Advances in Intelligent Systems and Computing 674

Alexandru-Adrian Tantar
Emilia Tantar
Michael Emmerich
Pierrick Legrand
Lenuta Alboaie
Henri Luchian *Editors*

EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI

Springer

Advances in Intelligent Systems and Computing

Volume 674

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.

Advisory Board

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

Alexandru-Adrian Tantar
Emilia Tantar · Michael Emmerich
Pierrick Legrand · Lenuta Alboaie
Henri Luchian
Editors

EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI

Springer

Editors
Alexandru-Adrian Tantar
Computer Science and Communications Research Unit
University of Luxembourg
Luxembourg
Luxembourg

Emilia Tantar
Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg
Luxembourg
Luxembourg

Michael Emmerich
Leiden Institute of Advanced Computer Science
Leiden University
Leiden
The Netherlands

Pierrick Legrand
Bâtiment Leyteire, URF Sciences et Modelisation
Université Bordeaux
Bordeaux
France

Lenuta Alboaie
Faculty of Computer Science
Alexandru Ioan Cuza University of Iasi
Iaşi
Romania

Henri Luchian
Faculty of Computer Science
Alexandru Ioan Cuza University
Iasi
Romania

ISSN 2194-5357 ISSN 2194-5365 (electronic)
Advances in Intelligent Systems and Computing
ISBN 978-3-319-69708-6 ISBN 978-3-319-69710-9 (eBook)
https://doi.org/10.1007/978-3-319-69710-9

Library of Congress Control Number: 2012944264

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

The overarching goal of the EVOLVE international conference series is to build a bridge between probability, statistics, set-oriented numerics, and evolutionary computing, as to identify new common and challenging research aspects and solve questions at the cross sections of these fields. There is a growing interest for large-scale computational methods with robustness and efficiency guarantees. This includes the challenge to develop sound and reliable methods, a unified terminology, as well as theoretical foundations.

The 2015 edition of the EVOLVE conference was held on June 14–18, in Iaşi, Romania, in conjunction with ECODAM Summer School on Evolutionary Computing in Optimization and Data Mining. The aim of the conference is to provide a bridge between probability, set-oriented numerics, and evolutionary computation and to bring together experts from these disciplines. This was reflected by the elaborate panel of invited speakers, namely Sorin Istrail (Brown University), Dan Simovici (University of Massachusetts Boston), Kalyanmoy Deb (Michigan State University), Carlos Coello Coello (CINVESTAV-IPN, Mexico City), and Kenneth De Jong (George Mason University) and tutorials by Pierre Del Moral (University of New South Wales), Iryna Yevseyeva (Newcastle University, UK), Michael Emmerich (Leiden University), and Madalina Drugan (Vrije Universiteit Brussels), all being leading experts in their field. The broad focus of the EVOLVE conference made it possible to discuss the connection between these related fields of study in computational science. The selected papers published in the proceedings book have been peer reviewed by an international committee of reviewers (at least three reviews per paper) and have been revised and enhanced by the authors after the conference. The contributions are categorized into five major parts, which are as follows:

- Multicriteria and Set-Oriented Optimization;
- Evolution in ICT Security;
- Computational Game Theory;
- Theory on Evolutionary Computation;
- Applications of Evolutionary Algorithms.

The 2015 edition shows a major progress in the aim to bring disciplines together, the research on a number of topics that have been discussed in previous editions of the conference matured over time, and methods have found their ways in applications. In this sense, the book can be considered an important milestone in this ongoing research effort.

June 2015

Alexandru-Adrian Tantar
Michael Emmerich
Emilia Tantar
Lenuta Alboaie
Henri Luchian
Pierrick Legrand

Organization

EVOLVE 2015 was organized by the Alexandru Ioan Cuza University of Iasi, Romania, in cooperation with the Leiden University, the Netherlands, the University of Luxembourg, and the University of Bordeaux, Inria Bordeaux Sud-Ouest.

