/5
Mongo	Scale 1	2/5
Scala	Scale 1	1/5
Vagrant	Scale 1	1/5
Soft skills	**Scale**	**Importance**
Ability to personal growth	Scale 2	4/5
Attention to detail	Scale 2	3/5
Communication	Scale 2	2/5
Raynet thinking	Scale 2	5/5
Practice	**Scale**	**Importance**
3 years HTML	Years	3
3 years CSS	Years	3
2 years Java	Years	2
1 year Spring	Years	1

Scale 1		**Scale 2**	
Expression	**Value**	**Expression**	**Value**
Nothing	0	Unknown	0
Beginner	1	Weak	1
Apprentice	2	OK	2
Wizard	3	Perfect	3
Guru	4		

Table 3. Evaluating job applicants using evaluation number

Applicant 1			Applicant 2		
Expr.	Value	Eval.	Expr.	Value	Eval.
Wizard	3	0.6	Guru	4	0.8
Wizard	3	0.6	Guru	4	0.8
Beginner	1	0.2	Guru	4	08
Wizard	3	0.45	Apprentice	2	0.3
Nothing	0	0	Beginner	1	0.1
Wizard	3	0.6	Wizard	3	0.6
Nothing	0	0	Beginner	1	0.15
Nothing	0	0	Nothing	0	0
Nothing	0	0	Beginner	1	0.05
Nothing	0	0	Nothing	0	0
Perfect	3	0.8	OK	2	0.53
Perfect	3	0.6	OK	2	0.4
Perfect	3	0.4	Perfect	3	0.4
Perfect	3	1	Weak	1	0.33
3	3	0.6	6	6	1.2
5	5	1	6	6	1.2
3	3	0.6	2	2	0.4
0	0	0	1	1	0.2
Evaluation number		**7.45**	**Evaluation number**		**8.26**

Applicant 3			Maximum - base	
Expr.	Value	Eval.	Value	Eval.
Apprentice	2	0.4	4	0.8
Beginner	1	0.2	4	0.8
Apprentice	2	0.4	4	0.8
Nothing	0	0	4	0.6
Nothing	0	0	4	0.4
Apprentice	2	0.4	4	0.8
Nothing	0	0	4	0.6
Nothing	0	0	4	0.4
Nothing	0	0	4	0.2
Beginner	1	0.05	4	0.2
Perfect	3	0.4	3	0.8
Perfect	3	0.4	3	0.6
Weak	1	0.13	3	0.4
Unknown	0	0	3	1
–1	2	0.4	3	0.6
–2	1	0.2	3	0.6
+1	3	0.6	2	0.4
–1	0	0	1	0.2
Evaluation number		**3.58**	**Eval. number**	**10**

4.3 Filling Questionnaire About Job Interview

The HR manager fills in the questionnaire after the job interview of all applicants. The questionnaire consists of these four categories of questions:

- C1 – hard skills
- C2 – soft skills
- C3 – practice
- C4 – special task

Evaluation for each category and complete evaluation for the job interview is shown in Table 4.

Table 4. Evaluation for the job interview

Applicant	C1	C2	C3	C4	Eval.
1	Low	Very high	Medium	Medium	Medium
2	High	Medium	Very high	Very high	High
3	Low	Medium	Low	Low	Low

Applicant 1 is evaluated as second, because soft skills are very good, but hard skills are insufficient.

Applicant 2 is evaluated as first (the best) because hard skills are good and practice length is very high.

Applicant 3 is evaluated as third, because hard skills are very insufficient, practice length is very low and soft skills are average.

4.4 Proposal and Visualization of Suitable Applicants

In this step, applicants are evaluated by EXS2 and visualised for the HR manager. Weight of inputs was determined as interview, so the job interview results are more important than the results from evaluating by evaluation (validation number). The results are shown in Table 5.

Table 5. Evaluating suitable job applicants

Applicant	VALIDATION	INTERVIEW	WEIGHT	SUITABILITY
2	Very high	High	Interview	Very high
1	Very high	Medium	Interview	High
3	Low	Low	Interview	Low

4.5 Selection of the Most Suitable Applicant

In the last step, the HR manager selects the most suitable candidate for the position called SW developer.

5 Conclusion

This article proposed a fuzzy system using an expert system for the selection of the most appropriate applicant for a job position within the company. The article described the evaluation of job applicants using an evaluation number and the evaluation of job interview by a prepared questionnaire and expert system – EXS1. Based on this information and knowledge base, the expert system (EXS2) suggests the most suitable candidates for the job position.

Acknowledgment. This work was supported by the project "LQ1602 IT4Innovations excellence in science" and during the completion of a Student Grant SGS02/UVAFM/2016 with student participation, supported by the Czech Ministry of Education, Youth and Sports.

References

1. Walek, B., Pektor, O., Farana, R.: Proposal of the web application for selection of suitable job applicants using expert system. In: Silhavy, R., Senkerik, R., Oplatkova, Z.K., Silhavy, P., Prokopova, Z. (eds.) Software Engineering Perspectives and Application in Intelligent Systems. AISC, vol. 465, pp. 363–373. Springer, Cham (2016). doi:10.1007/978-3-319-33622-0_33
2. Walek, B., Bartoš, J.: Expert system for selection of suitable job applicants. In: Proceedings of the 11th International FLINS Conference, pp. 68–73. World Scientific Publishing Co. Pte. Ltd., Singapore (2014)
3. Walek, B.: Fuzzy tool for selection of suitable job applicants. Glob. J. Technol. **2015**(8), 125–132 (2015)
4. Habiballa, H., Novák, V., Dvořák, A., Pavliska, V.: Using software package LFLC 2000. In: 2nd International Conference Aplimat 2003, Bratislava, pp. 355–358 (2003)
5. Triantaphyllou, E.: Multi-criteria Decision Making Methods: A Comparative Study, vol. 28, p. 288. Kluwer Academic Publishers, Boston (2000). ISBN: 0-7923-6607-7
6. Saaty, T.L.: Multi-decisions decision-making: in addition to wheeling and dealing, our national political bodies need a formal approach for prioritization. Math. Comput. Model. **46** (7–8), 1001–1016 (2007). ISSN: 0895-7177
7. Köksalan, M., Wallenius, J., Zionts, S.: Multiple Criteria Decision Making: From Early History to the 21st Century, vol. 11, p. 197. World Scientific, Hackensack (2011)
8. Ramík, J.: Pairwise comparison matrix with fuzzy elements. In: 32nd International Conference on Mathematical Methods in Economics (MME), Olomouc, 10–12 September 2014, pp. 849–854 (2014). ISBN: 978-80-244-4209-9
9. Ramík, J.: Isomorphisms between fuzzy pairwise comparison matrices. Fuzzy Optim. Decis. Making. **14**(2), 199–209 (2015). ISSN: 1568-4539
10. Saidi-Mehrabad, M., Fattahi, P.: Flexible job shop scheduling with tabu search algorithms. Int. J. Adv. Manuf. Technol. **32**(5–6), 563–570 (2007)
11. Gungor, Z., Serhatlıoğlu, G., Kesen, S.E.: A fuzzy AHP approach to personnel selection problem. Appl. Soft Comput. **9**(2), 641–646 (2009)

A Computationally Efficient Approach for Mining Similar Temporal Patterns

Vangipuram Radhakrishna[1(✉)], P.V. Kumar[2], and V. Janaki[3]

[1] VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India
radhakrishna_v@vnrvjiet.in

[2] University College of Engineering, Osmania University, Hyderabad, India
pvkumar58@gmail.com

[3] Vaagdevi Engineering College, Warangal, India
janakicse@yahoo.com

Abstract. Temporal association patterns are those patterns which are obtained from time stamped temporal databases. A temporal association pattern is said to be similar if it satisfies specified subset constraints. The apriori algorithm which is designed for static databases cannot be extended to find similar temporal patterns from temporal databases as patterns are vectors with supports computed at different time slots and Euclidean distance do not satisfy monotonicity property. The brute force approach to find similar temporal patterns requires computing 2^n true support combinations for 'n' items from finite item set and problem falls in NP-class. In this present research, we come up with novel approach to discover temporal association which are similar for pre-specified subset constraints, and substantially reduce support computations. The proposed approach eliminates computational overhead in finding similar temporal patterns. The results prove that the proposed method outperforms brute force approach.

Keywords: Support bounds · Subset specification · Temporal pattern · Prevalance

1 Introduction

The concept of finding frequent item sets is extensively studied in literature in reference to static or non-temporal databases [8,10]. Researchers have also come up with several interesting algorithms to find frequent item sets aiming at time and space efficiency. Temporal frequent item sets have also been studied, but in a different perspective. The past studies did not consider finding similar temporal patterns whose prevalence (support) values are similar satisfying subset specifications except in research contribution of authors [3,5–7,9]. The subset specification involves specifying the distance measure, reference support vector, user interest threshold value [3]. In this research, we come up with the novel approach for discovering temporal patterns whose prevalence values varies similar to the reference sequence. Our approach reduces the computational overhead in finding the support values of item sets. We now define some basic terminology followed in the design of formal expressions and algorithm procedure.

1.1 Basic Terminology

Temporal Pattern. A Temporal pattern is sequence of support values computed at every time slot defined over a finite time period, $T_p = \{t_1, t_2, ...t_n\}$ where $t_1, t_2....t_n$ are time slots.

Positive Prevalence and Positive Prevalence Sequence (PPS). Positive prevalence is support value of item set at a given time slot while positive prevalence sequence is sequence of positive support values computed at every time slot defined over finite time period. A positive temporal item set is denoted by T_n.

Negative Prevalence and Prevalence Sequence (NPS). Negative prevalence is non-existence support value of item set at a given time slot while its prevalence sequence is sequence of negative support values computed at every time slot defined over finite time period. A negative temporal itemset is denoted by using a complement notation as $\overline{T}_n$.

Reference Sequence and Threshold. It is prevalence support sequence chosen randomly and is of user interest. Threshold is the dissimilarity constraint which indicates the distance between reference pattern and temporal item set.

Pattern or Item Set Pruning. The item set is killed or pruned when maximum possible support sequence exceeds dissimilarity constraint.

Maximum Pattern Bound Support Sequence (HPBSS) and Maximum Possible Minimum Dissimilarity (D^U). It is the support sequence denoting maximum possible support value at every time slot. The distance computed of R w.r.t HPBSS is termed maximum possible minimum dissimilarity.

Minimum Pattern Bound Support Sequence (LPBSS) and Minimum Possible Minimum Dissimilarity (D^L). It is the support sequence denoting minimum possible support value at every time slot. The distance computed of R w.r.t LPBSS is termed minimum possible minimum dissimilarity.

Minimum Possible Support Bound (D^{min}). It is the sum of distances (D^U) and (D^L).

2 Related Works

An approach for finding temporal frequent patterns is discussed in [1] which uses the concept of TFP-tree. In [2], authors discuss the importance of temporal frequent patterns with reference to medical scenario and time series applications. In [3], the temporal patterns are extracted from the input database through using the bounding concepts of supports of item sets. They adopt the work of Calder's [11] which estimates bounds of supports. In [5,6] temporally similar natured patterns are retrieved for a set of predefined subset specifications. The subset constraints involve specifying reference support vector, user interest threshold value and appropriate distance metric. The objective of their approach is that, they essentially require computations of true support sequences of all item sets at previous level when deciding if a pattern at next level is similar or not. In this research, we design formal expressions to efficiently estimate prevalence values of temporal item sets. In [4], the patterns are mined from underlying time interval based data by using concept of Gaussian function.

3 Problem Definition

Similar Temporal Patterns are those patterns whose item set prevalence variations are similar as that of a reference sequence interest to the user and satisfying certain dissimilarity constraint. To the best of our survey and knowledge there is no contribution of algorithms or methods in this direction except the work initiated by the authors in 2009 [3,5,6]. In their work, authors propose an approach to find similar temporal item sets but they compute the true supports of all item sets of previous stage when estimating supports of superset item sets. In this paper, we overcome this dis-advantage of computing true support values of all item sets by defining formal expressions to estimate bound of supports. These expressions help us to estimate supports without the need to know true supports of all subsets when support for superset pattern is being estimated.

Consider the brute force approach to find temporal patterns that are similar with respect to a specified reference sequence, denoted by R, which is chosen by the user for a given threshold value, denoted by Δ. For item set with 'N' items, this approach requires generating $(2^N - 1)$ item set combinations and then computing true support values for all these $2^N - 1$ item set combinations. The complexity is thus O (2^N). This means that the complexity is exponential which indicates that the complexity class of this approach is NP-class. The present research is motivated from this fact. The objective of this research is thus to reduce the computational overhead in computing true supports by estimating support bounds of item sets. The true support for item set is computed if and only if it is really needed and required.

3.1 Design Expressions to Estimate Support Bounds

Let N be the number of items in the Finite set of items, denoted by F with size of item set denoted by L. We design the expressions to estimate bounds of supports of temporal patterns by considering two cases. These include considering all those combinations of Temporal patterns of size, $|L| = 2$ and Temporal patterns of size, $|L| > 2$.

Case-1: $|\boldsymbol{L}| = \mathbf{2}$ Let T_m and T_n be two any temporal singleton patterns in Level-1, such that

$$\begin{aligned} T_m &= (T_{m_1}; T_{m_2}; T_{m_3}; \quad .....T_{m_n}) \text{ and} \\ T_n &= (T_{n_1}; T_{n_2}; T_{n_3}; \quad .....T_{n_n}) \end{aligned}$$

where each T_{m_i} and T_{n_i} is support value at i^{th} timeslot. Then the support bounds of temporal pattern of size, $|L| = 2$, is obtained using Eq. 1

$$T_m T_n = T_m - T_m * \overline{T}_n \tag{1}$$

where $T_m * \overline{T}_n$ is defined recursively as

$$T_m * \overline{T}_n = \begin{cases} minimum(T_m, \overline{T}_n); & \forall\ time\ slots \\ maximum(T_m + \overline{T}_n - 1, 0); & \forall\ time\ slots \end{cases} \tag{2}$$

and T_m and $\overline{T}_n$ is positive and negative prevalence sequence respectively.

Case-2: $|\boldsymbol{L}| > \mathbf{2}$ Let $T_m T_n$ be the temporal pattern of size, L, for which we must estimate support bounds. We divide all such patterns into two sub-patterns. The first sub pattern (T_m) is assumed to be of length, L-1 and second sub pattern (T_n) is equal to length, L = 1. To estimate support bounds of temporal pattern of the form $T_m T_n$ whose size is $|L| > 2$, we assume that T_m is temporal pattern of length (n−1) and T_n is singleton temporal pattern of length = 1. Let $T_m T_n$ be any two given temporal patterns, such that

$$\begin{aligned} T_m &= (T_{m_1}; T_{m_2}; T_{m_3} \quad ....T_{m_n}) \\ \text{and}\ \ T_n &= (T_{n_1};\ T_{n_2};\ T_{n_3} \quad .....T_{n_n}) \end{aligned}$$

where each T_{m_i} and T_{n_i} is support value at i^{th} timeslot. Then the support bounds of temporal pattern of size, $|L| > 2$ is obtained using Eq. 3

$$T_m T_n = \begin{cases} T_{m_{Max}} - (T_{m_{Max}} * \overline{T}_n) \\ T_{m_{Min}} - (T_{m_{Min}} * \overline{T}_n) \end{cases} \tag{3}$$

Where

T_m is positive temporal pattern of size, (L−1), $\overline{T}_n$ is singleton temporal pattern of length = 1 and $(T_{m_{Max}} * \overline{T}_n)$ and $(T_{m_{Min}} * \overline{T}_n)$ are defined as given by Eq. 2.

3.2 Proposed Similar Temporal Association Pattern Mining Algorithm

In this section, we outline the proposed methodology to find the similar temporal patterns

Algorithm: Mining Similar Temporal Patterns

Input: Temporal Database with time stamped transactions; Reference sequence; User specified dissimilarity threshold and distance function.

Output: Similar temporal patterns

Step-1: Obtain Positive Prevalence and Negative Prevalence for each singleton temporal item (temporal pattern) from the finite set of items defined by F. This is the first time; we compute true support values of all temporal items initially.

Step-2: Obtain Positive Prevalence Sequence and Negative Prevalence Sequence for each singleton temporal item (temporal pattern) using prevalence values computed in step-1.

Step-3: // Pruning stratergy-1 for singleton patterns of size, L = 1
Compute true distance between reference and each temporal pattern of size = 1. If the true distance satisfies dissimilarity constraint, the pattern is treated as similar and also retained. If the true distance exceeds dissimilarity constraint, the pattern is treated dissimilar. To decide to prune or not, we compute distance (D^U) which is the maximum possible deviation above reference sequence defined in Sect. 3.3. If this distance exceeds threshold value, then prune the temporal pattern. In this case, we say the temporal pattern is not retained. Otherwise, i.e. if dissimilarity condition is satisfied, then we retain the candidate temporal pattern to compute prevalence values (support values) of higher size temporal patterns.

Step-4: // Pruning stratergy-2 for temporal patterns of size, $|L| > 1$

Case-1: Obtain the Maximum Pattern Bound Support Sequence and Minimum Possible Support Bound for all patterns of $size > 1$. Now, if the Minimum Possible Support Bound (D^{Min}) exceeds threshold, the pattern is dissimilar. To decide whether this temporal pattern be pruned or not, we compute distance (D^U) which is the maximum possible deviation above reference sequence. If $D^U > threshold, \Delta$, then prune the temporal pattern. In this case, we say the temporal pattern is not retained. However, if dissimilarity condition is satisfied, then we retain the candidate temporal pattern to compute prevalence values (support values) of higher size temporal patterns.

Case-2: Obtain Minimum Possible Support Bound (D^{min}) for all patterns of $size > 1$. Now, if the Minimum Possible Support Bound (D^{min}) satisfies threshold, the pattern may be similar but cannot be confirmed always, as it depends on item set distribution. This is because we are estimating maximum and minimum possible bounds and comparing these with reference. So, to decide whether this temporal pattern is temporally similar, we scan the database to find patterns true support sequence and from this we find its true distance with reference, R. If the $truedistance \leq threshold$, then temporal pattern is similar and is also retained. If the true distance is violating dissimilarity condition then we find If $D^U > threshold$, Δ, if so we prune the temporal pattern. In this case, we say the temporal pattern is not retained. However, if $D^U \leq \Delta$, then we retain the candidate temporal pattern to compute prevalence values (support values) of higher size temporal patterns.

Step-5: Prune all other candidate temporal patterns based on D^U. In the process of verifying temporal patterns for similarity w.r.t reference, discard and prune all temporal superset patterns, if there is at least one subset pattern of this superset pattern which is not retained. i.e. for temporal pattern T, to be considered similar all its sub-patterns must satisfy the retaining condition. If there is at least one subset such that it does not satisfy dissimilarity condition, then the superset pattern, T, is not similar.

Step-6: Output all candidate temporal patterns which are similar and retained.

3.3 Computing Support Bounds

To estimate distance bounds, we use the following equations as defined in research contributions of authors [3,5,6].

Maximum Possible Minimum Dissimilarity, (D^U). Let, $T_m = (T_{m_1}; T_{m_2}; T_{m_3} .... T_{m_n})$ and $R_s = (R_{s_1}; R_{s_2}; R_{s_3} .... R_{s_n})$ be the Maximum Pattern Bound Support Sequence and Reference Sequence respectively. Now, D^U is defined using Eq. 4.

$$D^U = \sqrt{(R_{s_1} - T_{m_1})^2 + (R_{s_2} - T_{m_2})^2 + ... + (R_{s_n} - T_{m_n})^2} \tag{4}$$

where each $R_{s_i} > T_{m_i} \forall \;\; i$

Minimum Possible Minimum Dissimilarity, (D^L). Let, $T_m = (T_{m_1}; T_{m_2}; T_{m_3} ... T_{m_n})$ and $R_s = (R_{s_1}; R_{s_2}; R_{s_3} ... R_{s_n})$ be the Minimum Pattern Bound Support Sequence and Reference Sequence respectively. Now, D^L is defined using Eq. 5.

$$D^L = \sqrt{(T_{m_1} - R_{s_1})^2 + (T_{m_2} - R_{s_2})^2 + ... + (T_{m_n} - R_{s_n})^2} \tag{5}$$

where each $R_{s_i} < T_{m_i} \forall i$

Table 1. Time-stamped temporal database

Transaction ID	Item	Transaction ID	Item
TD1	A, B, C, D	TD11	B, C
TD2	A, C	TD12	B, C
TD3	D	TD13	A, B, C, D
TD4	A, B, C, D	TD14	A, B, C, D
TD5	A, B, D	TD15	D
TD6	A, B	TD16	A, B, C
TD7	A, B, C, D	TD17	B, C, D
TD8	A, C	TD18	A, B, C, D
TD9	C	TD19	C
TD10	A, B, C, D	TD20	A, B, D

Minimum Possible Support Bound, (D^{min}). This is the distance which is used to decide whether pattern is similar or not and is given by $D^{min} = D^U + D^L$.

4 Case Study

Consider the Table 1 which depicts the transactions performed at two different time slots over 4-items. Tables 2 and 3 shows true support and positive, negative

Table 2. True support

Item set	True support at time slot 1	True support at time slot 2	True distance	Decision
A	0.8	0.5	0.29	×
B	0.6	0.8	0.2	×
C	0.7	0.8	0.25	×
D	0.6	0.6	0.14	×
AB	0.6	0.5	0.15	×
AC	0.6	0.4	0.2	×
AD	0.6	0.4	0.15	×
BC	0.4	0.7	0.07	✓
BD	0.5	0.5	0.1	✓
CD	0.4	0.4	0.14	×
ABC	0.4	0.4	0.14	×
ABD	0.5	0.4	0.15	×
ACD	0.4	0.3	0.21	×
BCD	0.4	0.4	0.14	×
ABCD	0.4	0.3	0.21	×

Table 3. Positive and negative supports

Item set	Positive prevalence	Negative prevalence	Item set	Positive prevalence	Negative prevalence
A	0.8,0.5	0.2,0.5	BD	0.5,0.5	0.5,0.5
B	0.6,0.8	0.4,0.2	CD	0.4,0.4	0.6,0.6
C	0.7,0.8	0.3,0.2	ABC	0.4,0.4	0.6,0.6
D	0.6,0.6	0.4,0.4	ABD	0.5,0.4	0.5,0.6
AB	0.6,0.5	0.4,0.5	ACD	0.4,0.3	0.6,0.7
AC	0.6,0.4	0.4,0.6	BCD	0.4,0.4	0.6,0.6
AD	0.6,0.4	0.4,0.6	ABCD	0.4,0.3	0.6,0.7
BC	0.4,0.7	0.6,0.3			

Table 4. Single temporal patterns

Pattern	SS	True distance	D^U	Similar	Retained
A	0.8,0.5	0.29, $\times$	0.07, ✓	$\times$	✓
B	0.6,0.8	0.20, $\times$	0, ✓	$\times$	✓
C	0.7,0.8	0.25, $\times$	0, ✓	$\times$	✓
D	0.6,0.6	0.14, $\times$	0, ✓	$\times$	✓

prevalence sequence computations respectively. Let $\Delta = 0.1$ and R = (0.4, 0.6) and distance measure is Normalized Euclidean [5,6].