Executive Committee

General Chair

Henri Luchian	Alexandru Ioan Cuza University of Iasi, Romania

Local Chair

Lenuta Alboaie	Alexandru Ioan Cuza University of Iasi, Romania

Proceedings Chair

Alexandru Tantar	University of Luxembourg, Luxembourg

Program Chairs

Michael Emmerich	Leiden University, The Netherlands
Pierrick Legrand	University of Bordeaux, France

Financial Chair

Corina Forascu	Alexandru Ioan Cuza University of Iasi, Romania

Tutorial Chair

Emilia Tantar	University of Luxembourg, Luxembourg

Advisory Board

Enrique Alba	University of Málaga, Spain
François Caron	Inria Bordeaux Sud-Ouest, France
Frédéric Ciérou	Inria Rennes Bretagne Atlantique, France
Carlos A. Coello Coello	CINVESTAV-IPN, Mexico
Michael Dellnitz	University of Paderborn, Germany
Frédéric Guinand	University of Le Havre, France
Arnaud Guyader	Université Rennes 2, Inria Rennes Bretagne Atlantique, France
Arturo Hernández-Aguirre	CIMAT, Guanajuato, Mexico
Günter Rudolph	TU Dortmund, Germany
Marc Schoenauer	Inria Saclay – Île-de-France, University Paris Sud, France
Franciszek Seredynski	Polish Academy of Sciences, Warsaw, Poland
El-Ghazali Talbi	Polytech'Lille University of Lille 1, France
Marco Tomassini	University of Lausanne, Switzerland
Massimiliano Vasile	University of Strathclyde, UK

Special Sessions Chairs

Vito Basto Fernandes	Leiria University, Portugal
Iryna Yevseyeva	Newcastle University, UK
Michael T.M. Emmerich	Leiden University, The Netherlands
Rodica Ioana Lung	University Babes-Bolyai, Romania
Dan Dumitrescu	University Babes-Bolyai, Romania
Jing Liu	Xidian University, China
Asep Maulana	Leiden University, The Netherlands
Lenuta Alboaie	Alexandru Ioan Cuza University of Iasi, Romania
Adrian Iftene	Alexandru Ioan Cuza University of Iasi, Romania
Marc Eduard Frincu	University of Southern California, USA
Stephane Genaud	ENSIIE Engineering School, France
Daniela Zaharie	West University of Timisoara, Romania

Mihaela Breaban — Alexandru Ioan Cuza University of Iasi, Romania
Madalina Ionita — Alexandru Ioan Cuza University of Iasi, Romania
Dragos Gavrilut — Alexandru Ioan Cuza University of Iasi, Romania

Program Committee

Series Chairs

Pierre del Moral — Inria Bordeaux Sud-Ouest, France
Alexandru Tantar — University of Luxembourg, Luxembourg
Emilia Tantar — University of Luxembourg, Luxembourg
Michael Emmerich — Leiden University, The Netherlands
Pierrick Legrand — University of Bordeaux, France

All contributions went through a thorough full-paper peer review process. We thank the referees for their voluntary effort.

Referees

Josiah Adeyemo
Thomas Baeck
Vitor Basto Fernandes
Francisco Chicano
Tudor Dan Mihoc
Luis Gerardo De La Fraga
Andre Deutz
Jianguo Ding
Dan Dumitrescu
Enrique Dunn
Michael T.M. Emmerich
Vitor Basto Fernandes
Francisco Fernandez
Marc Eduard Frincu
Edgar Galvan
Noemi Gasko
Stephane Genaud
David Iclanzan
Adrian Iftene
Ahmed Kattan
Joanna Kolodziej
Pierrick Legrand
Rui Li
Jing Liu
Francisco Luna
Rodica Lung
Asep Maulana
James McDermott
Nicolas Monmarche
Sanaz Mostaghim
Reka Nagy
Gustavo Olague
Eunice Ponce-De-Leon
Eduardo Rodriguez-Tello
Christoph Schommer
Ignacio Segovia-Dominguez
Ofer Shir
Mihai Suciu
Alexandru-Adrian Tantar
Emilia Tantar
Leonardo Trujillo
Sergio Ivvan Valdez
Hao Wang
Fatos Xhafa
Iryna Yevseyeva
Daniela Zaharie
Zhiwei Zhang

Invited Speakers

Kalyanmoy Deb (Koenig Endowed Chair Professor, Michigan State University, East Lansing, USA): A Theory-Based Termination Condition for Convergence in Real-Parameter Evolutionary Algorithms

Prof. Pierre Del Moral (University of New South Wales, Australia): Particle methodologies: a bridge across mathematics, physics, biology and information theory

Prof. Kenneth De Jong (George Mason University, USA): High-Performance Evolutionary Algorithms

Prof. Dr. Carlos Artemio Coello Coello (University of Massachusetts Boston, USA): Problems in Evolutionary Multiobjective Optimization, CINVESTAV, Mexico

Prof. Dr. Dan Simovici (University of Massachusetts Boston, USA): Exploring the Graph of the Web

Prof. Sorin Istrail (Brown University, USA): Applications to Medical Genetics and Computational Genomics

Invited Tutorials

Dr. Iryna Yevseyeva (School of Computing Science, Newcastle University, UK): Multicriteria Decision-Aiding: Compensating and Non-compensating Methods

Dr. Madalina Drugan (Vrije Universiteit Brussel, Belgium): Evolutionary Reinforcement Learning or Reinforcement Evolutionary Algorithms?

Dr. Michael T.M. Emmerich (LIACS, Leiden University): Set-Oriented Multicriteria Optimization: Deterministic and Stochastic Methods

Sponsoring Institutions and Partners

ALEXANDRU IOAN CUZA
UNIVERSITY of IAȘI

Contents

Computational Game Theory

Theory on Evolutionary Computation

Applications of Evolutionary Algorithms

Multicriteria and Set-Oriented Optimization

Aggregate Selection in Multi-objective Biochemical Optimization via the Average Cuboid Volume Indicator

Susanne Rosenthal[1], Bernd Freisleben[2], and Markus Borschbach[1(✉)]

[1] University of Applied Sciences, FHDW, Hauptstr. 2, Bergisch Gladbach, Germany
{Susanne.Rosenthal,Markus.Borschbach}@fhdw.de
[2] University of Marburg, Hans-Meerwein-Str. 6, Marburg, Germany
freisleb@informatik.uni-marburg.de

Abstract. The identification of peptides that optimize several physiochemical properties is an important task in the drug design process. Multi-objective genetic algorithms are efficient and cost-effective methods to scan the highly complex search space for optimal candidate peptides. A multi-objective genetic algorithm called NGSA-II has been proposed in previous work with the aim of producing diverse high-quality peptides in a low number of generations. An important component of NSGA-II is the selection process that determines the high-quality individuals for the succeeding generation. This paper presents two kinds of selection strategies for NSGA-II to guide the search process towards high-quality peptides while maintaining diversity within the genetic material. The proposed selection strategies rely both on tournaments, and use a combination of fitness-proportionate selection and a discerning selection criterion, which is front-based in one case and indicator-based in the other case. The two strategies are compared to each other with respect to the search behavior on a generic three-dimensional molecular minimization problem.

Keywords: Indicator-based selection · Average cuboid volume · Aggregate selection · Multi-objective molecular optimization

1 Introduction

Peptides play a central role in the area of drug design due to their high specificity and low toxicity profile. In the drug design process, native peptides or promising protein fragments are transformed into pharmaceutically acceptable components that have several optimized physiochemical properties [1].

A computer-aided drug design process is cost-effective and time-efficient. For peptide optimization, a customized multi-objective genetic algorithm (MOEA) called Non-dominated Sorting Genetic Algorithm (NSGA-II) has been proposed with sophisticated mutation and recombination methods [12,13]. Although these

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_1

variation operators have a considerable influence on the algorithm's performance, the selection operator is the main component that scans the search space and guides the search process [2]. In general, a good selection strategy is characterized by an appropriate balance between exploration and exploitation in the search process. Bäck [3] has shown that the selection process controls this balance by guiding the process more or less in the direction of optimal solutions. A search process is either more exploitative in the case that the search is directed stringently towards an optimum solution or more explorative otherwise.

In this paper, we present novel selection strategies for NSGA-II applied to molecular optimization. The design of the presented selection strategies is motivated by the fundamental tasks of the selection operator on guiding the search in a MOEA. Two selection strategies are presented. Both are based on tournament selection and a combination of fitness-proportionate selection and a discerning selection criterion, which is front-based in the first selection strategy and indicator-based in the other one. The Average Cuboid Volume (ACV) indicator is used as the discerning selection criterion in the indicator-based selection strategy and has been recently introduced as a convergence indicator with the ultimate aim of comparing solution sets of different sizes according to the proximity to the Pareto front in a statistically reasonable way [23]. The probability of selecting individuals by fitness-proportionate selection or the discerning criterion is controlled via a probability parameter. These two strategies are compared to each other to characterize their search behavior. The investigation of the selection performance also includes fine-tuning of the parameters. This comparison is based on a generic three-dimensional molecular minimization problem.

This paper is structured as follows. In Sect. 2, related work is discussed. The discerning selection criterion and the convergence indicator ACV are presented in Sect. 3. In Sect. 4, the novel selection strategies are introduced. Section 5 presents the remaining components of the customized NSGA-II. In Sect. 6, experimental results are presented. Section 7 concludes the paper and outlines areas of future work.