Table 4 depicts true distance and maximum possible minimum bound distance computation of singleton patterns. The symbols ✓ and $\times$ denotes true and false in tables wherever they appear. Table 5 depicts sample computation of maximum and minimum possible supports of pattern AB of size, L = 2 and its corresponding bounds. The pattern is not-similar (denoted by $\times$) but retained (denoted by ✓). Table 6 depicts computations of all temporal patterns of size, $L \geq 2$. Table 7 shows record of true support computations done by proposed approach. The traditional approach requires 15 true support computations while the proposed approach requires only 10 true support computations thus reducing 33.33% computations for the case study example database. In the best case, we require just 4 computations as no item from level-1 is retained.

Table 5. Size-2 temporal pattern (AB)

Pattern	HPBSS	LPBSS	D^{min}	D^U	Since D^{min}, Δ, pattern may be similar. So, find true support
AB	0.6,0.5	0.4,0.3	0.0707, ✓	$\times$	
Pattern	True support	D^{min}	D^U	Similar	Retained
AB	0.6,0.5	0.158	0.0707	$\times$	✓

Table 6. Temporal patterns, size $|L| \geq 2$

Pattern	Similar	Retained	Pattern	Similar	Retained
AC	×	✓	ABC	×	×
AD	×	×	ABD	×	×
BC	✓	✓	ACD	×	×
BD	✓	✓	BCD	×	×
CD	×	×	ABCD	×	×

Table 7. True support computations performed

Pattern	Computed	Pattern	Computed	Pattern	Computed
A	✓	AC	×	ABC	✓
B	✓	AD	✓	ABD	×
C	✓	BC	✓	ACD	×
D	✓	BD	✓	BCD	×
AB	✓	CD	✓	ABCD	×

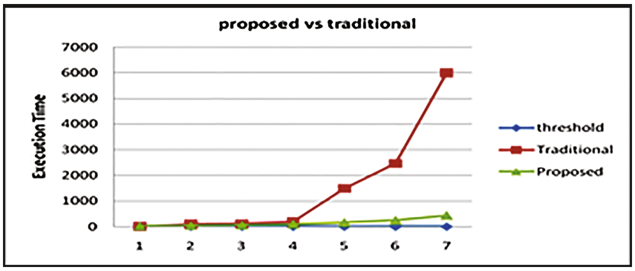


Fig. 1. Comparison of execution time

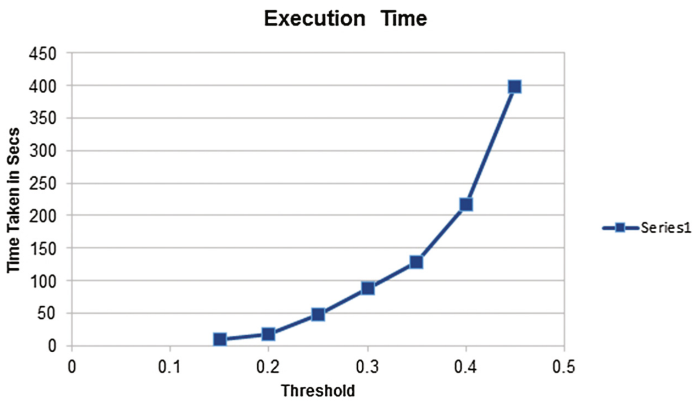


Fig. 2. Threshold vs. execution time

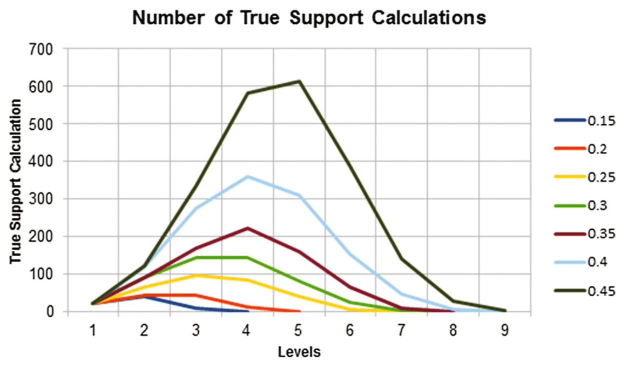


Fig. 3. True support computations

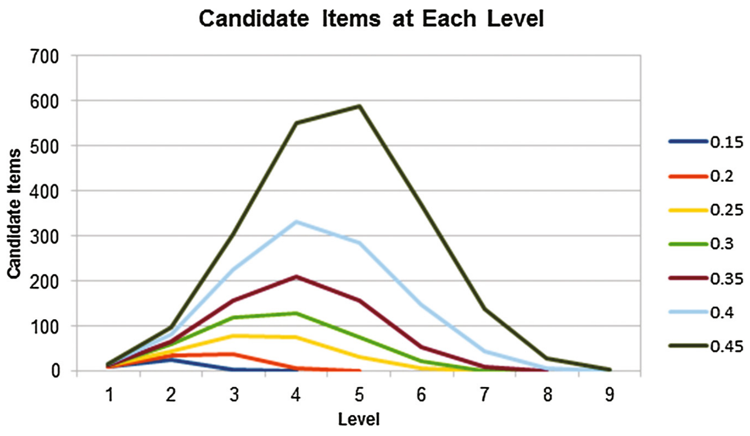
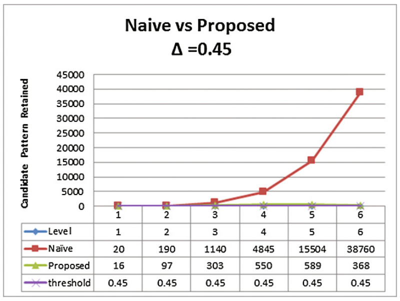
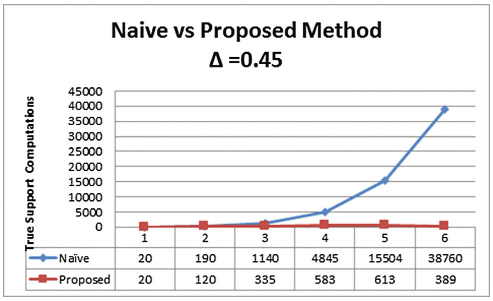


Fig. 4. Candidate items retained for different user threshold values



Level	1	2	3	4	5	6
Naïve	20	190	1140	4845	15504	38760
Proposed	16	97	303	550	589	368
threshold	0.45	0.45	0.45	0.45	0.45	0.45

Fig. 5. Candidate patterns retained at each level for $\Delta = 0.45$



	1	2	3	4	5	6
Naïve	20	190	1140	4845	15504	38760
Proposed	20	120	335	583	613	389

Fig. 6. True support comparisons at $\Delta = 0.45$

5 Results and Discussions

The results are generated from IBM synthetic data generator available online for experiment validations. It generates transactions randomly for different specifications given by user. In this paper, we used TD1000-L10-I20-T100 dataset generated and generated transactions randomly. The results are average readings obtained for all experiments done. Here, TD denotes 1000(X100) transactions is average size of transaction, I indicates items, T is number of time slots considered. Figure 1 shows comparison of execution time of Naïve approach to proposed approach. In naive approach, after Level-7, the execution times is towards exponential while the proposed approach is polynomial and terminates in finite time. The Fig. 2 shows comparison of proposed and naive approaches for variable threshold values 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45 w.r.t execution time and true support computations. Figure 3 and Fig. 4 denotes number of true support computations performed and candidate temporal pattern retained by proposed approach. Figures 5 and 6 depicts candidate pattern retained and true support computations at each level for $\Delta = 0.45$ using naive and proposed approaches.

The results show that the proposed approach generates the candidate patterns without showing exponential behavior and the number of true support computations performed are never polynomial. The advantage of proposed approach when compared to the only method available in literature [3,5,6] is that their method requires computing true supports of all sub-set candidate item sets of previous stage when estimating support bounds of all supersets of higher sizes in next level, where as our approach only requires true support of any one size (k−1) subset pattern when estimating true support of superset pattern in next levels. This condition itself eliminates excessive overhead in computing true supports as each stage generates a huge number of candidate item sets after level 6 itself.

6 Conclusions

The problem of finding temporal association patterns whose prevalence variations are similar to reference sequence is less studied in the literature and coined first by authors in their works [3,5,6]. In this paper, we design expressions to estimate support bounds of itemsets which is then used to prune candidate itemsets which cannot be similar and compute true supports of only those pattern for which it is required. This is because the distance computed between maximum bound distance obtained w.r.t true support sequence, satisfies montonicity property. The results show proposed approach out performs naive approach. In future, we wish to find temporal patterns using different similarity measures and apply normal distribution to estimate support bounds of temporal patterns.

References

1. Jin, L., Lee, Y., Seo, S., Ryu, K.H.: Discovery of temporal frequent patterns using TFP-Tree. In: Yu, J.X., Kitsuregawa, M., Leong, H.V. (eds.) WAIM 2006. LNCS, vol. 4016, pp. 349–361. Springer, Heidelberg (2006). doi:10.1007/11775300_30
2. Hirano, S., Tsumoto, S.: Mining similar temporal patterns in long time-series data and its application to medicine. In: Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2002, pp. 219–226 (2002)
3. Yoo, J.S., Shekhar, S.: Similarity-profiled temporal association mining. IEEE Trans. Knowl. Data Eng. **21**(8), 1147–1161 (2009)
4. Chen, Y.C., Peng, W.C., Lee, S.Y.: Mining temporal patterns in time interval-based data. IEEE Trans. Knowl. Data Eng. **27**(12), 3318–3331 (2015)
5. Yoo, J.S., Shekhar, S.: Mining temporal association patterns under a similarity constraint. In: Ludäscher, B., Mamoulis, N. (eds.) SSDBM 2008. LNCS, vol. 5069, pp. 401–417. Springer, Heidelberg (2008). doi:10.1007/978-3-540-69497-7_26
6. Yoo, J.S.: Temporal data mining: similarity-profiled association pattern. In: Data Mining: Foundations and Intelligent Paradigms. Intelligent Systems Reference Library, vol. 23, pp. 29–47 (2012)
7. Radhakrishna, V., Kumar, P.V., Janaki, V.: A survey on temporal databases and data mining. In: Proceedings of the International Conference on Engineering & MIS 2015 (ICEMIS 2015), 6 p. ACM, New York (2015). doi:10.1145/2832987.2833064, Article 52
8. Radhakrishna, V., Kumar, P.V., Janaki, V.: A novel approach for mining similarity profiled temporal association patterns using venn diagrams. In: Proceedings of the International Conference on Engineering & MIS 2015 (ICEMIS 2015), 9 p. ACM, New York (2015). doi:10.1145/2832987.2833071, Article 58
9. Radhakrishna, V., Kumar, P.V., Janaki, V.: An approach for mining similarity profiled temporal association patterns using gaussian based dissimilarity measure. In: Proceedings of the International Conference on Engineering & MIS 2015 (ICEMIS 2015), 6 p. ACM, New York (2015). doi:10.1145/2832987.2833069, Article 57
10. Radhakrishna, V., Kumar, P.V., Janaki, V.: A novel approach for mining similarity profiled temporal association patterns. Rev. Téc. Ing. Univ. Zulia **38**(3), 80–93 (2015)
11. Calders, T.: Deducing bounds on the support of itemsets. In: Meo, R., Lanzi, P.L., Klemettinen, M. (eds.) Database Support for Data Mining Applications. LNCS (LNAI), vol. 2682, pp. 214–233. Springer, Heidelberg (2004). doi:10.1007/978-3-540-44497-8_11

Estimating Prevalence Bounds of Temporal Association Patterns to Discover Temporally Similar Patterns

Vangipuram Radhakrishna[1(✉)], P.V. Kumar[2], V. Janaki[3], and N. Rajasekhar[4,5]

[1] VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India
radhakrishna_v@vnrvjiet.in

[2] University College of Engineering, Osmania University, Hyderabad, India
pvkumar58@gmail.com

[3] Vaagdevi Engineering College, Warangal, India
janakicse@yahoo.com

[4] Dayanandasagar College of Engineering, Bangalore, India
rajasekhar-ise@dayanandasagar.edu

[5] Institute of Aeronautical Engineering, Hyderabad, India
n.rajasekhar@iare.ac.in

Abstract. Mining Temporal Patterns from temporal databases is challenging as it requires handling efficient database scan. A pattern is temporally similar when it satisfies subset constraints. The naive and apriori algorithm designed for non-temporal databases cannot be extended to find similar temporal patterns from temporal databases. The brute force approach requires computing 2^n true support combinations for 'n' items from finite item set and falls in NP-class. The apriori or fp-tree based approaches are not directly extendable to temporal databases to obtain similar temporal patterns. In this present research, we come up with novel approach to discover temporal association patterns which are similar for pre-specified subset constraints, and substantially reduce support computations by defining expressions to estimate support bounds. The proposed approach eliminates computational overhead in finding similar temporal patterns. The results prove that the proposed method outperforms brute force approach.

Keywords: Temporal association pattern · Monotonicity · Outliers · Similar · Prevalence

1 Introduction

The concept of finding frequent itemsets is extensively studied in literature in reference to static or non-temporal databases. Researchers have also come up with several interesting algorithms to find frequent itemsets aiming at time and space efficiency. Temporal frequent itemsets have also been studied, but in a different perspective. The past studies did not consider finding similar temporal patterns whose prevalence (support) values are similar satisfying subset specifications except in research contribution of authors [3,5,6]. The subset specification involves specifying the distance measure, reference support vector, user interest threshold value. An approach for finding temporal frequent patterns is discussed in [1] which uses the concept of TFP-tree. In [2], authors discuss the importance of temporal frequent patterns with reference to medical scenario and time series applications. In [3,7–10], the temporal patterns are extracted from the input database through using the bounding concepts of supports of itemsets. They adopt the work of Calder's [11] which estimates bounds of supports. In [5,6] temporally similar natured patterns are retrieved for a set of pre-defined subset specifications. The subset constraints involve specifying reference support vector, user interest threshold value and appropriate distance metric. The drawback of this approach is that, they essentially require computations of true support sequences of all itemsets at previous level when deciding if a pattern at next level is similar or not. This is drawback of their method. We overcome this using the formal expressions designed in this paper. In [4], the patterns are mined from underlying time interval based data by using concept of Gaussian function.

In this research, we come up with the novel approach for discovering temporal patterns whose prevalence values varies similar to the reference sequence. Our approach reduces the computational overhead in finding the support values of itemsets. We now define some basic terminology followed in the design of formal expressions and algorithm procedure.

1.1 Basic Terminology

Temporal Pattern. A Temporal pattern is sequence of support values computed at every time slot defined over a finite time period, $T_p = \{t_1, t_2, ...t_n\}$ where $t_1, t_2....t_n$ are time slots.

Positive Prevalence and Positive Prevalence Sequence(PPS). Positive prevalence is support value of item set at a given time slot while positive prevalence sequence is sequence of positive support values computed at every time slot defined over finite time period. A positive temporal item set is denoted by T_n.

Negative Prevalence and Prevalence Sequence(NPS). Negative prevalence is non-existence support value of item set at a given time slot while its prevalence sequence is sequence of negative support values computed at every time slot defined over finite time period. A negative temporal itemset is denoted by using a complement notation as $\overline{T}_n$.

Reference Sequence and Threshold. It is prevalence support sequence chosen randomly and is of user interest. Threshold is the dissimilarity constraint which indicates the distance between reference pattern and temporal item set.

Pattern or Item Set Pruning. The item set is killed or pruned when maximum possible support sequence exceeds dissimilarity constraint.

Maximum Pattern Bound Support Sequence (HPBSS) and Maximum Possible Minimum Dissimilarity (D^U). It is the support sequence denoting maximum possible support value at every time slot. The distance computed of R w.r.t HPBSS is termed maximum possible minimum dissimilarity.

Minimum Pattern Bound Support Sequence (LPBSS) and Minimum Possible Minimum Dissimilarity (D^L). It is the support sequence denoting minimum possible support value at every time slot. The distance computed of R w.r.t LPBSS is termed minimum possible minimum dissimilarity.

Minimum Possible Support Bound (D^{min}). It is the sum of distances (D^U) and (D^L).

2 Related Works

Similar Temporal Patterns are those patterns whose pattern prevalence variations are similar as that of a reference sequence interest to the user and satisfying certain dissimilarity constraint. To the best of our survey and knowledge there is no contribution of algorithms or methods in this direction except the work initiated by the authors in 2009 [1–3]. In their work, authors propose an approach to find similar temporal item sets but they compute the true supports of all item sets of previous stage when estimating supports of superset item sets. In this paper, we overcome this dis-advantage of computing true support values of all item sets by defining formal expressions to estimate bound of supports. These expressions help us to estimate supports without the need to know true supports of all subsets when support for superset pattern is being estimated.

Consider the brute force approach to find temporal patterns that are similar with respect to a specified reference sequence, denoted by R, which is chosen by the user for a given threshold value, denoted by Δ. For item set with 'N' items, this approach requires generating (2N-1) item set combinations and then computing true support values for all these 2N-1 item set combinations. The complexity is thus O (2N). This means that the complexity is exponential which indicates that the complexity class of this approach is NP-class. The present research is motivated from this fact. The objective of this research is thus to reduce the computational overhead in computing true supports by estimating support bounds of item sets. The true support for item set is computed if and only if it is really needed and required.

2.1 Design Expressions to Estimate Support Bounds

Let N be the number of items in the Finite set of items, denoted by F with size of item set denoted by L. We design the expressions to estimate bounds of supports of temporal patterns by considering two cases. These include considering all those combinations of Temporal patterns of size, $|L| = 2$ and Temporal patterns of size, $|L| > 2$.

Case-1: $|\boldsymbol{L}| = \mathbf{2}$. Let T_m and T_n be two any temporal singleton patterns in Level-1, such that $T_m = (T_{m_1};\ T_{m_2};\ T_{m_3};\ .....T_{m_n})$ and $T_n = (T_{n_1};\ T_{n_2};\ T_{n_3};\ .....T_{n_n})$ where each T_{m_i} and T_{n_i} is support value at i^{th} time slot. Then the support bounds of temporal pattern of size, $|L| = 2$, is obtained using Eqs. 1 and 2.

$$[T_mT_n]_{max} = <min(T_{m_1}, T_{n_1}), min(T_{m_2}, T_{n_2}), ..., min(T_{m_p} - T_{n_p})> \quad (1)$$

$$[T_mT_n]_{min} = <max\{(1 - \overline{T_{m_1}} - \overline{T_{n_1}}), 0\}, max\{(1 - \overline{T_{m_2}} - \overline{T_{n_2}}), 0\}> \quad (2)$$

where $T_m, \overline{T}_n$ is positive and negative prevalence sequence separately.

Case-2: $|\boldsymbol{L}| > \mathbf{2}$. Let T_mT_n be the temporal pattern of size, L, for which we must estimate support bounds. We divide all such patterns into two sub-patterns. The first sub pattern (T_m) is assumed to be of length, L-1 and second sub pattern (T_n) is equal to length, L = 1. To estimate support bounds of temporal pattern of the form T_mT_n whose size is $|L| > 2$, we assume that T_m is temporal pattern of length (n-1) and T_n is singleton temporal pattern of length = 1. Let T_mT_n be any two given temporal patterns, such that $T_m = (T_{m_1};\ T_{m_2};\ T_{m_3}....T_{m_n})$ and $T_n = (T_{n_1};\ T_{n_2};\ T_{n_3}.....T_{n_n})$ where each T_{m_i} and T_{n_i} is support value at i^{th} time slot. Then the maximum and minimum support bound of temporal pattern of size, $|L| > 2$ is obtained using Eqs. 3 and 4.

$$[T_mT_n]_{max} = <(T_{m_1} - max\{(1 - \overline{T_{m_1}} - T_{n_1}), 0\})..., (T_{m_2} - max\{(1 - \overline{T_{m_2}} - T_{n_2}), 0\}) \quad (3)$$

$$[T_mT_n]_{min} = <max\{(1 - \overline{T_{m_1}} - T_{n_1}), 0\}..., max\{(1 - \overline{T_{m_1}} - T_{n_1}), 0\} \quad (4)$$

where T_m is positive temporal pattern of size, (L-1), $\overline{T}_n$ is singleton temporal pattern of length = 1.

2.2 Proposed Similar Temporal Association Pattern Mining Algorithm

Algorithm: Mining Similar Temporal Patterns

Input: Temporal Database with time stamped transactions; Reference sequence; User specified dissimilarity threshold and distance function.

Output: Similar temporal patterns

Step-1: Obtain Positive Prevalence and Negative Prevalence for each singleton temporal item (temporal pattern) from the finite set of items defined by F. This is the first time; we compute true support values of all temporal items initially.

Step-2: Obtain Positive Prevalence Sequence and Negative Prevalence Sequence for each singleton temporal item (temporal pattern) using prevalence values computed in step-1.

Step-3: // Pruning strategy-1 for singleton patterns of size, L = 1.
Compute true distance between reference and each temporal pattern of size = 1. If the true distance satisfies dissimilarity constraint, the pattern is treated as similar and also retained. If the true distance exceeds dissimilarity constraint, the pattern is treated dissimilar. To decide to prune or not, we compute distance (D^U) which is the maximum possible deviation above reference sequence defined in Sect. 2.3. If this distance exceeds threshold value, then prune the temporal pattern. In this case, we say the temporal pattern is not retained. Otherwise, i.e. if dissimilarity condition is satisfied, then we retain the candidate temporal pattern to compute prevalence values (support values) of higher size temporal patterns.

Step-4: // Pruning strategy-2 for temporal patterns of size, $|L| > 1$.

Case-1: Obtain the Maximum Pattern Bound Support Sequence and Minimum Possible Support Bound for all patterns of $size > 1$. Now, if the Minimum Possible Support Bound (D^{Min}) exceeds threshold, the pattern is dissimilar. To decide whether this temporal pattern be pruned or not, we compute distance (D^U) which is the maximum possible deviation above reference sequence. If $D^U > threshold$, $\varDelta$, then prune the temporal pattern. In this case, we say the temporal pattern is not retained. However, if dissimilarity condition is satisfied, then we retain the candidate temporal pattern to compute prevalence values (support values) of higher size temporal patterns.

Case-2: Obtain Minimum Possible Support Bound (D^{min}) for all patterns of $size > 1$. Now, if the Minimum Possible Support Bound (D^{min}) satisfies threshold, the pattern may be similar but cannot be confirmed always, as it depends on item set distribution. This is because we are estimating maximum and minimum possible bounds and comparing these with reference. So, to decide whether this temporal pattern is temporally similar, we scan the database to find patterns true support sequence and from this we find its true distance with reference, R. If the $truedistance \leq threshold$, then temporal pattern is similar and is also retained. If the true distance is violating dissimilarity condition then we find If $D^U > threshold$, $\varDelta$, if so we prune the temporal pattern. In this case, we say the temporal pattern is not retained. However, if $D^U \leq \varDelta$, then we retain the candidate temporal pattern to compute prevalence values (support values) of higher size temporal patterns.

Step-5: Prune all other candidate temporal patterns based on D^U. In the process of verifying temporal patterns for similarity w.r.t reference, discard and prune all temporal superset patterns, if there is at least one subset pattern of this superset pattern which is not retained. i.e. for temporal pattern T, to be considered similar all its sub-patterns must satisfy the retaining condition. If there is at least one subset such that it does not satisfy dissimilarity condition, then the superset pattern, T, is not similar.

Step-6: Output all candidate temporal patterns which are similar and retained.

2.3 Computing Support Bounds

To estimate distance bounds, we use the following equations as defined in research contributions of authors [3,5,6].

Maximum Possible Minimum Dissimilarity, ($\boldsymbol{D^U}$) Let, $T_m = (T_{m_1}; T_{m_2}; T_{m_3} .... T_{m_n})$ and $R_s = (R_{s_1}; R_{s_2}; R_{s_3} .... R_{s_n})$ be the Maximum Pattern Bound Support Sequence and Reference Sequence respectively. Now, D^U is defined using Eq. 4.

$$D^U = \sqrt{(R_{s_1} - T_{m_1})^2 + (R_{s_2} - T_{m_2})^2 + ... + (R_{s_n} - T_{m_n})^2} \tag{5}$$

where each $R_{s_i} > T_{m_i} \forall \ i$

Minimum Possible Minimum Dissimilarity, ($\boldsymbol{D^L}$) Let, $T_m = (T_{m_1};\ T_{m_2};\ T_{m_3} ... T_{m_n})$ and $R_s = (R_{s_1};\ R_{s_2};\ R_{s_3} ... R_{s_n})$ be the Minimum Pattern Bound Support Sequence and Reference Sequence respectively. Now, D^L is defined using Eq. 5.

$$D^L = \sqrt{(T_{m_1} - R_{s_1})^2 + (T_{m_2} - R_{s_2})^2 + ... + (T_{m_n} - R_{s_n})^2} \tag{6}$$

where each $R_{s_i} < T_{m_i} \forall i$

Minimum Possible Support Bound, ($\boldsymbol{D^{min}}$). This is the distance which is used to decide whether pattern is similar or not and is given by $D^{min} = D^U + D^L$.

3 Case Study

Consider the Table 1 which depicts the transactions performed at two different time slots over 4-items. Tables 2 and 3 shows true support and positive, negative prevalence sequence computations respectively. Let $\Delta = 0.1$ and R = (0.4, 0.6) and distance measure is Normalized Euclidean [5,6].

Table 4 depicts true distance and maximum possible minimum bound distance computation of singleton patterns. The symbols ✓ and × denotes true and false in tables wherever they appear. Table 5 depicts sample computation of maximum and minimum possible supports of pattern AB of size, L = 2 and its corresponding bounds. The pattern is not-similar (denoted by ×) but retained (denoted by ✓).

Table 6 depicts computations of all temporal patterns of size, $L \geq 2$. Table 7 shows record of true support computations done by proposed approach. The traditional approach requires 15 true support computations while the proposed approach requires only 10 true support computations thus reducing 33.33% computations for the case study example database. In the best case, we require just 4 computations as no item from level-1 is retained.

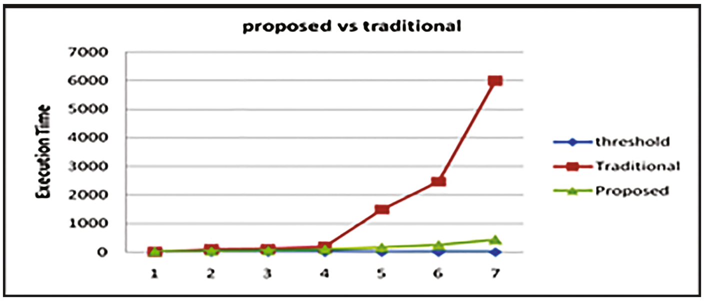


Fig. 1. Comparison of execution time

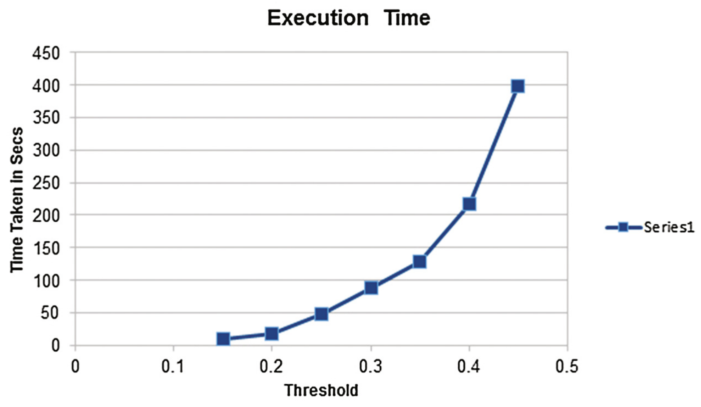


Fig. 2. Threshold vs. execution time

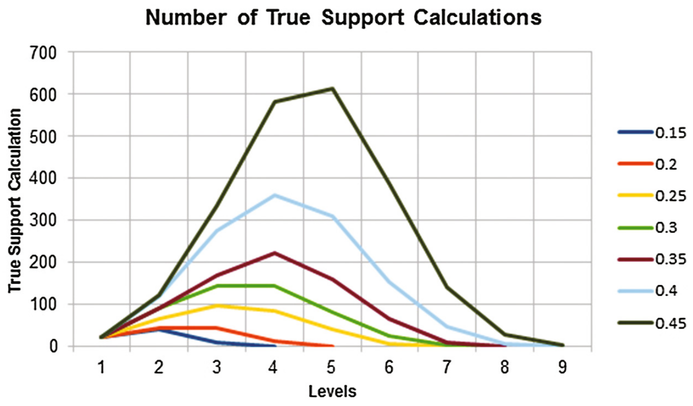


Fig. 3. True support computations

Table 1. Time-stamped temporal database

Transaction ID	Item	Transaction ID	Item
TD1	A, B, C, D	TD11	B, C
TD2	A, C	TD12	B, C
TD3	D	TD13	A, B, C, D
TD4	A, B, C, D	TD14	A, B, C, D
TD5	A, B, D	TD15	D
TD6	A, B	TD16	A, B, C
TD7	A, B, C, D	TD17	B, C, D
TD8	A, C	TD18	A, B, C, D
TD9	C	TD19	C
TD10	A, B, C, D	TD20	A, B, D

Table 2. True support

Itemset	True support at time slot 1	True support at time slot 2	True distance	Decision
A	0.8	0.5	0.29	×
B	0.6	0.8	0.2	×
C	0.7	0.8	0.25	×
D	0.6	0.6	0.14	×
AB	0.6	0.5	0.15	×
AC	0.6	0.4	0.2	×
AD	0.6	0.4	0.15	×
BC	0.4	0.7	0.07	✓
BD	0.5	0.5	0.1	✓
CD	0.4	0.4	0.14	×
ABC	0.4	0.4	0.14	×
ABD	0.5	0.4	0.15	×
ACD	0.4	0.3	0.21	×
BCD	0.4	0.4	0.14	×
ABCD	0.4	0.3	0.21	×

4 Experimental Results and Discussions

The results are generated from IBM synthetic data generator available online for experiment validations. It generates transactions randomly for different specifications given by user. In this paper, we used TD1000-L10-I20-T100 dataset generated and generated transactions randomly. The results are average readings obtained for all experiments done. Here, TD denotes 1000(X100) transactions is average size of transaction, I indicates items, T is number of time slots considered. Figure 1 shows comparison of execution time of Naïve approach to proposed approach. In naive approach, after Level-7, the execution times is

Table 3. Positive and negative supports

Item set	Positive prevalence	Negative prevalence	Item set	Positive prevalence	Negative prevalence
A	0.8, 0.5	0.2, 0.5	BD	0.5, 0.5	0.5, 0.5
B	0.6, 0.8	0.4, 0.2	CD	0.4, 0.4	0.6, 0.6
C	0.7, 0.8	0.3, 0.2	ABC	0.4, 0.4	0.6, 0.6
D	0.6, 0.6	0.4, 0.4	ABD	0.5, 0.4	0.5, 0.6
AB	0.6, 0.5	0.4, 0.5	ACD	0.4, 0.3	0.6, 0.7
AC	0.6, 0.4	0.4, 0.6	BCD	0.4, 0.4	0.6, 0.6
AD	0.6, 0.4	0.4, 0.6	ABCD	0.4, 0.3	0.6, 0.7
BC	0.4, 0.7	0.6, 0.3			

Table 4. Single temporal patterns

Pattern	SS	True distance	D^U	Similar	Retained
A	0.8, 0.5	0.29, ×	0.07, ✓	×	✓
B	0.6, 0.8	0.20, ×	0, ✓	×	✓
C	0.7, 0.8	0.25, ×	0, ✓	×	✓
D	0.6, 0.6	0.14, ×	0, ✓	×	✓

towards exponential while the proposed approach is polynomial and terminates in finite time. The Fig. 2 shows comparison of proposed and naive approaches for variable threshold values 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45 w.r.t execution time and true support computations. Figures 3 and 4 denotes number of true support computations performed and candidate temporal pattern retained by proposed approach. Figures 5 and 6 depicts candidate pattern retained and true support computations at each level for $\varDelta = 0.45$ using naive and proposed approaches.

The results show that the proposed approach generates the candidate patterns without showing exponential behavior and the number of true support computations performed are never exponential. The advantage of proposed approach when compared to the only method available in literature [3,5,6] is that their method requires computing true supports of all sub-set candidate item sets

Table 5. Size-2 temporal pattern (AB)

Pattern	HPBSS	LPBSS	D^{min}	D^U	Since D^{min}, $\varDelta$, pattern may be similar. So, find true support
AB	0.6, 0.5	0.4, 0.3	0.0707, ✓	×	
Pattern	True support	D^{min}	D^U	Similar	Retained
AB	0.6, 0.5	0.158	0.0707	×	✓

Table 6. Temporal patterns, size $|L| \geq 2$

Pattern	Similar	Retained	Pattern	Similar	Retained
AC	×	✓	ABC	×	×
AD	×	×	ABD	×	×
BC	✓	✓	ACD	×	×
BD	✓	✓	BCD	×	×
CD	×	×	ABCD	×	×

Table 7. True support computations performed

Pattern	Computed	Pattern	Computed	Pattern	Computed
A	✓	AC	×	ABC	✓
B	✓	AD	✓	ABD	×
C	✓	BC	✓	ACD	×
D	✓	BD	✓	BCD	×
AB	✓	CD	✓	ABCD	×

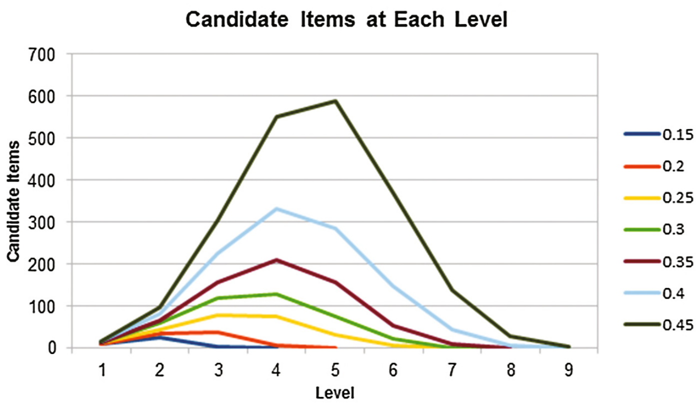
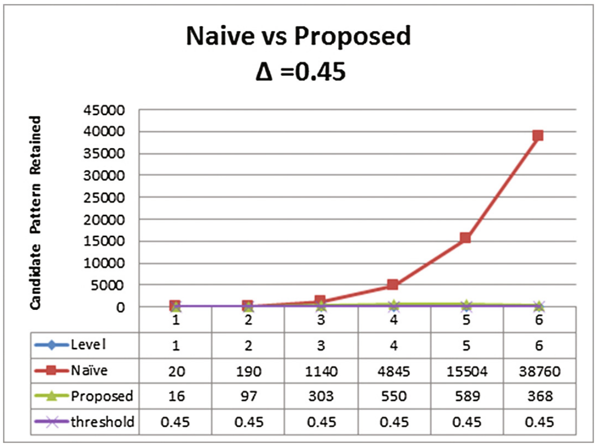


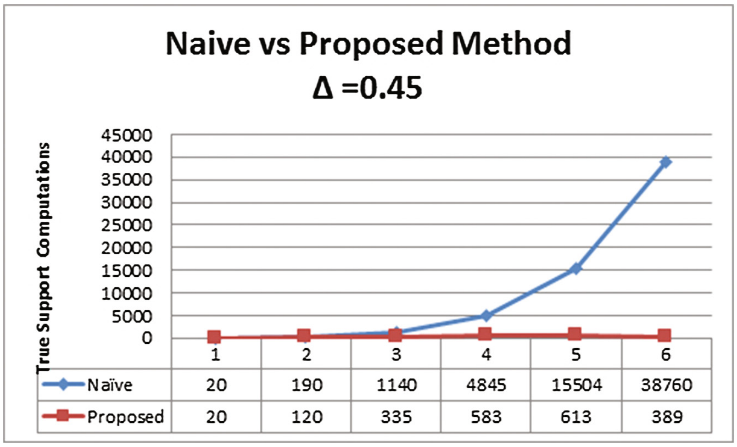
Fig. 4. Candidate items retained for different user threshold values

of previous stage when estimating support bounds of all supersets of higher sizes in next level, where as our approach only requires true support of any one size (k-1) subset pattern when estimating true support of superset pattern in next levels. This condition itself eliminates excessive overhead in computing true supports as each stage generates a huge number of candidate item sets after level 6 itself.



	1	2	3	4	5	6
Level	1	2	3	4	5	6
Naïve	20	190	1140	4845	15504	38760
Proposed	16	97	303	550	589	368
threshold	0.45	0.45	0.45	0.45	0.45	0.45

Fig. 5. Candidate patterns retained at each level for $\Delta = 0.45$



	1	2	3	4	5	6
Naïve	20	190	1140	4845	15504	38760
Proposed	20	120	335	583	613	389

Fig. 6. True support comparisons at $\Delta = 0.45$

5 Conclusions

Mining Temporal Patterns from temporal databases is challenging as it requires handling efficient database scan. A pattern is temporally similar when it satisfies subset constraints. The objective of present research is to design formal expressions which can be used to estimate item set prevalence bounds without the need to compute true prevalence of item sets unless it is required. In other words we compute true prevalence values only when maximum possible minimum dissimilarity value exceed threshold value. The proposed approach eliminates computational overhead in finding similar temporal patterns. The results prove that the proposed method outperforms brute force approach which requires 2^n true support computations for n items.

References

1. Jin, L., Lee, Y., Seo, S., Ryu, K.H.: Discovery of temporal frequent patterns using TFP-tree. In: Yu, J.X., Kitsuregawa, M., Leong, H.V. (eds.) WAIM 2006. LNCS, vol. 4016, pp. 349–361. Springer, Heidelberg (2006). doi:10.1007/11775300_30
2. Hirano, S., Tsumoto, S.: Mining similar temporal patterns in long time-series data and its application to medicine. In: Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2003, pp. 219–226 (2002)
3. Yoo, J.S., Shekhar, S.: Similarity-profiled temporal association mining. IEEE Trans. Knowl. Data Eng. **21**(8), 1147–1161 (2009)
4. Chen, Y.C., Peng, W.C., Lee, S.Y.: Mining temporal patterns in time interval-based data. IEEE Trans. Knowl. Data Eng. **27**(12), 3318–3331 (2015)
5. Yoo, J.S., Shekhar, S.: Mining temporal association patterns under a similarity constraint. In: Ludäscher, B., Mamoulis, N. (eds.) SSDBM 2008. LNCS, vol. 5069, pp. 401–417. Springer, Heidelberg (2008). doi:10.1007/978-3-540-69497-7_26
6. Yoo, J.S.: Temporal data mining: similarity-profiled association pattern. Data Mining: Foundations and Intelligent Paradigms. Intelligent Systems Reference Library, vol. 23, pp 29–47 (2012)
7. Radhakrishna, V., Kumar, P.V., Janaki, V.: A Survey on temporal databases and data mining. In: Proceedings of the International Conference on Engineering & MIS 2015 (ICEMIS 2015), Article 52, 6 pages. ACM, New York (2015). doi:10.1145/2832987.2833064
8. Radhakrishna, V., Kumar, P.V., Janaki, V.: A novel approach for mining similarity profiled temporal association patterns using venn diagrams. In: Proceedings of the International Conference on Engineering & MIS 2015 (ICEMIS 2015), Article 58, 9 pages. ACM, New York (2015). doi:10.1145/2832987.2833071
9. Radhakrishna, V., Kumar, P.V., Janaki, V.: An approach for mining similarity profiled temporal association patterns using gaussian based dissimilarity measure. In: Proceedings of the International Conference on Engineering & MIS 2015 (ICEMIS 2015), Article 57, 6 pages. ACM, New York (2015). doi:10.1145/2832987.2833069
10. Radhakrishna, V., Kumar, P.V., Janaki, V.: A novel approach for mining similarity profiled temporal association patterns. Rev. Téc. Ing. Univ. Zulia. **38**(3), 80–93 (2015)
11. Calders, T.: Deducing bounds on the support of itemsets. In: Meo, R., Lanzi, P.L., Klemettinen, M. (eds.) Database Support for Data Mining Applications. LNCS (LNAI), vol. 2682, pp. 214–233. Springer, Heidelberg (2004). doi:10.1007/978-3-540-44497-8_11

An Approach for Imputation of Medical Records Using Novel Similarity Measure

Yelipe UshaRani[1(✉)] and P. Sammulal[2]

[1] VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India
ushayelipe@gmail.com
[2] JNTUH College of Engineering, Jagityal, Karimnagar, India
sammulalporika@gmail.com

Abstract. Missing values are quite common in medical records. Fixing missing values is a challenging task to data mining analysts as any error in imputation process leads to potential hazards. In the present research, the main objective is to impute missing values using a new similarity measure and applying class based cluster approach which is used to perform dimensionality reduction. The proposed approach is demonstrated using an effective case study. The results show the proposed measure performs better and is efficient.

Keywords: Prediction · Classification · Clustering · Medical record · Missing values · Distance measure

1 Introduction

Medical data records are challenging to handle because of various hidden challenges which are generated under various practical conditions. Many times when we aim to mine medical records, we have first challenge of imputation. This is mainly from the obvious and widely accepted fact that medical records are not free from missing attribute values. In this scenario, the process of imputation received wide range attention from data mining and data analysts. Medical data records must be also normalized and scaled to perform any analysis. The process of normalization must be done before imputation so that the imputation process yields correct results. This affects classification accuracies. There are several preprocessing stages a medical record must be processed before analysis is performed. Dimension of record is also a concern when it must be analyzed. We must see that dimensions which are not affecting the final accuracies be only eliminated or discarded. In [1], the researchers debate whether to consider missing values or simply eliminate them from consideration for analysis. They discuss these using decision tree concepts. Clustering data records is a known problem and is also used for medical data. The work of authors in [2] mainly targets how to handle missing values using clustering. The support vector regression and clustering are used in [3] to perform imputation. The authors [4,5] discuss various problems with missing values and how to handle mixed attributes respectively.

In [6], authors discuss novel framework and a discussion on using auto regression method to handle missing values is done in [7]. In [9–11] authors come up with a new approach to handle missing values.

2 Proposed Imputation Method

The present work targets imputing missing attribute values and performing record classification after imputing. The approach is class based clustering approach. Here we cluster records with no missing values equal to number of classes. Then we obtain distance from these records to cluster centers. Our approach does not consider standard deviation of clusters generated. We aim to achieve dimensionality reduction of records to a dimension equal to number of class labels. Then we represent these records as a vector of values. This is then followed by finding distance between these transformed records and missing attribute value records. The imputation is done considering record to which the test record distance is minimum.

2.1 Research Objective

To impute missing attribute values in a given dataset of medical records and then perform classification of medical records. This is aimed to be achieved by targeting dimensionality reduction of initial attribute set of dataset. In this process, we map initial m-dimensional records to equivalent p-dimensional records where p is number of decision classes in dataset. The process of filling missing values is carried out using this transformed low dimensional representation of medical records.

2.2 Problem Definition

Given a set of medical records having both missing and non-missing attribute values, we aim to fill all attribute values of medical records and achieve high classification accuracies on datasets, so as to prove proposed method is feasible and adoptable for classification and imputation.

2.3 Proposed Framework

The framework for proposed imputation approach is discussed below. Initially we group records into two different groups, those without missing attribute values (G1) and another group (G2) having missing attribute values. The idea is to consider all records in group G1 (having no missing values) and first obtain clusters equal to number of decision labels and use knowledge of these clusters information to achieve dimensionality reduction and imputation. From clusters obtained considering records in first group (G1) and second group (G2), we obtain cluster mean for all clusters. Each record in (G1) is transformed into a p-dimensional vector where each value in vector is distance value from cluster center. Similarly, each record in group (G2) is also transformed to p-dimensional representation but discarding missing attribute values. We then find distance between missing attribute value record and transformed records of group (G1). The imputation is done considering record with minimum distance. The best approach for imputation is to consider decision class of medical record to be imputed and then perform imputation considering medical record to which this record has minimum distance w.r.t that class. In the case of numerical attribute, we can fill the mean of attribute value. After imputation is done, we have final set of medical records, with no missing values. This record set can be then used for finding classification accuracies. For prediction, we use same procedure adopted for imputation but determine class labels, instead of imputing missing values. The importance of the present approach is that we may impute and classify medical records by also reducing dimensionality.

3 Our Method

Step-1: Scan Dataset of Medical Records

$$Group_1 = Union\ (R_i \mid R_i[A_K]) \neq \Phi,\ \ \forall i, k) \tag{1}$$

and

$$Group_2 = Union\ (R_i \mid R_i[A_K]) = \Phi,\ \ \exists i, k) \tag{2}$$

with Φ, denotes undefined (empty or missing values), i denote indices of medical record with $i \in (1, m - h)$ and k denotes attribute of feature set with $k \in (1, n)$. The medical records present in the group,G_1, are used to build the knowledge base. It is this knowledge base, over which we test the medical records in group, G_2.

Step-2: Cluster Medical Records in Group, G_1. After classifying medical records in to two groups, G_1 and G_2, the objective is to obtain maximum number of decision classes. Let g be total number of decision classes. Cluster all those medical records in group, G_1 to a number of clusters equal to g. The reason for obtaining clusters is dimensionality reduction. Here we choose to transform records of higher dimensions to their low dimension representation.

Step-3: Determine Mean Vector of Each Cluster. After step-2, we have clusters formed. We find mean vector of each cluster [8,11]. Let cluster represented as C_d denotes d^{th} - cluster consisting four records namely $R_1, R_5, R_7,\ and\, R_9$ with only a single attribute say, A_1. If every record has single attribute [10], then cluster mean is given by Eq. (3).

$$\mu_d = \frac{R_1[A_1] + R_5[A_1] + R_7[A_1] + R_9[A_1]}{4} \tag{3}$$

The cluster mean of g_{th} cluster is computed using generalized Eq. (4) as followed in [9–11]

$$\mu_g = U_k[\frac{\sum R_l[A_k] \mid l \in \{l, q\} \ for \ each \ k \in \{1, n\}}{|l|}] \tag{4}$$

U_k Represents union of all values each separated by a symbol comma. Each cluster center, μ_g, is given by Eq. (5).

$$\mu_g = (\mu_g^1, \mu_g^2, \mu_g^3, ..., \mu_g^n) \tag{5}$$

Here μ_g is a sequence of 'n' values denoting gth cluster mean over 'n' attributes and μ_g^i indicates mean of i^{th} attribute in g^{th} cluster. The value of 'n' represents total number of attributes; $|g|$ indicates number of clusters.

Step-4: Compute Distance Between Each Record, R_i and Cluster Centers. For each cluster obtained, compute distance of every record to the mean of each cluster. This is done by finding Euclidean distance between each medical record with 'n' attributes and mean vector of every cluster. The p-dimensional vector obtained is the representative for each record. Each element from p-dimensional vector is a distance value to p^{th} cluster mean. We find distance using traditional Euclidean measure as followed in our previous work [9–11]. The distance is computed for records in both groups, G_1 and G_2.