2 Related Work

Different selection strategies have been proposed in the field of Evolutionary Algorithms (EA). Fitness-proportionate selection strategies are stochastic methods. These strategies are characterized by a non-zero probability of each solution to be selected for reproduction: High-quality solutions have a higher chance to be selected than low-quality ones. The most common strategies are Roulette Wheel Selection (RWS) [4] and Stochastic Universal Sampling (SUS) [17]. These strategies are usually visualized by a spinning wheel where each portion on the wheel represents an individual and the size of the portion is proportional to the individual's fitness over the total fitness of the entire population. RWS selects one individual by spinning the wheel, whereas in the case of SUS n equidistant pointers are placed around the wheel and n individuals are selected per spin. A more common selection strategy is Tournament Selection (TS) [5]. The motivation for

this strategy is the diversity within the genetic material ensured by change. Several individuals are randomly selected from the population and compared to each other regarding their fitness. The best individual is selected for reproduction. The rank-based selection strategy assigns a selection probability to each individual based on the discrete rank relative to the others in the entire population [6]. This probability assignment is realized via a mapping function that is optionally linear or non-linear. The selection performance strongly depends on the type of the mapping function. A more recent category are the indicator-based selection strategies. These strategies make use of an indicator as the selection criterion and have been proposed in the area of Multi-objective Evolutionary Algorithms (MOEA): The Indicator-Based Evolutionary Algorithm (IBEA) [8] makes use of a selection strategy that starts with the selection of the fittest individual and deletes it from the population. The fitness values of the remaining individuals are updated by a binary quality indicator, such as the epsilon-indicator I_ϵ, the hypervolume indicator I_{HV} [8] or the R^2-indicator [9]. In general, an arbitrary indicator is applicable in IBEA. The indicator R^2 is a recently introduced indicator to evaluate the optimal solution set [10]. R^2 does not require a Pareto optimal reference set, but it depends on weight vectors and an ideal reference point. It is popular for its low computational complexity, but for an increasing number of objectives the calculation becomes as expensive as the hypervolume [11]. The number of weight vectors is challenging, especially scaling them with the number of objectives. R^2 and the hypervolume are correlated by Pearsons' correlation coefficient with a statistically significant value of 0.76 [11]. Another indicator-based selection strategy is used in the S-metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA) [7]. SMS-EMOA is a steady-state algorithm and is especially designed to use the hypervolume indicator. The hypervolume indicator serves as the selection criterion: In each iteration, a new individual is produced, and based on the value of the hypervolume it is decided if this individual enters the non-dominated archive pool or not by calculating the hypervolume of non-dominated solution subsets excluding one of the non-dominated solutions. SPEA [19], PESA [20] and PAES [21] use region-based selection. SPEA also makes use of binary tournament selection. The individuals for the succeeding generation are selected from the union of the current and an external set containing all non-dominated solutions. The selection probability of an individual depends on a strength value that reflects the number of individuals dominated by or equal to this individual. The individual-based selection in PESA divides the objective space into hyperboxes. The selection probability of an individual depends on a squeeze factor that is the number of individuals sharing the same box. Binary tournament selection is used and the individual with the lowest squeeze factor is chosen. PAES is a (1+1) evolutionary strategy and uses an archive pool for selection and the hypergrid strategy. The selection is performed between a current solution and a mutant regarding the dominance. If the mutant enters the archive pool, the individual with the highest grid location count is deleted.

3 The Average Cuboid Volume Indicator

The ACV indicator has been introduced by Rosenthal and Borschbach [23]. ACV is intended to evaluate the global convergence behavior of differently sized populations for comparison purposes. The ACV indicator is given by

$$ACV = \frac{1}{n}\sum_{i=1}^{n}(\prod_{j=1}^{k}(x_{ij} - r_j)), \tag{1}$$

where n is the number of individuals that are evaluated, k the number of objectives and r_j the pre-defined reference point. At this point, it is assumed that the Multi-Objective Optimization Problem (MOP) has to be minimized. In this case, the pre-defined reference point is chosen as the theoretical minimum limit of the true Pareto front, which is usually known in a real-world MOP. As a consequence, the lower the ACV indicator values are, the better is the global convergence behavior of the evaluated solution set.

There are three main advantages of this indicator. First, ACV does not require knowledge of the true Pareto front, which is usually unknown in real-world problems. Second, ACV has a low computational complexity even if the number of objectives increases. Third, ACV reflects the convergence behavior of differently sized populations in a statistically reasonable way.

A normalized version of the ACV indicator is proposed below to ensure that all objective function values have the same influence on the indicator values. Therefore, a mapping of the objective values in the same range of $[0;1]$ is performed by dividing every difference of the objective value and the corresponding reference point component by the maximum norm:

$$ACV_{scaled} = \frac{1}{n}\sum_{i=1}^{n}\left(\prod_{j=1}^{k}\frac{(x_{ij} - r_j)}{\bar{x}_j}\right), \text{ with } \bar{x}_j = max_i\{x_{ij}\}, \ \forall j = 1, ..., k \tag{2}$$

Furthermore, the ACV indicator is also used to gain an insight into the spatial volume covered by the optimal solutions relative to the volume of the entire population. Therefore, a relative ACV measure is proposed to evaluate the average cuboid volume by the solutions of the non-dominated solutions or the individuals of the first front in relation to the average cuboid volume of the entire population:

$$ACV_{rel} = \frac{\frac{1}{f}\sum_{i=1}^{f}(\prod_{j=1}^{k}(x_{ij} - r_j))}{\frac{1}{n}\sum_{i=1}^{n}(\prod_{j=1}^{k}(x_{ij} - r_j))}, \tag{3}$$

where f is the number of non-dominated solutions in the population. ACV_{rel} is quite different in terms of its significance from the hypervolume indicator, since the quality of the non-dominated solutions is not related to the quality of the entire population by the hypervolume, which usually refers only to the non-dominated solutions. However, Pearson's correlation coefficient between the

ACV_{rel} and the hypervolume values is statistically significant with a value of 0.6. A very small value of ACV ($ACV \approx 0$) indicates that ACV of the first front is much smaller than ACV of the whole population. In the case of $ACV_{rel} \approx 1$, the ACV value of the first front is relatively high compared to the ACV value of the whole population. A further interpretation of the relative ACV values has to take into account the absolute ACV of the whole population.

4 Selection Strategies for NGSA-II

There are several issues when designing an appropriate selection strategy for a MOEA in biochemical optimization. The first issue refers to the question of how to guide the search in the direction of the Pareto optimal solutions. The second issue is to ensure a high spread of the non-dominated solutions. The third issue refers to the specific purpose of biochemical optimization: The selection has to ensure a high diversity of the genetic material inherited to the succeeding population. The high diversity of the genetic material supports the global search process. Ideally, the selection strategy has to comply with these three issues at the same time. Furthermore, another component is important for the selection process especially in the field of molecular optimization. The role of change in the selection procedure imitates the aspect of change in a natural evolutionary process. The proposed two selection strategies for NSGA-II differ from the traditional selection process of NSGA-II in avoiding the critical component crowding distance to provide a reliably good algorithm performance independent of the problem dimension.

4.1 Aggregate Selection

The aggregate selection strategy is motivated by the idea of guiding the search in the direction of high-quality solutions while maintaining a high diversity of the genetic material within the succeeding generation. This strategy starts with tournament selection of x individuals from the population (Fig. 1) to include the aspect of change of a natural evolutionary process in the selection strategy. The solutions of the tournament set are ranked into fronts via fast non-dominated sorting [18]. From this ranked tournament set, one individual is randomly chosen from the first front with a probability p_0 to guide the search process in the direction of high-quality solutions with a particular probability. With a probability $1 - p_0$, individuals are selected via front-based Stochastic Universal Sampling (SUS) selection to ensure the diversity of the genetic material and a non-dominated solution spread. SUS provides the opportunity to low quality solutions to find their way in the succeeding generation. Low-quality solutions potentially have high-quality genetic motifs, which produces high-quality solutions in later generations. The number N of pointers is the number of identified fronts by fast non-dominated sorting, and the segments are equal to the front size. Therefore, the parameters of this selection strategy are the tournament size (t.s.) and the probability of choosing the individuals from the first front. The value $p_0 = 0\%$ is further referred to as SUS selection.