Step-5: Find Nearest Record. For each missing record in group G_2 find similarity of this test record to each record in G_1, using proposed measure discussed in Sect. 5. The record is nearest to record whose similarity is maximum. So, this record is best choice for imputation to be carried. For categorical, we may directly impute the corresponding attribute value. For numeric attribute, we may impute frequency or same value as that of the corresponding attribute w.r.t missing. For similarity estimation, we use proposed measure instead of traditional Euclidean distance measure widely adopted.

4 Case Study

Consider medical record dataset consisting 9 medical records. In this section, we discuss how to impute missing values considering the medical dataset records in Table 1. We first normalize the medical record dataset, shown in Table 2. This is achieved by first replacing the categorical data attributes Z_1 and Z_3. The column Z_1 has 3 distinct values and column Z_3 has 2 distinct values. We assign $d_{11} = 1, d_{12} = 2, d_{13} = 3$ and $h_{31} = 1, h_{32} = 2$ for attribute values in Table 2.

In Table 2 below, NZ_1 and NZ_3 denote normalized data attribute values. We now discuss the proposed procedure for imputation of medical attribute values. We consider Table 3 having medical records with missing and non-missing attribute values. Split the dataset into two sets of records. One consisting missing attribute value records and other set with records having no missing attribute

Table 1. Medical dataset

Records	Attributes				Disease level
	Z_1	Z_2	Z_3	Z_4	
R_1	d_{11}	5	h_{31}	10	L1
R_2	d_{13}	7	h_{31}	5	L1
R_3	d_{11}	7	h_{32}	7	L1
R_4	d_{12}	5	h_{31}	10	L1
R_5	d_{13}	3	h_{32}	7	L2
R_6	d_{12}	9	h_{31}	10	L2
R_7	d_{11}	5	h_{32}	3	L2
R_8	d_{13}	6	h_{32}	7	L2
R_9	d_{12}	6	h_{32}	10	L2

Table 2. Normalized records

Records normalized	Attributes			
	NZ_1	Z_2	NZ_3	Z_4
NR_1	1	5	1	10
NR_2	3	7	1	5
NR_3	1	7	2	7
NR_4	2	5	1	10
NR_5	3	3	2	7
NR_6	2	9	1	10
NR_7	1	5	2	3
NR_8	3	6	2	7
NR_9	2	6	2	10

Table 3. Normalized records with and without missing values

Records normalized	NZ_1	Z_2	NZ_3	Z_4
NR_1	1	5	1	10
NR_2	3	7	1	5
NR_3	**1**	**7**	**?**	**7**
NR_4	2	5	1	10
NR_5	**3**	**3**	**2**	**?**
NR_6	2	9	1	10
NR_7	1	5	2	3
NR_8	3	6	2	7
NR_9	2	6	2	10

Table 4. Records to be imputed

Records normalized	Z_1	Z_2	Z_3	Z_4
NR_3	1	7	?	7
NR_5	3	3	2	?

Table 5. Records free from missing values

Records normalized	Z_1	Z_2	Z_3	Z_4
NR_1	1	5	1	10
NR_2	3	7	1	5
NR_4	2	5	1	10
NR_6	2	9	1	10
NR_7	1	5	2	3
NR_8	3	6	2	7
NR_9	2	6	2	10

Table 6. Clusters

Cluster	Medical records
Cluster-1	NR_7; NR_8; NR_2
Cluster-2	NR_4; NR_6; NR_1; NR_9

Table 7. Generated clusters with mean

Cluster	$Mean_1$	$Mean_2$	$Mean_3$	$Mean_4$
Cluster-1	2.33	6	1.67	5
Cluster-2	1.75	6.25	1.25	10

Table 8. Records in first cluster

Record	Z_1	Z_2	Z_3	Z_4
NR_2	3	7	1	5
NR_7	1	5	2	3
NR_8	3	6	2	7

Table 9. Records in second cluster

Record	Z_1	Z_2	Z_3	Z_4
NR_1	1	5	1	10
NR_4	2	5	1	10
NR_6	2	9	1	10
NR_9	2	6	2	10

Table 10. Distance of records in Table 5 to clusters

Record	Cluster-1	Cluster-2
NR_1	5.312459	1.47902
NR_2	1.374369	5.214163
NR_4	5.153208	1.299038
NR_6	5.878397	2.772634
NR_7	2.624669	7.189402
NR_8	2.134375	3.344772
NR_9	5.022173	0.829156

Table 11. Distance of records in Table 4 to clusters

Record	Cluster-1	Cluster-2
NR_3	2.603417	3.181981
NR_5	3.091206	3.561952

Table 12. Similarity of R3 w.r.t cluster-1 records

Record	Similarity to C1	Similarity to C2	Final similarity
NR_1	0.000000	0.000000	0.000000
NR_2	0.022234	0.000000	0.000000
NR_4	0.000000	0.000000	0.000000
NR_6	0.000000	0.307716	0.000000
NR_7	0.998863	0.000000	0.000000
NR_8	**0.574456**	**0.829944**	**0.476766**
NR_9	0.000000	0.000000	0.000000

Table 13. Similarity of R5 w.r.t cluster-1 records

Record	Similarity to C1	Similarity to C2	Final similarity
NR_1	0.000004	0.000000	0.000000
NR_2	0.000595	0.000000	0.000000
NR_4	0.000022	0.000000	0.000000
NR_6	0.000000	0.012499	0.000000
NR_7	0.577859	0.000000	0.000000
NR_8	**0.099576**	**0.717665**	**0.071462**
NR_9	0.000083	0.176880	0.000015

Table 14. Imputed record, R3

Record	NZ_1	Z_2	NZ_3	Z_4
NR_3	1	7	**2**	7

Table 15. Imputed record, R5

Record	NZ_1	Z_2	NZ_3	Z_4
NR_5	3	3	2	**7**

values. Table 3 is split into two tables Tables 4 and 5 respectively. Cluster medical records in Table 5 into two clusters say C1 and C2. This is because in our case, medical records are of only two classes. Records from R1 to R4 belong to Class, C1 and R5 to R9 are of class, C2. The clusters obtained are shown in Table 6 consisting records R2, R7, R8 in cluster-1 and R9, R6, R4, R1 in cluster-2. Table 7 represents mean of clusters computed for two clusters C1 and C2. Tables 8 and 9 represents records in both clusters. Table 10 represents distance of records free from missing values to the two clusters generated. In a Similar way, distance of records with missing attribute values to generated clusters is recorded in Table 11. Each record is represented as two dimensional (2-D) vector now. Tables 12 and 13 records similarity of R3 and R5 to all other records expressed as 2D vectors. The last column of Tables 12 and 13 denotes the similarity score obtained using proposed measure. The Imputed values are recorded in Tables 14 and 15. The values are imputed by choosing those records to which similarity happens to be maximum. In our case, the similarity score of both R3 and R5 is maximum to record R8. Hence R8 is the proper and better choice for carrying out the imputation.

5 Similarity Measure for Imputation

Let, R_m and R_n be two records that are expressed as p-dimensional vectors using the proposed approach for imputation which are obtained after class based clustering is carried out as given by Eq. 6.

$$R_m = (R_{m_1}, R_{m_2}, R_{m_3}, ...R_{m_p}) \; and \; R_n = (R_{n_1}, R_{n_2}, R_{n_3}, ...R_{n_p}) \tag{6}$$

The two records are compared for similarity using the similarity function defined below by Eq. 7.

$$IM.Sim = e^{-(\frac{R_{m_1} - R_{n_1}}{\sigma})^2} * e^{-(\frac{R_{m_2} - R_{n_2}}{\sigma})^2} * ... * e^{-(\frac{R_{m_p} - R_{n_p}}{\sigma})^2} \tag{7}$$

The record to which similarity of missing attribute value record is maximum happens to be best choice for imputation. For imputation, when the target attribute value is categorical, we choose the corresponding attribute value of record with maximum similarity for imputation and when it happens to be a numerical attribute; either the average value of class records of that class or the same value of corresponding is better choice. Equation 7 evaluates to values between 0 and 1, which is not possible with Euclidean and is a dis-advantage with Euclidean measure. Here we overcome the said dis-advantage and achieve high classification accuracy and accurate imputation using proposed approach and also similarity measure for imputation. Here σ refers to standard deviation of all corresponding attribute values of p-dimensional vector discussed.

Table 16. Accuracies as achieved in literature

Method	Accuracy %	Reference
LDA	84.5	Weiss
25-NN, Stand, Euclid	83.6 ± 0.5	WD/KG repeat
C-MLP2LN	82.5	RA, estimated
FSM	82.2	Rafal Adamczak
MLP+backprop	81.3	Weiss

6 Results and Discussions

The proposed imputation procedure is applied on Cleveland dataset available on the web in UCLA repository for carrying experiments. The dataset is given as input for the proposed algorithm and the missing values are imputed. The classification achieved on the dataset is on average 87.6% accuracy which is better than [9–11] and the results specified in [12]. The results specified in Table 16 below were the results obtained with the leave-one-out test, % of accuracy given, 2 classes used as specified in [12]. This shows the proposed approach is better compared to existing imputations carried out. The reason behind accuracy improvement is the dimensionality reduction achieved and the similarity measure which has tight bounds between values 0 and 1.

7 Conclusions

Medical records are diverse in nature and are also mostly not free from missing values of attributes. The idea and objective of this work is to devise a procedure to fill all missing attribute values. For this, we come up dimensionality reduction oriented approach. We reduce the records by transforming them into p-dimensional records where dimension is number of classes. The reduced records are then used to perform imputation. In present case, we use Euclidean measure and class based clustering concept to perform dimensionality reduction and use the proposed measure to determine similarity between records to choose record which is the best choice and perform efficient and accurate imputation. In future, we wish to extend the approach to perform classification and achieve accurate disease prediction.

References

1. Zhang, S., Qin, Z., Ling, C.X., Sheng, S.: "Missing is useful": missing values in cost-sensitive decision trees. IEEE Trans. Knowl. Data Eng. **17**(12), 1689–1693 (2005)
2. Zhang, C., Qin, Y., Zhu, X., Zhang, J., Zhang, S.: Clustering-based missing value imputation for data preprocessing. In: 2006 IEEE International Conference on Industrial Informatics, pp. 1081–1086 (2006)

3. Wang, L., Fu, D., Li, Q., Mu, Z.: Modelling method with missing values based on clustering and support vector regression. J. Syst. Eng. Electron. **21**(1), 142–147 (2010)
4. Kirkpatrick, B., Stevens, K.: Perfect phylogeny problems with missing values. IEEE/ACM Trans. Comput. Biol. Bioinf. **11**(5), 928–941 (2014)
5. Zhu, X., Zhang, S., Jin, Z., Zhang, Z., Xu, Z.: Missing value estimation for mixed-attribute data sets. IEEE Trans. Knowl. Data Eng. **23**(1), 110–121 (2011)
6. Farhangfar, A., Kurgan, L.A., Pedrycz, W.: A novel framework for imputation of missing values in databases. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **37**(5), 692–709 (2007)
7. Choong, M.K., Charbit, M., Yan, H.: Autoregressive-model-based missing value estimation for DNA microarray time series data. IEEE Trans. Inf. Technol. Biomed. **13**(1), 131–137 (2009)
8. Lin, W.-C., Ke, S.-W., Tsai, C.-F.: CANN: an intrusion detection system based on combining cluster centers and nearest neighbors. Knowl. Based Syst. **78**, 13–21 (2015)
9. UshaRani, Y., Sammulal, P.: A novel approach for imputation of missing attribute values for efficient mining of medical datasets - class based cluster approach. Rev. Tec. Fac. Ing. Univ. Del Zulia **39**(2), 184–195 (2016)
10. UshaRani, Y., Sammulal, P.: An innovative imputation and classification approach for accurate disease prediction. Int. J. Comput. Sci. Inf. Secur. (IJCSIS) **14**(S1), 23–31 (2016). Special issue on "Computing Applications and Data Mining"
11. UshaRani, Y., Sammulal, P.: A novel approach for imputation of missing values for mining medical datasets. In: 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Madurai, pp. 1–8 (2015). doi:10.1109/ICCIC.2015.7435816
12. http://www.is.umk.pl/projects/datasets.html

Intelligent Image Processing

Implementation of Particle Filters for Mobile Robot Localization

Michal Růžička[✉], Petr Mašek, and Stanislav Věchet

Faculty of Mechanical Engineering, Institute of Automation and Computer Science, Brno University of Technology, Technická 2896/2616 69, Brno, Czech Republic
{y110384, y70232}@stud.fme.vutbr.cz,
vechet.s@fme.vutbr.cz

Abstract. This paper deals with mobile robot localization purpose. The presented solution is designed for indoor environment only. GPS navigation system cannot be used in environment inside of buildings. Alternative methods have to be used for this purpose. The mobile robot localization is essential part of autonomous mobile robotics. Mobile robot localization together with odometry is necessary for mobile robot navigation. Presented paper contains explanation of localization approach, which is based on probabilistic method. Next part of this paper is description of experimental odometry method, which is based on computer vision.

Keywords: Navigation · Localization · Odometry · Computer vision · Particle filters · Mobile robot · Phase correlation

1 Introduction

This paper deals with indoor localization of mobile robot. Mobile robot localization is one of the main parts of mobile robotics. Mobile robots navigation system is composed from two essential parts. The first one is absolute localization method, which will be described in this paper. This part is main topic of this paper. The second one is odometry. The commonly used type of odometry is based on drives encoders measurement, where is captured a travelled distance of robots wheels. The encoders measure each revolution of robots wheel, where the travelled distance depends on wheels diameter. Travelled distance of couple mobile robot wheels defines travelled trajectory of mobile robot, which depends on type of vehicle chassis. The main problem of odometry is continuously cumulating random error of measurement. This paper presents experimental odometry approach, which is based on computer vision.

This paper deals with description of specific parts of localization and mapping system. Figure 1 contains schematic of localization and mapping system, where the inputs to system are image data, distance measured by range finder, odometry data and navigation data [4, 5]. Image data are used for mapping the environment. During mobile robot movement is continuously created the map of environment. Mapping is based on phase correlation [1–3, 7] and stitching method. Visual odometry has two inputs the first one is data from range finder, which measure camera distance to ceiling. The second one

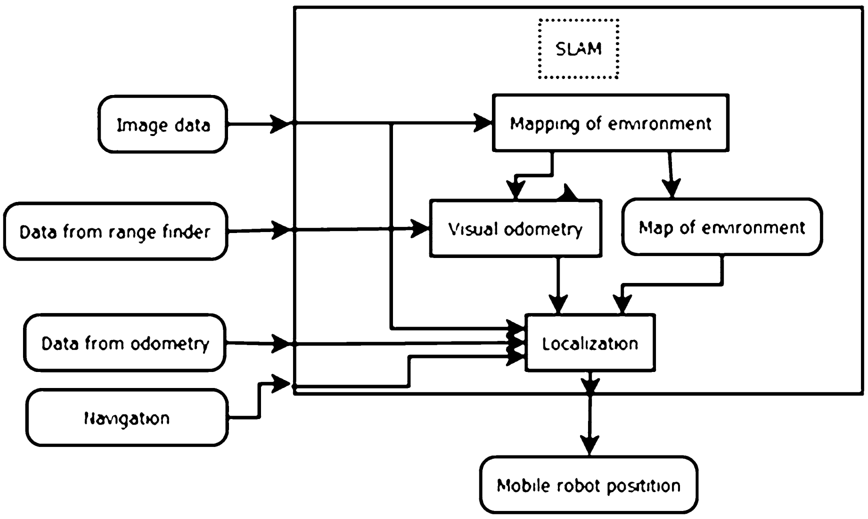


Fig. 1. Mapping and localization schematic.

is data from mapping module, especially global shift from phase correlation. Global shift is in pixels, which is recalculated to millimetres thanks to knowledge distance to ceiling and camera parameters. This is output of visual odometry part. The localization part has a few inputs. The first one are image data, which are used by particle filters for re-sampling. The second one is measured step from visual odometry, which initialize location estimation. Next one is continuously created map of environment. The next one is navigation data [4, 5], which determine vehicle turning angle in each step of measurement. The last one is data from odometry, which initialize location estimation as in the case visual odometry. Both odometry are folded together. The output of localization part is mobile robot position in the map of environment. The part visual odometry and part localization will be described in this paper.

This paper is organized as follows: The Sect. 2 describes experimental odometry approach. The Sect. 3 contains explanation of localization method, which is based on probabilistic approach. Practical experiments are described in the Sect. 4. The last section is reserved to conclusions and future works.

2 Visual Odometry

Let's focus to experimental odometrical approach. The main odometrical sensor is camera against to commonly used approach, where main sensors are encoders. Camera is positioned perpendicular to ceiling. Presented odometry uses mapping part, where global shift is input to visual odometry. The global shift is computed by phase correlation [1–3, 7], which is part of mapping module. The next input is distance measured by range finder. Global shift is in pixel units, which have to be recalculated to real units of distance. Camera diagonal angle of view can be used for this purpose. This can be

Table 1. Variables description in (1).

Value name	Description	Unit
Shift	Real shift	Millimetres
Dist	Distance from camera to ceiling	Millimetres
Angle	Camera diagonal angle of view	Degrees
diagLengthPix	Number of pixels on diagonal of image matrix	Pixels
globalShift	Global shift from mapping part	Pixels

experimentally determined, or find out in camera datasheet, if it is written there. Converting formula is (1). Table 1 contains legend for (1). This part of the whole system is not important, but the current testing platform does not contain odometrical systems. This will be described in a second practical experiment. This part was made to replace missing odometry into current testing platform, despite that will be considered to final solution.

$$shift = |(dist * \tan(\text{angle})/\text{diagLengthPix}) * \text{globalShift}| \tag{1}$$

3 Localization Method

Localization is based on probabilistic method [6], especially on particle filters. Simplified schematic of localization algorithm is on Fig. 1. Localization part has a few inputs: Image data, data from odometry, data from visual odometry, navigation data and continuously creating map of environment.

Weighted mean is calculated from both odometry. When is measured the previously determined step, then position estimation is started.

Sample generating is equipped by motion model of mobile robot. This depends on type of chassis. Differential chassis was chosen in this case. Motion model is on (2) and (3).

$$x = d \cdot \cos(\varphi) \tag{2}$$

$$y = d \cdot \sin(\varphi) \tag{3}$$

There x and y represents robot position in environment. Variable d is traveled distance per step. Variable φ represents orientation of mobile robot. Sample generating uses that motion model as was written previously. Samples are generated by formulas (4) and (5).

$$x_i = d_N \cdot \cos(\varphi_N + \varphi) \tag{4}$$

$$y_i = d_N \cdot \sin(\varphi_N + \varphi) \tag{5}$$

Where x_i and y_i represents position of sample (also called particle) with index i. Variables d_N and φ_N was randomly generated by distribution function with parameters: μ = d, $\sigma_d = 1$ and φ_N for μ = φ, $\sigma_\varphi = 4$. Variable φ is current turning angle of chassis in this step. This angle was passed from navigation. Navigation [4, 5] contains the next necessary information, whether mobile robot going forward or backward. This is integrated in localization as well.

This is all samples are placed into the map of environment. For each sample is extracted the area around the sample position. Dimensions are the same as image data dimensions. Currently captured image data are compared with each particle area. Phase correlation [1–3, 7] was used for this comparison. This is not a same phase correlation as for mapping task, but this is simplified version of this algorithm, which is able to register shift only.

The next step is weights calculation. Shifts are base for this purpose. The general rude: smaller shift, the higher weight.

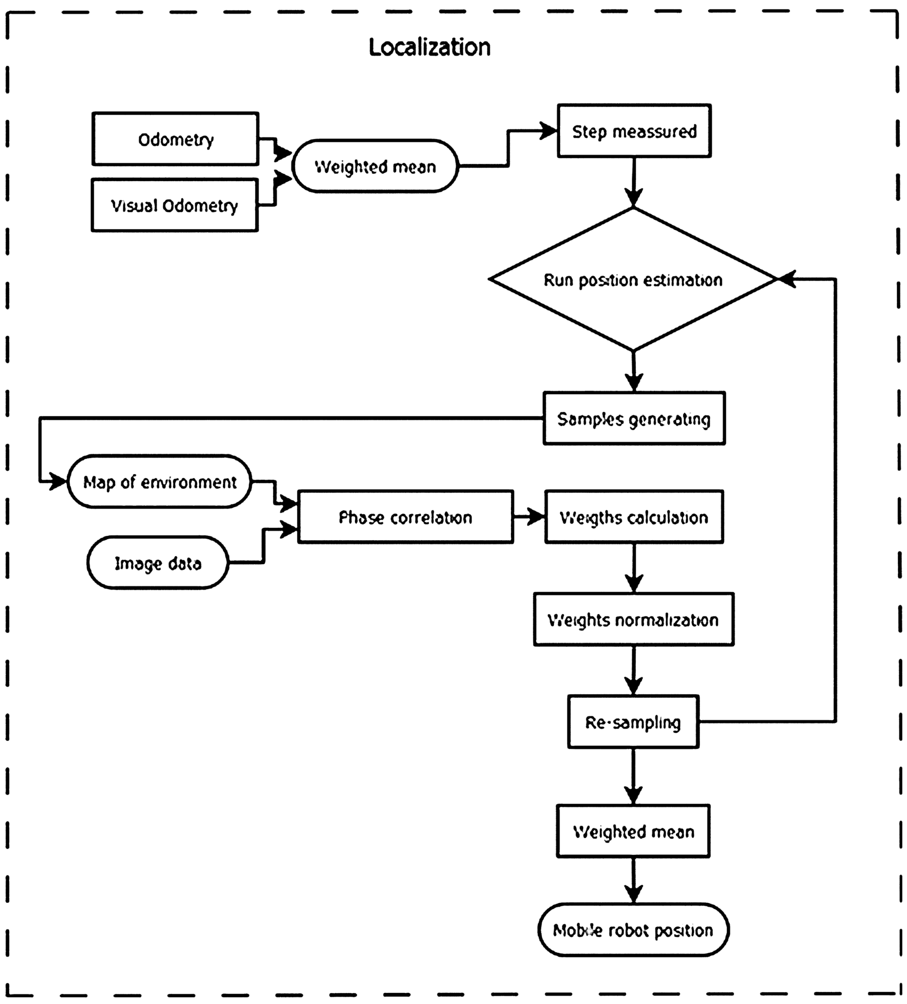


Fig. 2. Localization schematic.

After that all weights have to be normalized, this means all weights have to be converted to closed interval <0, 1>.

Such data are re-sampled. All chosen samples are preserved, otherwise not considered for next algorithm steps.

Robot position is calculated from weighted mean of positions re-sampled samples (Fig. 2).

4 Practical Experiments

This section is reserved to practical experiments, which demonstrate functionality of presented solution. In all experiments were used 5 samples on the start of algorithm. 25 samples were used during position estimation. The higher samples count, means higher computational demands. All experiments were performed on hardware with following hardware specifications:

1. CPU: Intel i7
2. Ram: 4 GB
3. Camera: Logitech c930

4.1 Simulation

This experiment is deals with verifying of visual odometry and localization functionality. There were not used real data.

The experiment visualization is on Fig. 3. Left side positioned blue circle represents starting position of mobile robot. Green circles are mobile robot estimated positions. Right side positioned blue circle is mobile robot current position. Red dots represent samples, which were not considered to next steps localization algorithm.