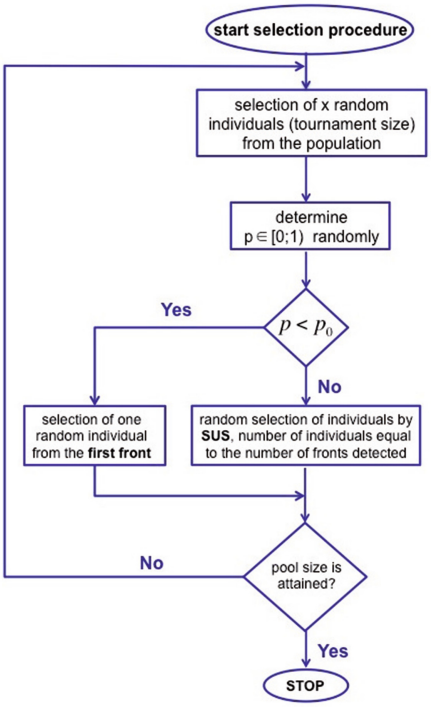


Fig. 1. Aggregate selection strategy

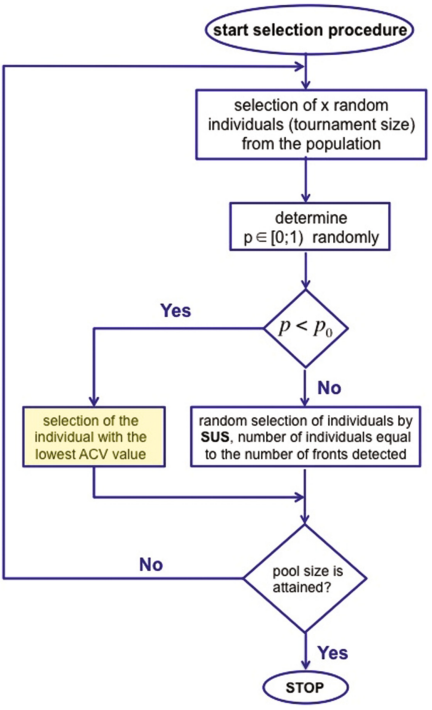


Fig. 2. ACV-based selection strategy with SUS.

4.2 ACV-Based Selection

The ACV-based selection strategy (Fig. 2) is equal to aggregate selection where the selection criterion - an individual is selected from the first front - is substituted by an ACV-based selection criterion. The basic idea of the ACV-based selection criterion is motivated by the following consideration of the aggregate selection strategy: Random selection of one individual from the first front does not guarantee the selection of a high-quality individual with respect to all objective values, since the ranking into the first front is relative to the objective values of other individuals in the tournament set. An ACV_{scaled} value for each individual of the tournament set is determined and the individual with the lowest ACV_{scaled}-value is selected. The ACV-value of an individual x_0 is calculated by applying Eq. (2) on $X = \{x_0\}$ with $n = 1$. The selection criterion differing from aggregate selection is highlighted. The individual with the lowest ACV_{scaled}-value is the highest-quality solution and selected for reproduction. In the case of multiple lowest ACV_{scaled}-values, a random one is selected.

4.3 Computational Complexity

The selection components that are mainly responsible for the difference of the computational complexity (CC) between the aggregate and ACV-based selection are Non-Dominated Sorting (NDS) of the tournament set and determination of the ACV_{scaled}-values for each solution in the tournament set. In the following, k is the number of objective functions and N the t.s. The CC of NDS is $O(k \cdot N^2)$ [18]. The selection of the solutions with the lowest ACV_{scaled}-value starts with the determination of the maximum value for each objective: This takes $k \cdot (N-1)$ operations for comparison. Furthermore, $k \cdot N$ divisions are performed to complete the scaling. The calculation of ACV_{scaled} for a t.s. of N takes k subtractions and $(k-1)$ multiplications. The determination of the minimal ACV_{scaled}-value takes $(N-1)$ operations for comparison. In total, this procedure has a CC of $k \cdot (N-1) + k \cdot N + N \cdot (k + (k-1)) + (N-1) = 4kN - k - 1$ operations, which is a total CC of $O(k \cdot N)$ and therefore lower than the CC for NDS.

5 Other Components of NSGA-II

5.1 Individual Encoding and Search Space

The individuals represent peptides of length 20. The individuals are character strings of length 20 consisting of the 20 characters symbolizing the canonical amino acids. As a consequence, the search space has a complexity of 20^{20}. This encoding is the most intuitive way and it represents all feasible - and only feasible - solutions. Another advantage is that the biochemical objective functions make use of this character encoding. Therefore, this encoding does not require a conversion of the data format.

5.2 Variation Operators

Several mutation and recombination operators have been tested within NSGA-II [12–14]. The combination of recombination and mutation operators that achieved the best performance is the linear recombination operator 'LiDeRP' and the adaptation of the deterministic dynamic mutation of Bäck and Schütz [15]. The combination of LiDeRP and the deterministic dynamic mutation provides the most successful balance of exploitation and exploration of the search process.

The recombination operator LiDeRP varies the number of recombination points over the generations via a linearly decreasing function:

$$x_R(t) = \frac{l}{2} - \frac{l/2}{T} \cdot t, \tag{4}$$

which depends on the length of the individual l, the total number of the generations T and the index of the actual generation t.

The deterministic dynamic operator of Bäck and Schütz [15] determines the mutation probabilities via the following function with $a = 2$.

$$p_{BS} = (a + \frac{l-2}{T-1}t)^{-1}, \tag{5}$$

The mutation rate is bounded by $(0; \frac{1}{2}]$. The mutation rate of the first generation has been adapted to a lower starting mutation rate with $a = 5$. This is due the fact that the combination of a high mutation and recombination probability corresponds to a random creation of an individual.

5.3 Fitness Functions

Three fitness functions - also termed objective functions - are used in the biochemical minimization problem. These fitness functions are as generic as possible in the way that physio-chemical characteristics of peptides are usually calculated by the descriptor values of each amino acid. The biochemical functions work on the amino acid sequence or the primary structure. Two fitness functions are provided by the BioJava library [16]: The first function determines the Molecular Weight (MW) of each individual, since molecules for drug design have to provide a maximally low MW for a good membrane permeability. The second function is the Needleman-Wunsch algorithm that is used as a method for the global sequence alignment to a pre-defined reference individual. This algorithm refers to the common hypothesis that a high similarity between molecules refers to similar molecular properties. The third fitness function calculates the average hydrophilicity via the hydrophilicity scale of Hopp and Woods with a window size equal to the peptide length [22]. A common problem of drug peptides is a low solubility in aqueous solutions, especially peptides with stretches of hydrophobic amino acids. These fitness functions act comparatively, since the individuals are compared to a non-varying reference individual and therefore the MOP has to be minimized. The absolute value is applied to the fitness function values to achieve only positive function values.

6 Experimental Results

6.1 Experimental Setting

The starting population has a size of 100 randomly initialized individuals representing 20-mer peptides. For statistical reasons, each configuration is repeated 30 times until the 18th generation since we focus on early convergence [12,13]. These experiments are evaluated with regard to their convergence velocity and the diversity within the solutions. The ACV indicators ACV_{scaled} and ACV_{rel} as proposed in Eqs. (2) and (3) are used as convergence metrics for the entire population and the quality of the Pareto optimal set. The reference point is chosen as (0/0/0) which is the theoretical minimum limit of the Pareto front for the three-dimensional minimization problem. The diversity within the population is assessed via:

$$\Delta = \sum_{i,j=1,i<j} \frac{|d_{ij} - \bar{d}|}{N} \qquad \text{with} \quad N = \binom{n}{2} = \frac{n(n-1)}{2},$$

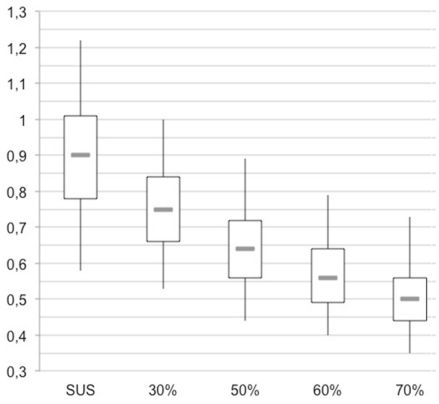


Fig. 3. ACV_{scaled} of aggregate selection with different p_0-values and $t.s. = 10$.

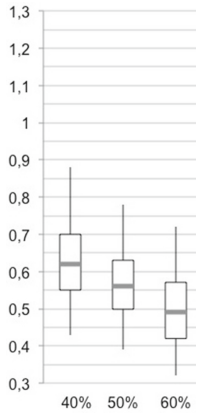


Fig. 4. ACV_{scaled} of ACV selection with different p_0-values and $t.s. = 10$.

where d_{ij} is the Euclidean distance of each possible combination of solutions. n is the number of solutions, here $n = 100$, and $\bar{d}$ is the average distance over all determined distances. The values of diversity are scaled under the same criterion for an optimal graphical presentation. Boxplots are created for the ACV and diversity values of each configuration to provide important information about location parameters and spread of the numerical data. Outliers are symbolized by dots. An outlier is more than 1.5-times of the inter-quartile range away from the boxplot quartiles. In general, a good performance of a configuration is achieved if the ACV value is as small as possible and the diversity is as large as possible.