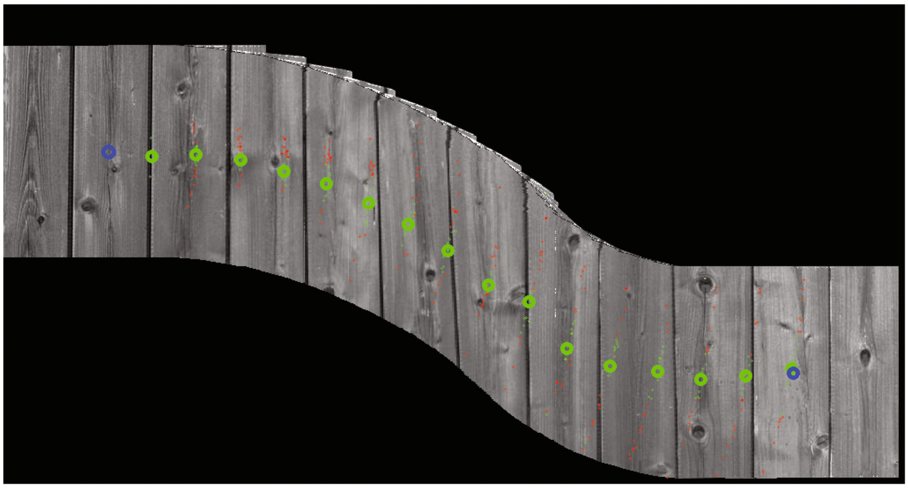


Fig. 3. Simulation of localization.

On the other hand green dots represent re-sampled samples. There is evident, that visual odometry works well. Position estimation works well to. Let's move to experiment with real data.

4.2 Localization with Real Data in Online Mode

This experiment was performed on real mobile robot in online mode. Localization ran in real time mode. The mobile robot [9] does not contain own odometry system. Visual odometry was used only in this case. The experiment visualization is on Fig. 4. The legend is the same as previous case, except current mobile robot position. The measurement step was 0.6 m. There is evident localization works well. Position calculating time takes 0.5 s. This fact applies only for 25 samples, which were used in this case. Mapping frame rate is approximately 15 frames per second.

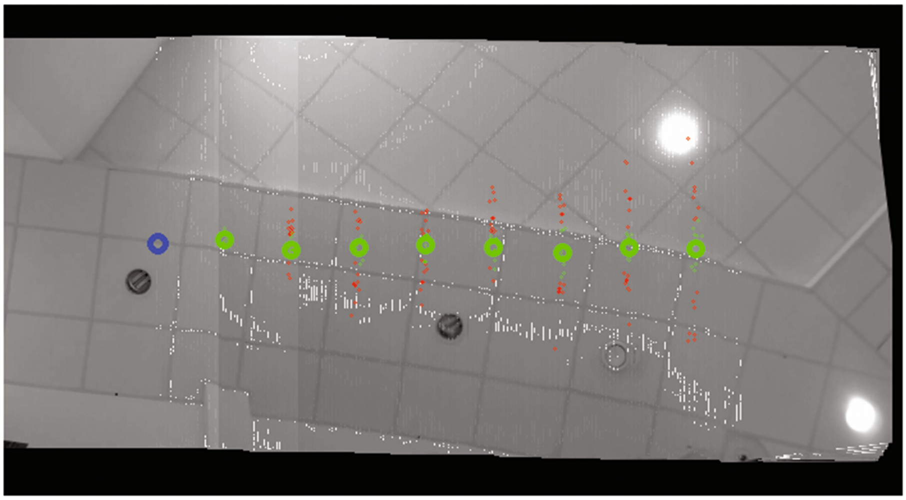


Fig. 4. Real time localization.

5 Conclusions

This paper presented mobile robot two parts of mapping and localization system for mobile robots, which operate inside of buildings. The main advantage of this solution is no need to modify the environment, where robot operates. The visual odometry and localization system were designed and tested. Presented solution was verified by couple of practical experiments. The localization is based on probabilistic approach, which seems appropriate for this purpose. The system is still in development phase. Many problems have to be resolved. Source code is completely written in C++ language with the support of OpenCV libraries.

The future works will be intensive due to system is not able to resolve closure loop [8] problem and problem of kidnapped robot yet. We are preparing mobile robot, which contains odometry system based on encoders. Micro PC will be main control unit for robots low level systems. The image data together with odometrical data will be streamed to server, which will be equipped by strong computing power to reach real time localization.

Acknowledgement. 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. **706**(2), 1605–1608 (2009)
2. Berjak, J., Druckmüller, M.: Automatic analysis and recognition of moving objects in the picture by method of phase correlation, p. 35 (2004)
3. 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), vol. 16 in MENDEL, pp. 578–583. Brno University of Technology, VUT Press, Brno (2010)
4. Vechet, S., Ondrousek, V.: Motion planning of autonomous mobile robot in highly populated dynamic environment. In: Jabłoński, R., Březina, T., (eds.) Mechatronics, Warsaw, vol. 9, pp. 453–461. Springer, Heidelberg (2011)
5. 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), vol. 17 in MENDEL, pp. 546–551. Brno University of Technology, VUT Press, Brno (2011)
6. Krejsa, J., Vechet, S.: Infrared beacons based localization of mobile robot. Elektronika Ir Elektrotechnika, pp. 17–22, Kaunas (2012). doi:10.5755/j01.eee.117.1.1046
7. Druckmüllerová, H.: Phase-correlation based image registration, 100 p. Brno University of Technology, Faculty of Mechanical Engineering, Brno, Supervisor Mgr. Jana Procházková, Ph.D. (2010)
8. Vechet, S., Krejsa, J., Houska, P.: The enhancement of PCSM method by motion history analysis. In: Recent Advances in Mechatronics, pp. 107–110 (2007). WOS:000251017700022, doi:10.1007/978-3-540-73956-2_22
9. Hrbacek, J., Ripel, T., Krejsa, J.: Ackermann mobile robot chassis with independent rear wheel drives. In: Proceedings of 14th International Power Electronics and Motion Control Conference (Epe-Pemc 2010) (2010). WOS:000319521600268, doi:10.1109/Epepemc.2010.5606853

Direct Point Cloud Visualization Using T-spline with Edge Detection

Jana Prochazkova(✉) and Jiri Kratochvil

Department of Mathematics, Faculty of Mechanical Engineering,
Brno University of Technology, Technicka 2, Brno, Czech Republic
prochazkova.j@fme.vutbr.cz, y174169@stud.fme.vutbr.cz

Abstract. This article presents a hybrid method for a processing of a cloud point. Proposed method is suitable for reverse engineering where the need of precise model representation is essential. Our method is composed of mathematical representation using T-spline surfaces and edge extraction using k-neighborhood and Gauss mapping. The advantages of this method that we are able to find mathematical expression of the model where modification of parameters expresses the edges directly.

Keywords: Point cloud · Reverse engineering · Edge detection · T-spline · T-mesh

1 Introduction

Widely spread laser scanning technology produces multiple 3D point clouds with high-density and accuracy; therefore, there is a need for generation of accurate models with correct topology. The point clouds can describe even relatively small objects (e.g. for reverse engineering), as well as extensive urban areas for spatial analysis obtained by Airborne Laser Scanning (ALS). The engineering technology deals especially with the first group of point clouds, models of a spare parts etc.

The point cloud processing deals with problems that are very similar to a common image processing: e.g. How to find specific object within the point cloud or image? In both cases, it is necessary to determine an algorithm that is able to decide whether the data contains some given object. In the area of image processing, soft computing methods are frequently used. Even for the point cloud processing, the soft computing methods are very promising. Nonetheless, after such required object is detected, we need to reconstruct it's precise shape in many cases. Our article is focused on this issue. Proposed method presents a second step in a work-flow that allows to reconstruct a precise shape of given object in a point cloud. It assumes that the object was found by any object detection algorithm and takes this particular part of the point cloud as an input. The output is precise mathematical representation that describes the object and enables the possibility of e.g. spatial analysis. Therefore, our presented method is tightly connected with the soft-computing based object detection. One step without the other can not produce results suitable for many different applications.

The process of 3D model creation starts with data collection. There are multiple ways to collect required 3D data but there are two of the most common method groups: contact methods and non-contact methods. Digitizing is an example of contact method where the 3D scanner probe the subject through physical touch. The advantage of contact scanners is their precision thus they are widely used in manufacturing. However, this method is not suitable for arbitrary object due the fact that the scanning itself can modify or damage the object. The other disadvantage is the speed – contact scanning is quite slower in comparison with the other methods.

The laser scanning belongs to the second group called non-contact methods (ultrasound, x-ray measurements, etc.). There are two main types of scanning principles: time-of-flight and optical triangulation. During the time-of-light scanning the scanner emits a focused pulse of laser light and waits for its to return to a sensor. The laser range finder finds the distance from the surface by measurement of round-trip time of a given light pulse. The triangulation needs the camera in addition. The laser emitter, camera and laser dot on the object form a triangle. The distance between camera and emitter is known and its two adjacent angles too; hence, the computation of the laser dot position is simple (details in the paper [4]). The output of laser scanning part is the set of (x, y, z) coordinates called point cloud.

The next phase is a processing of collected data. We distinguish two main applications: digital modeling and reverse engineering methods. Digital modeling is based on border representation (B-rep), primitive models (the model consists of known geometric solids) or surface models (e.g. NURBS). Reverse engineering methods are time consuming and creates precise models with edges. We propose a novel hybrid method utilizing mathematical T-spline representation (digital modeling) and feature detection (reverse engineering part).

The paper is organized as follows: Sect. 2 describes the theoretical background of spline and T-spline. In Sect. 3 we explain the partial steps of the algorithm: (1) T-mesh topology, (2) Edge detection algorithm, (3) Interpolation and visualization.

1.1 Prior Work

Wide area of literature covers digital modeling and reverse engineering methods of collected data processing. First, we describe a digital modeling methods – B-Rep, primitive models and surface models. Delaunay triangulation and its modification [13, 19, 23] are one of the most spread used algorithms in border representation (B-Rep). Nevertheless, we have to mention new area of research – Building Information Models (BIM) that are also based on this type of representation [18, 26].

The primitive models are also well known in literature. The work [24] uses plane, cylinder and sphere as basic object models. The constrains (parallel distance, distance) and cost functions are described to detect correct topology.

In the work [12] authors use 3D-primitives such as planes, cylinders, spheres or cones to describe regular roof sections, and they are combined with mesh-patches that represent irregular roof components. By way of contrast, the article [34] presents the algorithm consists of segmented surfaces, their estimated quadric models and corresponding surface classification. Interesting method for automatic modeling and recognition of 3D industrial site point clouds is presented in [17]. The algorithm is based on knowledge that industrial point cloud is a congregation of pipes, planes and objects. Hence, the task can be divided into three separate sub-problems: pipe modeling, plane classification, and object recognition.

The spline reconstruction based on simplified surface that captures the topological structure is described in [3]. Interesting approach is the fusion of 3D model and uncalibrated stereo reconstruction [11] and the work [25] presents object reconstruction using the method of moments.

In the field of object visualization the edges provide significant visual information about the result shape and they accent the visual perception. The preservation of sharp features is essential for reverse engineering with surface reconstruction, simplification and segmentation and in building and roof reconstructions, see [1,28,29,33] for more details.

We have only two possible ways of edge detection. Firstly, there are polygonal methods that are based on a mesh utilizing techniques. Most of these methods use Delaunay triangulation as pre-processing step and subsequently, different detection methods are processed [16,27,32].

Secondly, the point based methods characteristic is the lack of knowledge concerning topology (normals, connectivity information), i.e. it works directly with point cloud without any polygonal pre-processing. Detailed overview of these methods is in article [31]. The authors of this article construct Gauss map clustering on local neighborhoods in order to discard all points which are unlikely to belong to a sharp feature.

An algorithm to extract closed sharp feature lines using first order segmentation to extract candidate feature points and process them as a graph to recover the sharp feature lines is presented in [5]. An unconventional combination of the edge data from a point cloud of an object and its corresponding digital images is the main idea of the paper [30].

The final recovery of the sharp feature lines is one of the key problems. Some authors [9,10] use the extrema triangles to build a set of sharp feature edges or direct modification of vertices in triangles [2]. The article [8] describes the recovery process in details, it consists of the smoothing through spline fitting and grouping weights along crease strip. Problematic are the triangles, that interconnected different sides of the crease, and the construction of corners. In our approach we propose appropriate settings of control net to correct visualization in Sect. 3.4.

2 NURBS, T-spline

T-spline are the generalization of Non-Uniform Rational B-splines (NURBS). NURBS have become the integral part in many common application and the theory is well described in details e.g. in [20]. NURBS curves are based on a B-spline basis functions that are connected to the sequence of control points and knot vectors. NURBS surface is the tensor product of NURBS curves (theoretically described in our previous work [15]).

Let $U = (u_0, \ldots, u_m)$ be a knot vector ($u_i \leq u_{i+1}$, $i = 0, \ldots, m-1$). Than the i-th B-spline basis function of degree p is defined recurrently as:

$$N_{i,0} = \begin{cases} 1 \text{ if } \ u_i \leq u < u_{i+1} \\ 0 \text{ otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{1}$$

Let $\{P_i\}$ are the control points and $\{w_i\}$, $i = 0, \ldots, n$ are the weights, and the $N_{i,p}(u)$ are $p-$th degree B-spline basis functions. Then the NURBS curve is defined as:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{2}$$

NURBS surface is defined as a tensor product of NURBS curves. Every NURBS topology needs the regular $k+1 \times l+1$ control points net P_{ij} with weights w_{ij}, $i = 0, 1, \ldots, k$, $j = 0, 1, \ldots, l$, knot vectors $U = (u_0, \ldots, u_m)$, $V = (v_0, \ldots, v_n)$ and degrees p, q. Then NURBS surface can be expressed as:

$$S(u,v) = \frac{\sum_{i=0}^{k} \sum_{j=0}^{l} N_{i,p}(u) N_{j,q}(v) w_{ij} \mathbf{P}_{ij}}{\sum_{i=0}^{k} \sum_{j=0}^{l} N_{i,p}(u) N_{j,q}(v) w_{ij}} \tag{3}$$

If the weights of all points are equal to one, then we can rewrite Eq. (3) as:

$$S(u,v) = \sum_{i=0}^{k} \sum_{j=0}^{l} N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{ij} \tag{4}$$

The first step in the formation of the T-spline is the abandonment of the regular network; however, the advantages as the local modification scheme or control point insertion remain preserved. The general idea of T-spline was firstly published by Sederberg [21]. Moreover, the algorithm T-spline simplification can be used to reduce the number of control points in the control net [22].

The main computation principle of surface points is the same as NURBS. The surface point is computed with the combination of the control points and B-spline basis functions, only the knot vectors are not global but connected unambiguously with every control point. The derivation of this knot vectors is clearly described in next section about T-mesh.

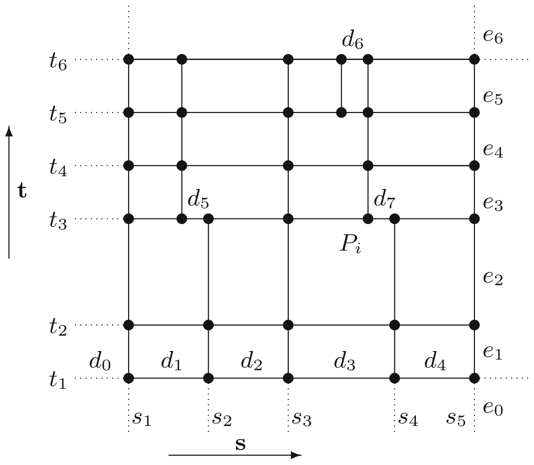


Fig. 1. T-mesh–knot coordinates and knot intervals

2.1 T-mesh

The control grid for a T-spline surface is called T-mesh. The Fig. 1 shows the example of T-mesh in (s,t) parameter space. Let s_i and t_i denote the knot coordinates and d_i and e_i denote knot intervals. (Knot interval is difference between two adjacent knot coordinate). T-mesh often contains T-junctions. This special type is a vertex which is shared by one s-edge and two t-edges, or by one t-edge and two s-edges. In Fig. 1 T-junction is the point labeled P. Every grid point has its own pair of knot vectors and their derivation is simple (explain in [21]). Point P coordinates are $(s_4 - d_7, t_3)$ and vectors are detected in parameter space $R(\alpha) = (s_4 - d_7, \alpha t_3)$ and $R(\beta) = (\beta(s_4 - d_7), t_3)$. Thus, for point P_i, $\mathbf{s}_i = (s_2, s_3, s_4 - d_7, s_4, s_5)$, $\mathbf{t}_i = (t_1, t_2, t_3, t_4, t_5)$.

2.2 T-spline Surface

T-spline surface of degree p is defined by control points P_i, $i = 1, \ldots, n$. Every point is connected with two knot vectors of length $2p-1$ derived from T-mesh as was shown in previous section. In our work, we use degree $p = 3$. T-spline surfaces for arbitrary degree are overviewed in paper [7]. It is also possible to add weights to the control points to make weighted T-spline surface [14].
The mathematical expression of T-spline is:

$$P(s,t) = \sum_{i=1}^{n} P_i B_i^3(s,t). \tag{5}$$

The basis functions $B_i^3(s,t)$ are given by

$$B_i^3(s,t) = N[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}](s) N[t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}](t).$$

where $N[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}](s)$ is B-spline basis function associated with the knot vector

$$s_i = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$$

and $N[t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}](t)$ is associated with the knot vector

$$t_i = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}].$$

3 Data Processing

The input to our algorithm is a point cloud obtained by laser scanner. Our method can be described in four steps. (1) Find the T-mesh topology using y-cube method, (2) detection of edges, (3) correction of T-mesh due to detected edges, (4) result surface computation.

3.1 T-mesh Topology

Following part outlines the key issues of the T-mesh classification. We propose universal method that is able to determine the appropriate T-mesh due to the input point cloud. Further, the appropriate knot vectors are computed with centripetal method (e.g. in [6]).

We start with the computation of the box size that depends on the size of scanning area and on the density of the measured points. Then we shift the unit cubes in y direction and divide the point cloud (Fig. 2). Subsequently, we use the quick sort algorithm due to size of x coordinate. (We use library STL.) On Fig. 3 is a simple point cloud and the application of method y-cube. Lines make T-mesh topology in first direction of knot vector **s** and the values of this line knot vector are computed with centripetal method with averaging.
Let d be the total chord length for i-th box:

$$d^i = \sum_{k=0}^{n_i} \sqrt{|\mathbf{P}_k^i - \mathbf{P}_{k-1}^i|},$$

where n_i is the number of the points $\mathbf{P}_k^i$ in i-th box. Then

$$\overline{s}_0^i = 0 \quad \overline{s}_n^i = 1$$

$$\overline{s}_k^i = \overline{s}_{k-1}^i + \frac{\sqrt{|\mathbf{P}_k^i - \mathbf{P}_{k-1}^i|}^{\,i}}{d} \quad k = 1, \ldots, n-1. \tag{6}$$

We applied the technique of averaging on the values $\overline{s}_k$ and we get (p is a degree):

$$s_0^i = \ldots = s_p^i = 0 \quad s_{m-p}^i = \ldots = s_{p+n_i}^i = 1,$$

$$s_{j+p}^i = \frac{1}{p} \sum_{i=j}^{j+p-1} \overline{s}_j \quad j = 1, \ldots, n_i - p. \tag{7}$$

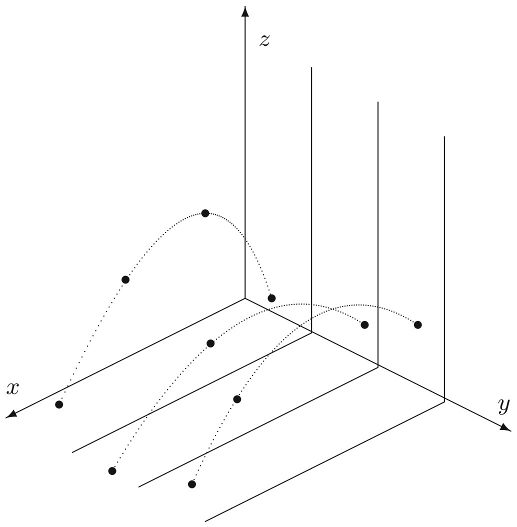


Fig. 2. Method y-cube

We set the second knot vector **t** as uniform ($t_{i+1} - t_i$ is constant). The usage of non-uniform vector will make the algorithm time-consuming, computational demanding and unsuitable for other computations. We have to note that first and last $p + 1$ members are equal one to guarantee that the surface interpolates the corner points.

3.2 Edge Detection Algorithm

In this section, we present a simple algorithm for edge detection within the point cloud. The algorithm consists of three steps. Firstly, the triangulated irregular network (TIN) is needed. We do not use the Delaunay triangulation algorithm because we know the correct topology from previous step. Secondly, the non edge points are removed by flatness test. And lastly, the edge points are detected by implementing the Gauss map.

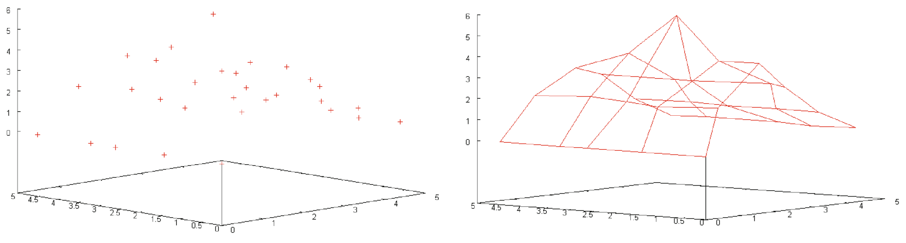


Fig. 3. Cloud of points and its T-mesh

The flatness test [31] is based on deviation of normal vectors of each point within the given point cloud. Let N_P be the neighborhood of arbitrary point P containing k nearest neighbors and let the I_P be the set of all indexes ($I_p = 1, 2, \ldots, k$). We make a set T of all possible triangles with vertex P and two neighborhood points from the set N_P. The mean normal vector n of one of these triangles is given by:

$$n_{kl} = PP_k \times PP_l \tag{8}$$

where $P_k, P_l \in N_p$ and $k \neq l$, $k, l \in I_P$. Points with the deviation of normal vectors within the given range therefore lie on a flat surface and thus are removed. Remaining points are selected for further processing with Gauss map. The discrete Gauss map of the neighborhood of P is defined as the mapping of set T onto the unit sphere centered at point P. We simply determine the points S_{kl} on the sphere by the mapping:

$$S_{kl} = P + \frac{n_{kl}}{|n_{kl}|} \tag{9}$$

This projection creates a set of point clusters on the sphere surface. By estimating the number of these clusters we can decide whether the point P is the edge point or not. Possible number of clusters with explanation of point P: 1–flat surface point, 2–edge point, 3 or more–possibly the corner point.

3.3 T-mesh Topology Correction

The algorithm described in previous section detects the edges in the point cloud. But we want to model the result surface with this edges visible. We have to use the property of basic B-spline function $B_i^p(t)$.

Theorem 1. *All derivatives of $N_{i,p}(t)$ exists in the interior of a knot span. At a knot $N_{i,p}(t)$ is $p - k$ differentiable, where k is the multiplicity of the knot.*

The proof is by induction on degree p [20].

Corollary 1. *If $p = k$, where p is degree of B-spline function $N_{i,p}(t)$ and k is the multiplicity of the knot, then at the knot $N_{i,p}(t)$ is discontinuous.*

Hence we attach special type of knot vector to edge points. We use T-spline surface degree equal to three. Therefore, multiplicity of a knot equal to three guarantees the continuity equal to zero, so that the edge will be visible. For the circled points from T-mesh on Fig. 4 are $\mathbf{s}$ knot vectors

$$(d_1, d_1 + d_2, d_1 + d_2, d_1 + d_2, d_1 + d_2 + d_3).$$

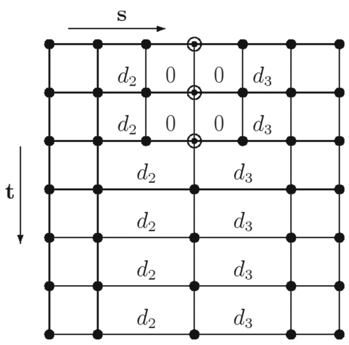


Fig. 4. The modification of control net with edge points (in circles)

3.4 T-spline Visualization

Input: control points $\mathbf{P}_i$ with corresponding knot vectors–$\mathbf{s}, \mathbf{t}$, surface degree–3, parameters u, v.
Output: T-spline surface point $S(s, t)$.

1. We find the influence points for parameters s, t, i.e. points, which knot vectors contain parameters s, t.
2. For these points we computer basic polynomials $N^3_{0i}(s), N^3_{0i}(t)$.
3. We figure out the expression $\sum_i \mathbf{P}_i N^3_{0i}(s), N^3_{0i}(t)$, which intend the result point.

We do not use de Boor algorithm [20] to compute expressions $N^3_i(s), N^3_i(t)$. We utilize a direct computation because all of polynomials are always same. The basic polynomials for T-spline degree three are given by Eq. (10).

$$N[\mathbf{t}_i](t) = \begin{cases} \frac{(t-t_0)^3}{(t_1-t_0)(t_3-t_0)(t_2-t_0)} \\ \text{for} \quad t \in \langle t_0, t_1) \\ \frac{(t-t_0)^2(t_2-t)}{(t_2-t_1)(t_3-t_1)(t_2-t_0)} + \frac{(t_3-t)(t-t_0)(t-t_1)}{(t_2-t_1)(t_3-t_1)(t_3-t_0)} + \frac{(t_4-t)(t-t_1)^2}{(t_2-t_1)(t_4-t_1)(t_3-t_1)} \\ \text{for} \quad t \in \langle t_1, t_2) \\ \frac{(t-t_0)(t_3-t)^2}{(t_3-t_2)(t_3-t_1)(t_3-t_0)} + \frac{(t_4-t)(t_3-t)(t-t_1)}{(t_3-t_2)(t_4-t_1)(t_3-t_1)} + \frac{(t_4-t)^2(t-t_2)}{(t_3-t_2)(t_4-t_2)(t_4-t_1)} \\ \text{for} \quad t \in \langle t_2, t_3) \\ \frac{(t_4-t)^3}{(t_4-t_3)(t_4-t_2)(t_4-t_1)} \\ \text{for} \quad t \in \langle t_3, t_4) \\ 0 \quad \text{otherwise} \end{cases} \tag{10}$$

The result visualization of the arbitrary point cloud with edge is in Fig. 5. In Fig. 6 the part of prism is visualized using T-spline control net and suitable knot vectors. For illustration we use data without noise.

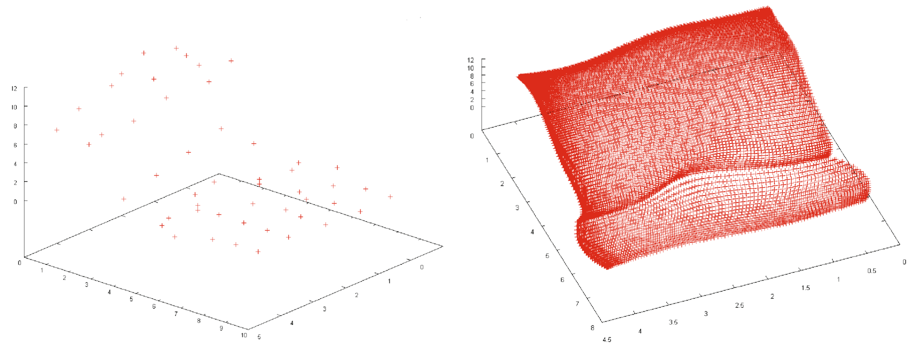


Fig. 5. General cloud of points and its visualization with visible edge (gnuplot)

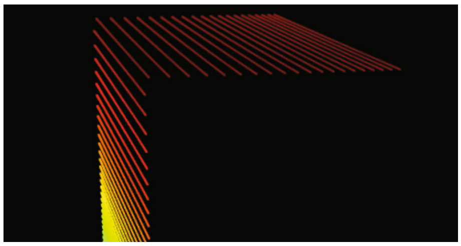


Fig. 6. Visualization of the prism with visible edge (lidarview.com)

4 Conclusion

We suggest semi-automatic algorithm to classify T-mesh topology to obtain correct mathematical representation of arbitrary point cloud. Then the edge detection (based on k-neighborhood and Gauss mapping) that is able to extract the edges of the object, is described. In the last part we present the mathematical description where the change of the knot vectors in T-mesh topology makes the edges clearly visible.

Our future work will be focused on the improvement a computational optimization. The parameter for classification of T-mesh topology is suitable for sparse data. We would like to add the algorithm of T-spline simplification that can reduce the number of necessary points significantly and our method will be suitable for dense data too. Also the result shape of the edges often contains noise and appropriate filters will be suitable to eliminate it. Nevertheless, our approach creates a basis for development of stable method for the purpose of reverse engineering.

Acknowledgments. This work was supported by Project LO1202 by financial means from the Ministry of Education, Youth and Sports under the National Sustainability Programme I.

References

1. Alharthy, A., Bethel, J.: Heuristic filtering and 3D feature extraction from LIDAR data. In: ISPRS Commission III, Symposium 2002, pp. 23–28 (2002)
2. Attene, M., Falcidieno, B., Rossignac, J., Spagnuolo, M.: Sharpen bend: recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. IEEE Trans. Visual Comput. Graph. **11**(2), 181–192 (2005)
3. Baining, G.: Surface reconstruction: from points to splines. Comput. Aided Des. **29**(4), 269–277 (1997)
4. Curless, B.: From range scans to 3D models. SIGGRAPH Comput. Graph. **33**(4), 38–41 (1999)
5. Demarsin, K., Vandestraeten, D., Volodine, T., Roose, D.: Detection of closed sharp edges in point cloud using normal estimation and graph theory. Comput. Aided Des. **39**, 276–283 (2007)
6. Farin, G.: Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, 4th edn. Academic Press, New York (1997)
7. Finnigan, G.T.: Arbitrary Degree T-Splines. All theses and Dissertations. Paper 1431 (2008)
8. Gumhold, S., Wang, X., Macleod, R.: Feature extraction from point clouds. In: Gumhold, S. (ed.) Proceedings of the 10th International Meshing Roundtable, pp. 293–305. Sandia National Laboratory (2001)
9. Hildebrand, K., Polthier, K., Wardetzky, M.: Smooth feature lines on surface meshes. In: Proceedings of the Third Eurographics Symposium on Geometry Processing, SGP 2005, Article 85. Aire-la-Ville, Switzerland (2005)
10. Hubeli, A., Gross, M.: Multiresolution feature extraction for unstructured meshes. In: Proceedings of IEEE Visualization, pp. 287–294 (2001)
11. Klecka, J., Horak, K.: Fusion of 3D model and uncalibrated stereo reconstruction. In: Matoušek, R. (ed.) Mendel 2015. AISC, vol. 378, pp. 343–351. Springer, Cham (2015). doi:10.1007/978-3-319-19824-8_28
12. Lafarge, F., Mallet, C.: Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation. Int. J. Comput. Vis. **99**(1), 69–85 (2012)
13. Lee, D.T., Schachter, B.J.: Two algorithms for constructing a Delaunay triangulation. Int. J. Comput. Inform. Sci. **9**(3), 219–242 (1980)
14. Liu, L., Zhang, Y.J., Wei, X.: Weighted T-splines with application in reparameterizing trimmed NURBS surfaces. Comput. Methods Appl. Mech. Eng. **295**, 108–126 (2015)
15. Martisek, D., Prochazkova, J.: Relation between algebraic and geometric view on NURBS tensor surfaces. Appl. Math. **5**, 419–430 (2010)
16. Ohtake, Y., Belyaev, A.: Automatic detection of geodesic ridges and ravines on polygonal surfaces. J. Three Dimensional Images **15**(1), 127–132 (2001)
17. Pang, G., Qiu, R., Huang, J., You, S., Neumann, U.: Automatic 3D industrial point cloud modeling and recognition. In: Machine Vision Applications (MVA), pp. 22–25 (2015)
18. Patraucean, V., Armeni, I., Nahangi, M., Yeung, J., Brilakis, I., Haas, C.: State of research in automatic as-built modelling. Adv. Eng. Inform. **29**(2), 162–171 (2015)
19. Peter, S., Drysdale, R.L.S.: A comparison of sequential Delaunay triangulation algorithms. In: Peter, S. (ed.) Proceedings of the 11th Annual Symposium on Computational Geometry, SCG 1995, pp. 61–70. ACM, New York (1995)
20. Piegl, L., Tiller, W.: The NURBS Book. Springer, Berlin (2002)

21. Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A.: T-splines and T-NURCCS. ACM Trans. Graph. **22**(3), 477–483 (2003)
22. Sederberg, T.W., Zheng, J., Cardon, D.L., Lyche, T.: T-splines simplification and local refinement. ACM Trans. Graph. **23**(3), 276–283 (2004)
23. Shewchuk, J.R.: Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In: Lin, M.C., Manocha, D. (eds.) WACG 1996. LNCS, vol. 1148, pp. 203–222. Springer, Heidelberg (1996). doi:10.1007/BFb0014497
24. Somani, N., Perzylo, A., Cai, C., Rickert, M., Knoll, A.: Object detection using boundary representations of primitive shapes. In: IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 108–113 (2015)
25. 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, Brno, pp. 241–248 (2014). ISSN: 1803–3814
26. Steder, B., Rusu, R.B., Konolige, K., Burgard, W.: Point feature extraction on 3D range scans taking into account object boundaries. In: Robotics and Automation (ICRA), pp. 2601–2608 (2011)
27. Stylianou, G., Farin, G.: Crest lines extraction from 3D triangulated meshes. In: Hierarchical and Geometrical Methods in Scientific Visualization, pp. 269–281 (2003)
28. Verma, V., Kumar, R., Hsu, S.: 3D building detection and modeling from aerial LIDAR data. IEEE Comput. Vis. Pattern Recogn. **2**, 2213–2220 (2006)
29. Vosselman, V.: Building reconstruction using planar faces in very hight density data. In: International Archives of Photogrammetry and Remote Sensing, pp. 87–92 (1999)
30. Wang, Y., Ewert, D., Schilberg, D., Jeschke, S.: Edge extraction by merging 3D point cloud and 2D image data. In: Emerging Technologies for a Smarter World (CEWIT), pp. 1–6 (2013)
31. Weber, C., Hahmann, S., Hagen, H.: 2010. Sharp feature detection in point clouds. In: Shape Modeling International Conference (SMI 2010), pp. 175–186 (2010)
32. Weinkauf, T., Gnther, D.: Separatrix persistence: extraction of salient edges on surfaces using topological methods. Comput. Graph. Forum **28**(5), 1519–1528 (2009)
33. You, S., Hu, J., Neumann, U., Fox, P.: Urban site modeling from LiDAR. In: Kumar, V., Gavrilova, M.L., Tan, C.J.K., L'Ecuyer, P. (eds.) ICCSA 2003. LNCS, vol. 2669, pp. 579–588. Springer, Heidelberg (2003). doi:10.1007/3-540-44842-X_59
34. Zhang, G., Vela, P.A., Brilakis, I.: Detecting, fitting, and classifying surface primitives for infrastructure point cloud data. In: Computing in Civil Engineering, pp. 589–596 (2013)

Development of Methods of the Fractal Dimension Estimation for the Ecological Data Analysis

Jakub Jura[1(✉)], Aleš Antonín Kuběna[2], and Martin Novák[1]

[1] Department of Instrumentation and Control Engineering, FME, Czech Technical University in Prague, Prague, Czech Republic
{jakub.jura,martin.novak3}@fs.cvut.cz
[2] Institute of Information Theory and Automation, Czech Academy of Sciences, Prague, Czech Republic
akub@vsup.cz

Abstract. This paper deals with an estimating of the Fractal Dimension of a hydrometeorology variables like an Air temperature or humidity at a different sites in a landscape (and will be further evaluated from the land use point of view). Three algorithms and methods of an estimation of the Fractal Dimension of a hydrometeorology time series were developed. The first results indicate that developed methods are usable for the analysis of a hydrometeorology variables and for a testing of the relation with autoregulation functions of ecosystem.

Keywords: Time series analysis · Fractal · Fractal dimension · Hydrometeorology · Small Water Cycle · Microclimatology

1 Introduction

Fractal object is an ideal entity and its nature is epistemological. There is possible to see partial characteristics of an ideal fractal object on a real object. Especially there is possible to see here any degree of factuality. The difference between fractal and euclidean solid lies in the property which is called self-similarity – repeating same/or similar theme/shape in a different scales. This property indicates relatively high segmentation of the object. And this segmentation is possible to see again and again in higher enlargement of a scale. This principle shown Benoit Mandelbrot [1] and Lewis Fry Richardson on the example of measuring of the coastline length.

When we solving the task how to properly show the collected hydrometeorological data (Fig. 2) from the database [2], we can observe fractal character of these data. The key question sounds, does the fractal properties of hydrometeorological data have a relation to any ecosystem functions (like autoregulation).

2 Fractal Dimension

If we come back to example of coastline length measuring, so there is need to discuss the relation between value of a scale (or better measuring stick size) and the length of a coastline. It is clear that for fractal objects the length of coastline increases with decreasing a size of measuring stick. The steepness of this relation represents the degree of factuality of an object. And after the formal transformation is based for Fractal Dimension estimation.

There are many methods for Fractal dimension estimation. We are interested in a methods intended for the real fractal objects. Since the fractal theory is primarily intended for geometrical objects, so also Fractal Dimension estimation methods are also primarily intended to the geometrical objects (e.g. Box counting [3]). Unfortunately our fractal object is a time series and previous method is not possible to use directly.

3 Motivation

This research is continuation of the project Development of methods for evaluation of flows of energy and matters in the selected Eco-Systems and this project provided us a database of a hydrometeorological data which had collected from 16 meteorological measuring stations (Fig. 3) in South Bohemia near Třebon town - Fig. 1. Each station had measured a few basic hydrometeorological variables (Fig. 2) like a Table 1:

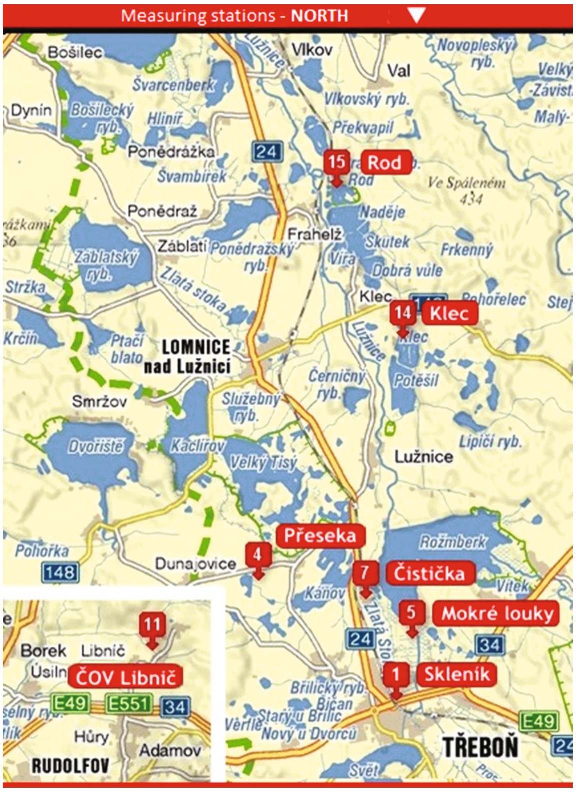


Fig. 1. Map of meteorological stations placement [2].

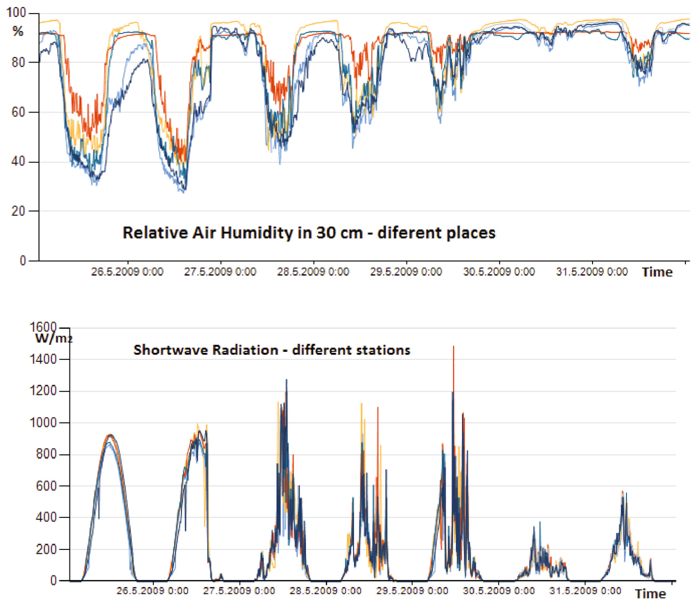


Fig. 2. Graph of measured data (air humidity and incoming solar radiation) [2].

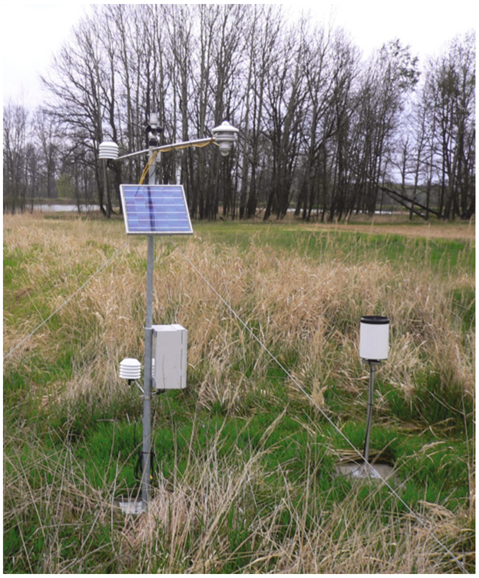


Fig. 3. One of meteorological station in South Bohemia [2]

Table 1. Selected hydrometeorological variables from database TOKENELEK [2]

Air temperature in 2 m and 30 cm above the ground	°C
Relative air humidity in 2 m and 30 cm above the ground	%
Incoming solar radiation	W/m^2
Reflected solar radiation	W/m^2
Precipitation	mm
Wind speed and wind direction	m/s, DEG

Measuring stations were placed at a different typeset of a sites {field, wet meadow, concrete surface, fishpond, pasture, village} [4]. The aim of this article is to add a new point of view to the autoregulation in the landscape exploration. Especially the autoregulation of a temperature depends on the water in the landscape and on the state of a Small Water Cycle SWC [5–7].

Note: It is possible to estimate an autoregulation as an ability to regulate given environmental variable to a given value. E.g. the temperature is in the nature environment regulated to the temperature which is convenient to a local flora and fauna.

The relation between diversity (especially biodiversity) of an ecosystem and a stability and the level of the autoregulation is currently discussed. To help to find the relation between fractal characteristics of a hydrometeorological variables and stability and the level of the autoregulation is the aim of this work. Partial aim is to develop the reliable method for estimation of Fractal Dimension of mentioned hydrometeorological variables.

4 Fractal Dimension of Time Series

In general the idea of calculating of the fractal dimension is derived from a coastline measuring and its (in general) formulated [8] as:

$$D = \frac{\log N}{\log(1/r)} \tag{1}$$

Where:

D … the fractal dimension
N… the number of an object's internal homotheties
r … the ratio of homothety

One of methods of the estimation of the Fractal dimension of real object is the Box Counting Method (e.g. [3, 9, 10]). This method is based on the covering of the object by the n-dimensional spheres, or cubes of given size ε.

$$D_C = \lim_{\varepsilon \to 0} \frac{\log[C(\varepsilon)]}{\log \frac{1}{\varepsilon}} \tag{2}$$

Where:

D … the estimation of FD
ε … the size of the n-dimensional cubes
C(ε) … the minimal number of n-dimensional cubes

But this method is suitable primarily for the geometrical object – not for time series (because induce an additional problem with choice of time-size scale). This method is possible to adapt to time series [11], but we move away from the original principle of Fractal dimension by use this method. Our aim is to use here methods which will closely corresponds to the principle of a coastline length measuring.

5 Proposed Methods of Fractal Dimension Estimation

Here proposed methods is based on the principle of smoothing of the time series – and changing the coefficient of the smoothing. From the dependency between the length of the curve and smoothing coefficient is the value of Fractal dimension estimated. The primarily smooth course have a low Fractal Dimension estimation, because we smoothing of the smooth object. And the fractal dimension of a linear course is equal to zero. Next is description of three smoothing methods used for estimation of Fractal dimension – Moving Average, Moving Maximum and Minimum and method of Local Regression.

6 Moving Average (FDMAvg)

This method is based on the analogy with length of the coastline measuring. Similarly as a measuring stick size is changed (or box size ε in the case of Box counting method), so the window of the moving average is changed here.

6.1 Description of Algorithm

Firstly the set of Window sizes E = {1,2,3,5,10,20, …} is determined. Given window size is labelled here in according to Box Counting as εn, or EPSn and is chosen from this set. For each data row index "k" and data x(k), from X the value AVG(k) of mean with given window size εn is calculated (for k from 1 to m- εmax,, where m is inex of last element).

$$AVG(k) = \frac{1}{\varepsilon}\sum_{i=k}^{k+\varepsilon_n} x_i \tag{3}$$

Next is calculated the variance "VAR".

Def: variance VAR is for our purpose the absolute value of a difference between two consequential values of AVG(k). Continue the mean value of the variance VV_AVG(n) for each window size εn is calculated:

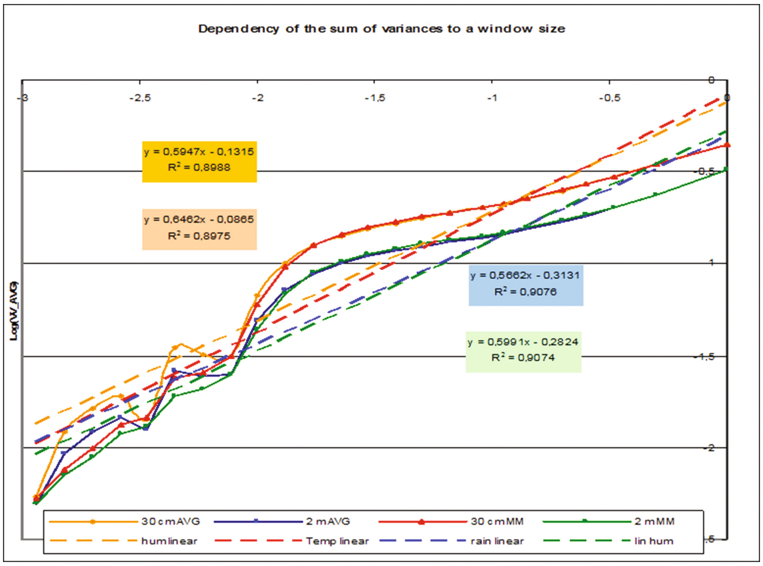


Fig. 4. Estimation of fractal dimension of temperatures at 30 cm and 2 m above the ground - station 9 CIGLEROVSKY_STO_All_Year_2008

$$VV\_AVG(n) = \frac{1}{m - \varepsilon_{\max}} \sum_{k=1}^{m-\varepsilon_{\max}} \left| \frac{1}{\varepsilon} \sum_{i=k}^{k+\varepsilon} x_i - \frac{1}{\varepsilon} \sum_{i=k}^{k+\varepsilon} x_{i+1} \right| \tag{4}$$

the estimation of the Fractal Dimension is the slope of the line approximating the dependency Log(VV_AVG) on Log(1/ε) - Fig. 4.

6.2 Methodology of Use FDMAvg Algorithm

This method (similarly as others methods for estimation of Fractal Dimension) is sensitive to parameters of the calculation. For reach out the reliable results it is need to keep the methodological recommendation.

Determination of minimal window size ($\varepsilon_{\min}$). Since our data are sampled with the period 10 min, so we can use as a minimal window size the wide of one sample. The effect of the dynamical characteristic of the measuring instrument is insignificant. Hypothetically, in the case of a smaller sample period than time constant of measuring instrument, the result of fractal dimension estimation will be related more to the measuring instrument than measured object.

Determination of maximal window size ($\varepsilon_{\max}$). The determination of the maximum window size is not equally clear as a minimum windows size. For example Felix Hausdorf works with 1/5 of all range. Although a time series really has a finite length, theoretically can same series be longer (or newer end). Moreover the Fractal dimension estimation is significantly affected by this. Because basic data set has a length one month, so we work with maximal window size 1/5 of month.

Step of the window size changes. Because the method is based on a calculation of the slope of the line. And the line parameters are estimated by the minimal square method, so we have to ensure a uniform step of window size ε in logarithmic scale. In the opposite case the impact of the segment of points which are more close to a small values is significantly bigger than an impact of outlying ones.

Total amount of used window size is primarily not important for calculation (the line parameters influences minimally). A higher density of ε is useful for a visual evaluation of the results. For example on the Fig. 4 is possible to see the step (close to one day ε size) and many waves in higher values. For a higher reliability of a comparison of many calculations is advantageous use same set of ε for all of them.

7 Moving Min&Max (FDMM)

This method is very similar as a previous – Moving Average FDMAvg. For analysis is used sums of moving minimums VV_MIN, sums of moving maximums VV_MAX and above all their mean value VV_MM. VV_MIN and VV_MAX have usually similar course. Bigger difference is possible to observe for sharply unsymmetric variables, like a precipitation. Normally the difference is small and we can estimate the Fractal dimension from the VV_MM.

7.1 Description of Algorithm

The window sizes set is same as for method FDMAvg and also other procedures. The minimums MIN(k) and maximums MAX(k) are calculated instead of mean value AVG(k).

$$MIN(k) = \min(x_i); MAX(k) = \max(x_i) \tag{5}$$

for $i = \{k, \ldots, k + \varepsilon\}$

Next is calculated the "variance" VAR for all rows. And the mean value of these variances VV_MIN(n) and VV_MAX(n) for each window size ε_n.:

$$VV\_MIN(n) = \frac{1}{m - \varepsilon_{\max}} \sum_{k=1}^{m-\varepsilon_{\max}} |MIN(k) - MIN(k+1)| \tag{6}$$

$$VV\_MAX(n) = \frac{1}{m - \varepsilon_{\max}} \sum_{k=1}^{m-\varepsilon_{\max}} |MAX(k) - MAX(k+1)| \tag{7}$$

$$VV\_MM(n) = \frac{1}{2}(VV\_MAX(n) + VV\_MIN(n)) \tag{8}$$

And finally the estimation of the Fractal Dimension is again derive from the slope of the line approximating the dependency Log(VV_MM) on Log(1/ε). (Or VV_MAX or VV_MIN).

8 Local Regression (FDLR)

For the parametrised smoothing of the time series was used the smoothing spline method, which is special type of the local regression loes curve [12]. The aim of this method is to approximate the data set of observations [t_i, x_i] by the function y = y(t): [min(t_i) max(t_i)] → R, which is compromise between accuracy and smoothness. For the selected differential operator L: $C^3 \to C^3$and weight ratio p.

$$RSS_p(y) = p\sum_i (y(t_i) - x_i)^2 + (1-p)\int_{t_{\min}}^{t_{\max}} L^2(y)dt \tag{9}$$

We use L(y) = y''

$$RSS_p(y) = p\sum_i (y(t_i) - x_i)^2 + (1-p)\int_{t_{\min}}^{t_{\max}} (y'')^2 dt \tag{10}$$

And approximation and substitution

$$\frac{d^2y}{dt^2} \approx \frac{y(t+\Delta) - 2y(t) + y(t-\Delta)}{\Delta^2} = \frac{(y_{n+1} - 2y_n + y_{n-1})^2}{\Delta^2} \tag{11}$$

Where selection of Δ has fulfill the condition of equidistant partition of the interval by the step Δ which cover all unobserved data. If $\lambda = \frac{1-p}{p}\Delta^{-3}$ o we solve the optimization task:

$$\begin{gathered} y = \arg\min_y \left(\sum_{n \in S} (y_n - x_n)^2 + \lambda \sum_n (y_{n+1} - 2y_1 + y_{n-1})^2 \right) \\ \lambda \in (0, \infty) \end{gathered} \tag{12}$$

S is set of indexes for which an observation exists. If $\lambda \to \infty$ than y → at + b (optimum is close to linear regressive approximation of data. Conversely, for $\lambda \to 0$, the output function y approximates measured date accurately.

For any λ (0, ∞), the optimisation task is convex and the sole local and global optimum is solution of the system of equations:

$$\frac{\partial}{\partial y_n}\left(\sum_{n \in S} (y_n - x_n)^2 + \lambda \sum_n (y_{n+1} - 2y_1 + y_{n-1})^2 \right) = 0 \tag{13}$$

or better:

$$(F + \lambda E)y = \widehat{x} \tag{14}$$

where:

$$E = \begin{pmatrix} 1 & -2 & 1 & & & \\ -2 & 5 & -4 & 1 & & \\ 1 & -4 & 6 & -4 & 1 & \\ & 1 & -4 & 6 & -4 & 1 \\ & & \dots & \dots & 5 & -2 \\ & & & 1 & -2 & 1 \end{pmatrix} \tag{15}$$

$$F = diag(f)$$

Where $f_n = 1$ for indexes for which the observation exist and $f_n = 0$ for indexes for which the observation does not exist. Vector $\widehat{x}$ contains values of x_i for indexes i for which the observation exist; and $x_n = 0$ for i out of set S contains approximation at the point without measuring. Linear task was solved in Matlab software with help of the Sparse Matrix procedures.

This method provides us the matrix of smooth data which were used in similar way as a previous. The variance VV_LR was calculated, ε_n was replaced by the p_n and the Fractal Dimension was estimated from the given dependency Fig. 6.

$$VV\_LR = |M(i,j) - M(i+1,j)| \tag{16}$$

Also for this method is need to define the range of ε_n, especially the p_n and situation is differ from previous because the p_n parameter has no time dimension. The dependency on the parameter p from –32 to 32 has three parts:

1. The first part is constant – there is not smoothing, because fineness of smoothing is smaller than sampling time of data.
2. The second part decreasing linearly an is used for Fractal Dimension estimation and is shown on the Fig. 6.
3. The third part goes directly to zero.

The differences in Estimated Fractal Dimension by FDLR method between various hydrometeorological variables, corresponds relatively with results of FDAvg and FDMM.

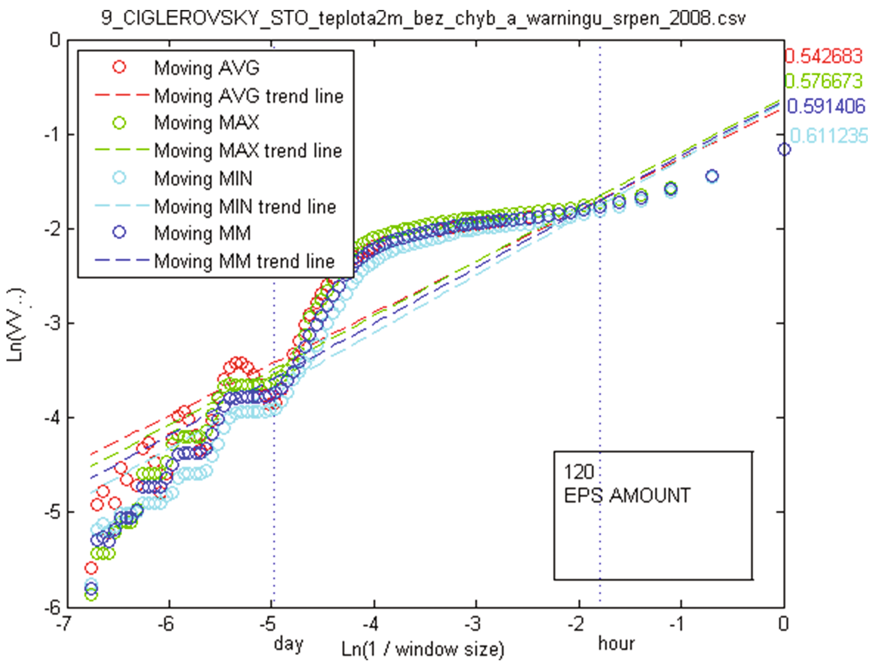


Fig. 5. Dependency *Log(VV_LR) on Log(1/ε).*

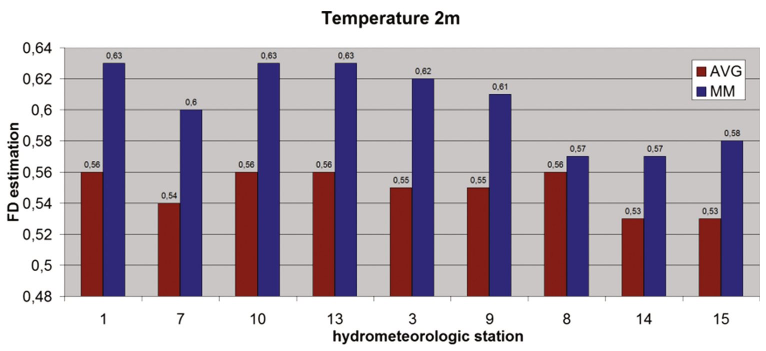


Fig. 6. Estimation fractal dimension of temperature in 2 m above the ground. Station 1 is in a small town, stations 7 has concrete surface, stations 10,13 are from meadow, stations 3 and 9 are from wet terrain and stations 8,14 and 15 from lake.

9 Preliminary Results

Before calculating has started, the database records had to been checked to an operational errors (in the database had been records which had represented breakdowns states of a measuring device). Moreover the basic analysis of data credibility had been made also.

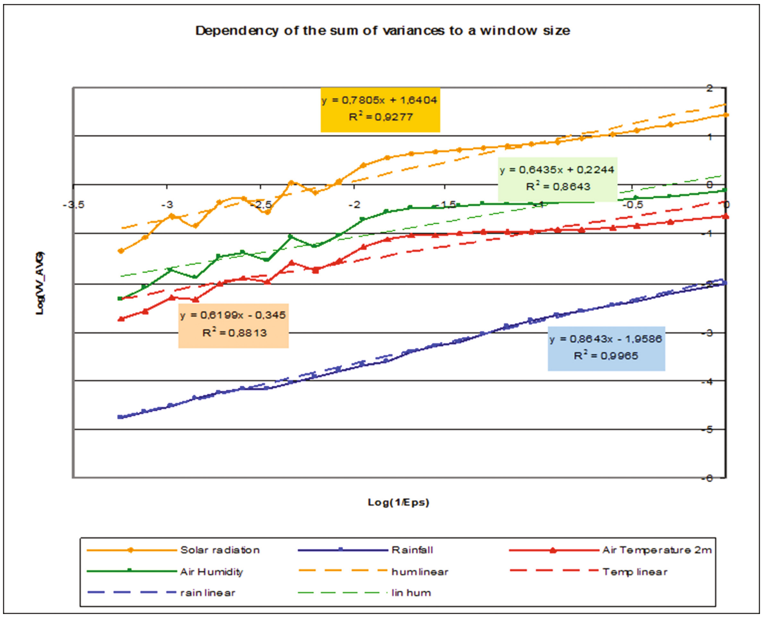


Fig. 7. Estimation of fractal dimension of different variables from station 9 CIGLEROVSKY_STO_july_2008, FDMAvg method.

The series of the first estimation was done for a few selected variables on a different sites – Fig. 7. Here is possible to see that the Fractal Dimension of a temperature is higher at the locality with higher evapotranspiration (green areas versus concrete area or lake).

This consideration is also supported by the example of a results of a Fractal Dimension estimation of Temperatures which is measured in 2 m and 30 cm above the ground (Fig. 4). The Fractal Dimension of variables measured more close to the ground points to influence of a Small Water Cycle. At the other side, the rigorous verification of this conclusion is not aim of this article and should to be done with consideration of many aspects of microclimatology.

Also the differences between various hydrometeorological variables are observable at the level of a Fractal Dimension and also at the level of a course of dependency shown at Fig. 7. Question is what represent a waves between ε = ((–2)…(–3)) in a mentioned dependency (shown at Fig. 7) if we measure physically connected variables like a: air temperature, relative air humidity and incoming solar radiation. For example the spectral analysis would give an explanation of this phenomenon.

10 Conclusion

In this article is described three methods for an estimating of the Fractal Dimension of a hydrometeorology variables like an air temperature, relative air humidity, incoming and reflected solar radiation, precipitation, soil humidity etc. at a different sites in a South

Bohemia landscape. Mentioned methods of Fractal Dimension estimation are based on relation between smoothed length of the course of the given hydrometeorological variable and independent smoothing parameter. For smoothing of courses are used the moving average (and moving maximum and minimum) and local regression methods. All methods are developed at the level of algorithm and also at the level of methodology, containing special parameters setting.

The preliminary results indicate that developed methods are usable for the analysis of hydrometeorology variables and for a testing of the relation with autoregulation functions of ecosystem.

Acknowledgements. This work was supported by the project SGS15/189/OHK2/3T/12 and GACR P402/12/G097.

References

1. Mandelbrot, B.: Fractals: Form, Chance and Dimension (In Czech). W.H. Freeman & Company, San Francisco (1977)
2. Tokenelek (2007). http://tokenelek.fsid.cvut.cz/
3. Breslin, M.C., Belward, J.A.: Fractal dimensions for rainfall time series. Math. Comput. Simul. **48**, 437–446 (1999)
4. Huryna, H.: Effect of different types of ecosystems on their meteorological conditions and energy balance components. University of South Bohemia (2015)
5. Ripl, W.: Management of water cycle and energy flow for ecosystem control: the energy-transport-reaction (ETR) model. Ecol. Model. **78**, 61–76 (1995)
6. Bila, J., Jura, J., Pokorny, J., Bukovsky, I.: Qualitative modeling and monitoring of selected ecosystem functions. Ecol. Model. **222**, 3640–3650 (2011)
7. Bila, J., Jura, J., Pokorny, J., Bukovsky, I.: Qualitative modeling in the landscape development monitoring. Recent Researches in System Science, vol. 222, pp. 35–41. WSEAS Press, Athens (2011)
8. Mandelbrot, B.: The Fractal Geometry of Nature. W.H. Freeman, San Francisco (1982)
9. Lopes, R., Betrouni, N.: Fractal and multifractal analysis: a review. Med. Image Anal. **13**, 634–649 (2009)
10. Liaw, S.-S., Chiu, F.-Y.: Fractal dimensions of time sequences. Phys. A Stat. Mech. Appl. **388**, 3100–3106 (2009)
11. Jura, J., Bíla, J.: Computation of the fractal dimension of meteorological quantities. In: Mendel 2010: 16th International Conference on Soft Computing 2010, Brno University of Technology (2010)
12. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, Heidelberg (2009)

The Detection and Interpretation of Emergent Situations in ECG Signals

Jiří Bíla(✉) and Jan Vrba

Department of Instrumentation and Control Engineering, FME,
Czech Technical University in Prague, Prague, Czech Republic
{jiri.bila, jan.vrba}@fs.cvut.cz

Abstract. The paper continues in previous works published by the authors where the emergent situations were detected by the violence of structural invariants. In this paper is used only one type of structural invariant – Matroid and Matroid Bases (M, BM) investigating the influence of their violation to interacting elements (components) in so called basic group (compartment). The application of the presented approach is demonstrated in cases of discovery of diseases (diseases of cardio-vascular system). In the second plan of this paper is introduced a new method of the discovery of a semantic content of emergent shapes in ECG signals. The method is illustrated in the case of cardiac arrhythmia diagnosis.

Keywords: Emergent phenomena · Detection of emergent situations · Matroid and its bases · ECG signals · Interpretation method · Approximation of semantic content

1 Introduction

The paper continues in topics and technologies introduced in papers [1–4]. Especially papers [2, 4] are needed for the effective reading and the understanding this paper. Similarly as in paper [1] we start here with 5 working hypotheses:

Hypothesis 1(H1): The emergent phenomenon is induced by a sharp change of complex system structure ("jump on the structure").

Hypothesis 2(H2): In case that we accept *H1*, the eventuality of an emergent phenomenon appearance may be detected as a sharp violence of complex system structure (or the violence of Structural Invariants respectively).

Hypothesis 3(H3): The appearance of emergent situation is detected as the possibility of extension (reduction) of the basis of the matroid (matroid formed on the complex system) [18] by at least one element.

Hypothesis 4(H4): One of possibilities how to represent detection of appearance of emergent situation by extension of a matroid basis is to anticipate the increase of number of interacting elements in so called basic group (compartment).

Hypothesis 5(H5): In order to attain an emergent phenomenon (by an extension (reduction) of a matroid basis) the complex system increases the number of elements (transformations, properties, processes) in *basic group* (compartment) by a minimum number of elements (transformations, properties, processes).

The technique of work with emergencies is used twice in this paper. At first for the detection of emergent situations and at second in the method of interpretation. In both these cases are used hypotheses H3, H4, H5. In the process of the detection we calculate only with number of elements in a basis of matroid and we compute the possibility of appearance of emergence (understood here as a start of a disease).

At the process of interpretation of an unknown state structure we consider emergent process as a result of a *semiotic* interaction between a configuration of signs (symbols) and the mind of human solver. It means that *we do not model emergence process* but we form the sign configuration that induces emergence phenomenon [1, 2]. The interpretation process is oriented in the danger of heart failure.

Notes to organization of the paper.

In Sect. 2 there are introduced some works that are relevant to the topic of our paper. The background and method for the detection of emergent situations are introduced in Sects. 3 and 4. The theory and the explication of interpretation method are in Sect. 6. The application of the interpretation for ECG signals for cardiac arrhythmia diagnosis are in Sect. 7.

2 Some Related Works

Essential knowledge sources about complex systems, emergence phenomena and complexity are concentrated in [2, 4, 6, 7]. The detection of emergent situations is not too frequent topic in papers and some of them were investigated in [1, 5].

The analysis of ECG signals for cardiac arrhythmia is still very important field of the research. The main problem consists in no negligible traces of deterministic chaos in these signals. There have been applied and explored very sophisticated methods for the processing of ECG signals with arrhythmia. We name here only a few of them. A pattern recognition technology [9, 10] (rather old but still very effective). Very progressive methods based on neural networks and on their combination with Hilbert (or Hilbert-Huang) transform, [11–13]. Hidden Markov Chains [14, 15] - as still attractive field for analysis of ECG signals. Unfortunately - none of these introduced sources did not deal with interpretation of new shapes in ECG signals. In this paper we turn to methods of linear discrimination analysis used for the investigation of ECG signals time ago, [16, 17] and the results of these works we used for application of the introduced interpretation method.

3 Background for the Detection of Emergent Situations

The simple scheme for the detection of emergent situations by means of algebras of transformations [1] is the following one:

$$\mathrm{G1} \rightarrow (\mathrm{G1} \oplus \mathrm{X}) \rightarrow \text{Chaotic phase} \rightarrow \mathrm{EP} \rightarrow \mathrm{SOP} \rightarrow \mathrm{G2}, \tag{3.1}$$

where G1, G2 are algebras of transformations characterizing complex system in situations of balances and X is a set of transformations that extends (reduces) kernel part of G1. (In case that is used matroid approach the kernel part will be the basis of the matroid.) Symbol ⊕ has no specific significance and depends on a real case of method application. EP symbolizes “emergent phenomenon” and SOP is “self organizing process”.

In paper [1] were discussed problems of “chaotic phases” and “self-organizing processes” that need more detailed investigation however not necessarily introduced here.

For the description of the discussed complex system will be applied state model. Transformations transfer the system from one state to another state and the only problem is to detect a possible appearance of an emergent situation. For it is used a common decision rule:

$$\mathrm{IF}\ (\#\mathrm{X} \geq \min \Delta\, \mathrm{f(RN)}) \Rightarrow (\mathrm{PAES}) \tag{3.2}$$

where “min Δ f(RN)” is a minimal difference between further and actual Ramsey number and PAES is “a possible appearance of an emergent situation”. (Example is in Sect. 4.)

4 Detection of Diseases of Cardio-Vascular System

One of very important information about the health of human heart is ECG diagram. In Fig. 1 is illustrated how the part of so called PQRST complex corresponds to states in one cycle of the heart activity. In our state approach description we see 6 transformations that realize this cycle:

$$\begin{aligned}\tau_U(\mathrm{U}(k-1)) \rightarrow \mathrm{P}(k) \rightarrow \tau_P(\mathrm{P}(k)) \rightarrow \mathrm{Q}(k) \rightarrow \tau_Q(\mathrm{Q}(k)) \rightarrow \mathrm{R}(k) \rightarrow \tau_R(\mathrm{R}(k)) \\ \rightarrow \mathrm{S}(k) \rightarrow \tau_S(\mathrm{S}(k)) \rightarrow \mathrm{T}(k) \rightarrow \tau_T(\mathrm{T}(k)) \rightarrow \mathrm{U}(k),\end{aligned} \tag{4.1}$$

where P, Q, R, S, T, U are phases (states) of ECG signal (according to Fig. 1) and τ_P, τ_Q, τ_R, τ_S, τ_T, τ_U are transformations. Without a loss of generality we consider a composition algebra

$$\mathrm{A} = \langle \{\tau_P, \tau_Q, \tau_R, \tau_S, \tau_T, \tau_U\}, \mathrm{o}\rangle, \tag{4.2}$$

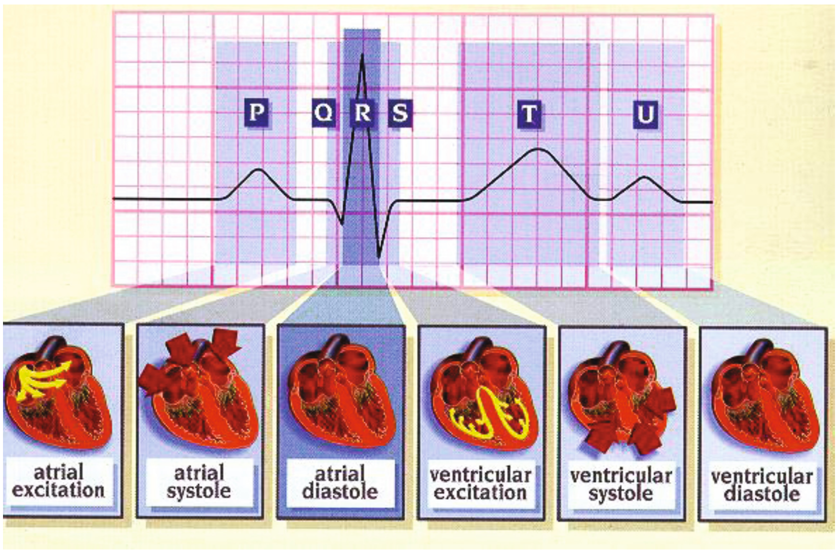


Fig. 1. Phases P, Q, R, S, T, U of ECG signal (source [8])

where "o" is symbol for the composition of transformations. Expression (4.1) illustrates transformation activities as, e.g.,

$$P(k) = \tau_U(U(k-1)),\ Q(k) = \tau_P(P(k)),\ldots,\ U(k) = \tau_T o \tau_S o \tau_R(R(k)), \qquad (4.3)$$

Within the framework of our theory we suppose that individual disease of heart are presented in ECG diagram as a composition of deformation transformation τ_*, τ_+, τ_{**}, … that modify (deform) the parts of ECG diagram of a healthy man. The result of application of our theory is the estimation how many of such transformation induce the emergence of a disease.

As an example let us consider the carrier of algebra A (4.2) as a carrier of a matroid M. Considering hypotheses H1 – H5 from the Introduction and the principles introduced in [1, 2] we search for a minimal Ramsey number that could extend the basis of the actual matroid M.

Our matroid M has 6 elements and three element bases – R(#B1,#B2) = 6 = R(3,3). So that - for a minimal extension of basis of this matroid we need at least 3 elements – R(3, 4) = 9, [19].

For illustration we may consider three deformation transformations:
τ_{Q+} … deforms phase Q, τ_{R+} … deforms phase R, τ_{S+} … deforms phase S.

Considering the influence of these transformation into ECG diagram of a healthy man we obtain basic conditions for heart disease called "Left Bundle Branch Block (LBBB)". By the same way is possible to consider about another heart diseases.

In work [17] have been published results of investigation of cardiac arrhythmia diagnosis using discrimination analysis on ECG signals. Translating the method from

[17] into our terminology and context – the authors used 9 transformations (τ_i) describing ECG signal and deforming transformations of the type of linear shifts (τ_d):

$$i = 1, \ldots, 9, \tau_i : \text{PQRST} \to R^1, \tau_i(\text{PQRST}) = x_i, \tau_d(x_i) = \alpha_d x_i + \beta_d, \alpha_d, \beta_d \in R^1 \tag{4.4}$$

Basis for matroid with 9 elements has 3 or 4 elements. Further Ramsey number providing extension of this matroid basis by at least one element is 14. According to results published in [17] all discovered diseases needed at least 5 deformation transformation applied in the ECG signal of a *healthy man*. (In Sect. 7 there will be introduced in details the following hearth diseases:

LBBB – (Left Bundle Branch Block), **RBBB** (Right Bundle Branch Block), **VPC** (Ventricular Premature Contraction), **APC** (Atrial Premature Complex)).

5 Interpretation Space, Sign Model and Connections in the Interpretation Process

The concepts Interpretation Space and Sign model have been introduced, explained and used in our previous works many times, e.g., in [2, 4, 5].

Interpretation Space was presented as a large fragmental map of contents (and explanations and meanings) of objects and phenomena from a complex system.

Sign Model was represented as a multi-strata structure of signs and symbols used for modeling the objects and phenomena from a complex system.

A mutual relation between Interpretation Space and Sign model (considering only two neighboring representation levels (j and j+1) is illustrated in Fig. 2.

The assigning of the internal representation of **IS** in stratum "j" to signs and sign formations in ^{j}**SM** is represented by mapping ^{j}Sem (called here "semiotic mapping"). Similarly - the assigning of signs and sign formation in jth of ^{j}**SM** to semantic concepts and meanings in ^{j}**IS** is represented by mapping ^{j}I (called here "interpretation mapping").

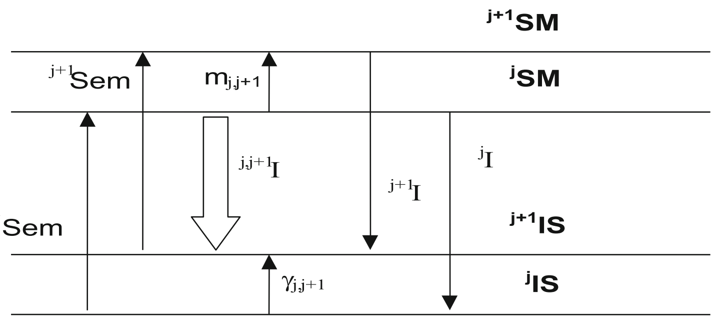


Fig. 2. Connections in the interpretation process

Collection of mapping $\langle {}^{j}\mathrm{Sem}, {}^{j}\mathrm{I}, \gamma_{j,j+1}, m_{j,j+1} \rangle$ is called in this paper "connection".

Connections are defined in detail according to objects and phenomena in strata (^{j}**IS**, ^{j}**SM**) resp. (${}^{j+1}$**IS**, ${}^{j+1}$**SM**).

The diagram in Fig. 2 contains one empirical knowledge that is in our paper one of the principal: mapping ${}^{j,\ j+1}$I is not attainable, in general, by a constructive way using the known components $\gamma_{j,j+1}$, $m_{j,j+1}$, ^{j}I, so that for the construction of ${}^{j,\ j+1}$I is impossible to use "equation" (5.1)

$$ {}^{j,j+1}\mathrm{I} \ = \gamma_{j,j+1} \mathrm{o}\, {}^{j}\mathrm{I} \ ={}^{j+1}\ \mathrm{I}\ \mathrm{o}\ m_{j,j+1}. \tag{5.1}$$

This situation introduces one of principal emergence phenomena that appears in the interpretation from stratum (j) to stratum (j+1). We consider this emergent phenomenon as a result of a *semiotic processes* of interaction between a configuration of signs (symbols) and the mind of human solver.

6 The Method of Interpretation

We introduce in this Section one adaptation of the original method introduced in [5]. The method allows to approximate mapping ${}^{j,\ j+1}$I for unknown emergent shapes. As an example of demonstration of this method are used in this paper ECG signals (in continuation of works [12, 13]). In works [2, 5] have been studied emergent phenomena as processes induced by a "saltatory" violation of system structure resp. of its *structural invariants*. The interpretation in our context is understood as an emergent process induced by the violation of a special structural invariant - a Matroid (M) and its bases (MB).

From many descriptive means that could be applied in our case, we use in this paper states (descriptors of ECG QRS complexes, e.g., H-QR, H-RS, Slope-QR, …, see Table 1.), state descriptions (ECG QRS complexes represented as *vectors of values of descriptors, i.e., vectors of real numbers*) and *matroids*.

Table 1. Descriptors and their units

	Features	Feature description	Units
1	H-QR	The amplitude between Q and R in a QRS complex	mV
2	H-RS	The amplitude between R and S in a QRS complex	mV
3	QRS-dur	The time duration between Q and S in a QRS complex	ms
4	QTP-int	The time duration between Q and T in a QRS complex	ms
5	Ratio-RR	The ratio of RR is the length of a single RR-interval	1
6	Slope-QR	The slope between Q and R in a QRS complex	mV/ms
7	Slope-RS	The slope between R and S in a QRS complex	mV*ms
8	Area-QRS	The area of QRS complex	mV*ms
9	Area-R′ST′	The area of R′ST′ in a QRS complex	mV*ms

6.1 Description of the Method

(i) Let us consider the set ^{j}W in the stratum ^{j}**SM**, that contains states (descriptors of ECG QRS complexes, e.g., H-QR, H-RS, Slope-QR, …, see Table 1.) and the set *B*(^{j}W) that contains state descriptions (i.e., ECG QRS complexes represented as vectors of values of descriptors).

(ii) Let us consider a novel vector of values of descriptors $^{j}w \in B(^{j}W)$ with its name **AX** in stratum $^{j+1}$**SM**. We want to interpret **AX** into stratum $^{j+1}$**IS** (i.e., to discover its unknown semantic content).

(iii) There is constructed a matroid (**M**) and its bases (**MB**) on the set of classes of ECG signals (represented as vectors of intervals of descriptor values) in $^{j+1}$**SM** using relation **DNT** (**IND**) (e.g., in [2, 5]).

(iv) The induction of an emergent process is executed by the extension of some of matroid bases from **MB** by **AX**. In this case (i.e., if it is possible to extend some of bases) ***AX*** is interpreted by a new concept created by human solver. The extension of some of matroid base (B∈**MB**) in level "j+1" by ***AX*** induces a desired emergent phenomenon and introduces a *new cognitive dimension* (***AX***). It means that we observe the considered system not only by dimensions from B but also by dimension ***AX***.

(v) In case when ***AX*** is not able to extend any of matroid bases then ***AX*** is interpreted by "similar" objects from $^{j+1}$**SM**.

End of the method description.

7 Example of Interpretation of Novel Shape ECG Signal of Arrhythmia Diagnostics

The interpretation method will be explained in this paper for the case of special selection of ECG signals using data from paper [15].

In stratum $^{j+1}$**SM** we have names (Ai) of five typical ECG signals: **NORM** (Normal), **LBBB** (Left Bundle Branch Block), **RBBB** (Right Bundle Branch Block), **VPC** (Ventricular Premature Contraction), **APC** (Atrial Premature Complex) and signals **AX1** and **AX2** with unknown semantic contexts.

In stratum ^{j}**SM** are descriptions (DAi) of signals from stratum $^{j+1}$**SM** – in Tables 1, 2 and 3.

In stratum $^{j+1}$**IS** we have semantic contents of ECG signals from $^{j+1}$**SM** (they will be introduced in next text).

In stratum ^{j}**IS** we have semantic contents of descriptions DAi from ^{j}**SM** (i.e., contents of descriptive variables from Tables 1, 2, and 3).

7.1 Semantic Contents of ECG Signals from $^{j+1}$SM

NORM is an ECG signal of a standard patient without any clinical disorders.

Table 2. Vectors of intervals of descriptor values

Features	Heartbeat case	Heartbeat case	Heartbeat case
	NORM (range)	**LBBB** (range)	**RBBB** (range)
H-QR	[0.695, 2.690]	[0.205, 2.060]	[0.695, 2.240]
H-RS	[0.80, 3.645]	[0.705, 2.815]	[0.955, 3.30]
QRS-dur	[33.0, 79.0]	[86.0, 115.0]	[46.0, 130.0]
QTP-int	[43.0, 90.0]	[135.0, 230.0]	[110.0, 210.0]
Ratio-RR	[0.8, 1.2]	[0.85, 1.3]	[0.775, 1.5]
Slope-QR	[0.019, 0.134]	[0.004, 0.050]	[0.012, 0.111]
Slope-RS	[0.017, 0.214]	[0.043, 0.055]	[0.023, 0.136]
Area-QRS	[20.0, 82.0]	[0.0, 146.15]	[25.01, 120.96]

Table 3. Vectors of intervals of descriptor values (cont)

Features	HC	HC		
	VPC (range)	**APC** (range)	**AX1** (val)	**AX2** (val)
H-QR	[0.105, 3.095]	[0.275, 1.870]	2.92	1.5
H-RS	[0.87, 3.575]	[0.490, 2.345]	0.41	1.2
QRS-dur	[52.0, 210.0]	[34.0, 61.0]	223.0	74.0
QTP-int	[120.0, 485.0]	[50.0, 112.0]	526.4	164.4
Ratio-RR	[0.45, 0.760]	[0.41, 0.760]	0.25	0.91
Slope-QR	[0.002, 0.061]	[0.006, 0.079]	0.14	0.031
Slope-RS	[0.011, 0.108]	[0.013, 0.162]	0.15	0.058
Area-QRS	[4.16, 289.92]	[10.59, 69.35]	351.2	55.2
Area- R′ST′	[0.0, 265.0]	[0.0, 152.0]	282.4	52.1

LBBB (Left Bundle Branch Block) is a fault in transfer of impulses by myocard (as a consequence of violence of heart transfer system). LBBB indicates a disorder in left ventricle chamber. The reasons of LBBB could be: cardiomyophaty; fault of left heart flap; hypertension; ischemic cardiomyophaty. LBBB increases the danger of heart failure, unexpected heart death, infarct of myocard.

RBBB (Right Bundle Branch Block) is a fault in a transfer of impulses by myocard inducing a late activity of right heart chamber (as a consequence of violence of heart transfer system). The reasons of RBBB could be: pressure overload of right heart; embolia; fault of right heart flap; ischemic cardiomyophaty. RBBB is detected sometimes in ECG signals of sportsmen nevertheless increases the danger of heart failure, unexpected heart death, infarct of myocard.

VPC (Ventricular Premature Contraction) may be perceived as a "skipped beat" or felt as palpitations in the chest. In a VPC, the ventricles contract first and before the atria have optimally filled the ventricles with blood, which means that circulation is inefficient. The reasons of VPC could be: adrenaline excess; cardiomyopathy, hypertrophic or dilated; certain medicines such as digoxin, which increases heart contraction or tricyclic antidepressants; alcohol; caffeine; myocardial infarction; hypercapnia (CO_2 poisoning); hypoxia. Single beat VPC abnormal heart rythms are not a danger and can be asymptomatic in healthy individuals. They may also cause chest pain, a faint feeling, fatigue, or hyperventilation after exercise. Several VPCs in a sequence become in a form of ventricular tachycardia (VT), which is a potentially fatal abnormal heart rhythm.

APC (Atrial Premature Complex) results from a premature, ectopic, supraventricular impulse that originates somewhere in the atria outside of the SA node. A single complex occurs earlier than the next expected sinus complex. After the APC, sinus rhythm usually resumes. The reasons of APC could be: stress; caffeine; alcohol; heart failure; myocard infarct; valvular disease; chronic lung disease; hyperthyroidism; electrolyte abnormalities (hypokalemia); medications (digoxin). Occasional premature atrial contractions are a common and normal finding and do not indicate any particular health risk. Rarely, in patients with other underlying structural heart problems, APCs can trigger a more serious arrhythmia such as atrial flutter or atrial fibrillation.

7.2 Application of the Interpretation Method

The interpretation method described in Sect. 6. will be now applied for signals ***AX1*** and ***AX2*** (from Table 3.). The application is managed by items (i), …, (v):

(i) ${}^{\mathrm{j}}\mathrm{W}$ = {H-QR, H-RS, QRS-dur, QTP-int, Ratio-RR, Slope-QR, Slope-RS, Area-QRS, Area-R′ST′},

$$B\left({}^{\mathrm{j}}\mathrm{W}\right) = \{\langle 0.71, 1.3, 65.0, 72.4,\ 0.95,\ 0.11,\ 0.15, 35.4, 11.4\rangle, \langle\ldots\rangle,\ \ldots\},$$

Vectors $B({}^{\mathrm{j}}\mathrm{W})$ are concentrated in Tables 2 and 3.

(ii) $\mathbf{AX1} = \langle 2.92,\ 0.41, 223.0, 526.4,\ 0.25,\ 0.14,\ 0.15, 351.2, 282.4\rangle$,
$\mathbf{AX2} = \langle 1.5, 1.2, 74.0, 164.4,\ 0.91,\ 0.031,\ 0.058, 55.2, 52.1\rangle$,

(iii) There is constructed a matroid (**M**) and its bases (**MB**) on the set of classes of ECG signals (represented as vectors of intervals of descriptor values) in ${}^{\mathrm{j+1}}\mathbf{SM}$ using relation **DNT** (**IND**), (e.g., in [2, 5]).
For relation **DNT** was used the expert criterion and the remaining conditions were adapted for our case on one expression:
Classes $\mathrm{A_i}$, $\mathrm{A_j}$ are dependent if holds:

$$\left(\mathrm{A_i}\mathbf{DNT}\mathrm{A_j}\right) \Leftrightarrow \left(\sum\nolimits_{k=1,9}\left(\left(\mathrm{range_k}(\mathrm{A_i}) \cap \mathrm{range_k}(\mathrm{A_i}) \neq \varnothing\right) = 1\right) > 7\right) \quad (7.1)$$

Example 7.1: Let consider classes VPC and LBBB.
$\Sigma_{k=1,9}((\mathrm{range_k}(\mathrm{VPC}) \cap \mathrm{range_k}(\mathrm{LBBB}) \neq \varnothing) = 1) = 8$, so that (VPC **DNT** LBBB).
End of example 7.1.
Matroid from Table 4 has 5 two elements bases:
B1 = {APC, RBBB}, B2 = {APC, LBBB}, B3 = {VPC, NORM},
B4 = {RBBB, NORM}, B5 = {LBBB, NORM}.

(iv) The induction of an emergent process is executed by the extension of some of matroid bases from **MB** by **AX**.
For operation with representation of ***AX1*** we use expression (7.1) (however with one point range(***AX1***)).

Table 4. Qualitative matrix of relation DNT

Ai	APC	VPC	RBBB	LBBB	NORM	AX1	AX2
APC	–	1	0	0	1	0	1
VPC		–	1	1	0	0	1
RBBB			–	1	0	0	1
LBBB				–	0	0	0
NORM					–	0	1
AX1						–	*
AX2						*	–

The extension of some basis B1, …, B5 is productive one only for B1 and B2:

$$\mathrm{B1AX1} = \{\mathrm{APC},\ \mathrm{RBBB}, \boldsymbol{AX1}\}, \tag{7.2}$$

$$\mathrm{B2AX1} = \{\mathrm{APC},\ \mathrm{LBBB}, \boldsymbol{AX1}\}. \tag{7.3}$$

The prompter of interpretation of ***AX1*** leads human solver to semantic content absolutely different than have classes APC and RBBB (or APC and LBBB): Indicates situation which does not increase danger of the heart failure, infarct of myocard, heart death, atrial flutter or atrial fibrillation.

(v) In case when ***AX*** is not able to extend any of matroid bases then ***AX*** is interpreted by "similar" objects from ${}^{j+1}\mathbf{SM}$.
Signal ***AX2*** may be interpreted by classes APC, VPC and RBBB – see Table 4. It means that is necessary to respect possibilities of danger of heart failure, unexpected heart death, infarct of myocard, ventricular tachycardia, atrial flutter or atrial fibrillation.

8 Conclusion

A simple method that computes a possible appearance of emergence and emergent phenomenon for one special Structural Invariant (Matroid and Base of the matroid) has been introduced.

Anticipating the fact that emergent phenomena are induced by sharp changes of complex system structure there have been investigated in the paper some of such changes (represented here, e.g., by deformation transformations).

The problems of interpretation of unknown objects were introduced.

There was used an ECG model based in the approach of linear discrimination analysis, the semiotic model of communication (Sign model, Interpretation space and Connections). In this paper is presented a principle of the method (not a statistical report about its application).

The presented approach is slightly more difficult from the conceptual point of view than from the side of operation procedures and does not necessitates time consuming computations.

Acknowledgements. This work was supported by the project SGS15/189/OHK2/3T/12. This support is very gratefully acknowledged.

References

1. Bila, J., Mironovova, M., Rodríguez, R., Jura, J.: Detection of emergent situations in complex systems by violations of structural invariants on algebras of transformations. Int. J. Enhanc. Res. Sci. Technol. Eng. **4**(9), 38–46 (2015)
2. Bila, J.: Processing of emergent phenomena in complex systems. Int. J. Enhanc. Res. Sci. Technol. Eng. **3**(7), 1–17 (2014)
3. Bila, J.: Algebras of transformations in the detection of unexpected situations of UX3 type. In: MENDEL 2010: 16th International Conference on Soft Computing, Brno, CR, vol. 2010, pp. 495–501 (2010)
4. Bila, J.: Emergent phenomena in natural complex systems. In: Sanayei, A., Zelinka, I., Rössler, Otto E. (eds.) ISCS 2013: Interdisciplinary Symposium on Complex Systems. ECC, vol. 8, pp. 89–100. Springer, Heidelberg (2014). doi:10.1007/978-3-642-45438-7_9
5. Bila, J., Bukovsky, I.: Modelling and interpretation of new solution in problem solving. In: ICCC 2011: 12th International Carpathian Control Conference, Velke Karlovice, CR, pp. 22–27
6. Ellis, G.F.R.: Top-down causation and emergence: some comments on mechanisms (2007). http://www.mth.uct.ac.za/~ellis/cos0.html. Accessed 11 Sep 2007
7. Reid, R.G.B.: Biological Emergences: Evolution by Natural Experiment. A Bradford Book. The MIT Press, Cambridge (2007)
8. ECG: Electrocardiography, Methods, Measurements and Instrumentation (In Czech). http://gerstner.felk.cvut.cz/biolab/X33BMI
9. Christov, I., Gómez-Herrero, G., Krasteva, V., Jekova, I., Gotchev, A., Egiazarian, K.: Comparative study of morphological and time-frequency ECG descriptors for heartbeat classsification. Med. Eng. Phys. **28**, 876–887 (2006)
10. Chazal, P., O′Dwyer, M., Reilly, R.B.: Automatic classification of heart-beats using ECG morphology and Heartbeat interval features. IEEE Trans. Biomed. Eng. **51**, 1196–1206 (2004)
11. Javadi, M., Asghar, S.A., Sajedin, A., Ebrahimpour, R.: Classification of ECG arrhythmia by a modular neural network based on mixture of experts and negatively correlated learning. Biomed. Sig. Process. Control **8**(3), 289–296 (2013)
12. Mironovova, M., Bila, J.: Fast fourier transform for feature extraction and neural network for classification of electrocardiogram signals. In: FGCT 2015: The Fourth International Conference on Future Generation Communication Technologies, Luton, GB, pp. 112–117 (2015)
13. Rodríguez, R., Mexicano, A., Bila, J., Nghien, N.B., Ponce, R., Cervantes, S.: Hilbert-huang transform and neural networks for electrocardiogram modeling and prediction. In: ICNC 2014: 10th IEEE International Conference on Natural Computation, Xiamen, pp. 561–567 (2014)
14. Shing-Tai, P.: ECG signal analysis by using hidden Markov model. In: iFuzzy: 2012 IEEE International Conference on Fuzzy Theory and its Application, Taichung, 288–293 (2012)
15. Liang, W., Zhang, Y., Tan, J., Li, Y.: A novel approach to ECG classification based upon two-layered HMMs in body sensor networks. Sensors **14**, 5994–6011 (2014)
16. Yeh, Y.C., Wang, W.J.: QRS complexes detection for ECG signal: the difference operation method. Comput. Methods Program Biomed. **91**, 245–254 (2008)

17. Yeh, Y.C., Wang, W.J., Chiou, C.: Cardiac arrhythmia diagnosis method using linear discrimination analysis on ECG signals. Measurement **42**, 778–789 (2009)
18. Oxley, J.G.: Matroid Theory. Oxford Science Publications, Oxford (2001)
19. Weinstein, W.: Ramsey Number. A Wolfram Web Resource (2004). http://mathword.wolfram.com/RamseyNumber.html. Accessed 5 Sep 2004

Author Index