6.2 Evaluation

Two categories of configurations are evaluated for each selection strategy: The first test series performs a variation of the selection parameter - the probability parameter p_0. Different probabilities are tested with the fixed tournament size parameter $t.s. = 10$. Figure 3 represents the effect of aggregate selection with different values of the probability p_0. The increase of p_0 results in a decrease of the ACV_{scaled}-values in differing intensity: The ACV_{scaled} decrease by an increase of p_0 from 0% to 50% is remarkably higher than the decrease intensity for a further probability increase. SUS or $p_0 = 0\%$ achieves a remarkably high spread of the ACV_{scaled}-values indicating an unstable selection process. Figure 5 represents the diversity achieved with the aggregation selection and different p_0-values. SUS achieves the highest diversity, the further increase of p_0 results in a decrease of the diversity. The comparison of Figs. 3 and 5 reveals that the convergence improvement is at the cost of diversity. The convergence improvement of $p_0 = 50\%$ to $p_0 = 60\%$ is, on the average, higher than the average decrease of the diversity values for the same p_0-value. Therefore, the optimal choice is configured to $p_0 = 60\%$. Figure 7 depicts the ACV_{rel}-values of the aggregate selection for

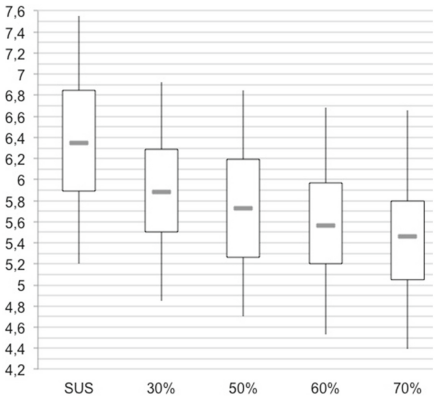


Fig. 5. Diversity of aggregate selection with different p_0-values and $t.s. = 10$.

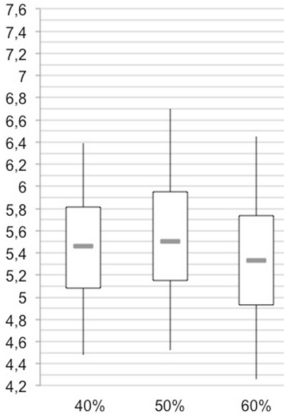


Fig. 6. Diversity of ACV-based selection with different p_0-values and $t.s. = 10$.

the different p_0-values. The increase of p_0 in this case results in a slight but continuous increase of the ACV_{rel}-values indicating that the ACV-values of the first front solutions are relatively high compared to the ACV-values of the entire population. These results reveal that the front-based selection of the tournament set does not guarantee the selection of the highest quality solutions. Figures 4, 6 and 8 depict the ACV_{scaled}, diversity and ACV_{rel}-values of the ACV-based selection with three different p_0-values. Figure 4 reveals that an increase of p_0 results in a significant decrease of ACV_{scaled}. The diversity values (Fig. 6) reveal, on the average, slight differences for the different p_0-values. The highest diversity values on the average are achieved for $p_{=}50\%$. The comparison of ACV_{scaled} and diversity values of the aggregate and ACV-based selection expose that the ACV_{scaled} and diversity values of the aggregate selection with $p_0 = 60\%$ are highly related to the corresponding values of the ACV-based selection with $p_0 = 50\%$. Figure 8 represents the ACV_{rel}-values of the ACV-based selection. The increase of $p_0 = 40\%$ to 50% results in an improvement of the ACV_{rel}-values. A further increase of p_0 reveals a stagnation of ACV_{rel}. This indicates that the increase of p_0 potentially provides the guarantee selecting the highest-quality solutions. In general, the ACV_{rel}-values of the ACV-based selection are lower than the corresponding values for the aggregate selection. Since aggregate selection with $p_0 = 60\%$ and ACV-based selection $p_0 = 50\%$ achieve the best performance according to convergence and diversity, the variation of the parameter t.s. is investigated for a further improvement. Figures 9 to 11 depict the ACV_{scaled}, diversity and ACV_{rel}-values of the test runs with the aggregate selection and different t.s. The increase of the t.s. results in a continuous decrease of ACV_{scaled} (Fig. 9) and therefore in an improved convergence performance. The higher solution number in the tournament set provides a higher-quality diversity between the solutions. The diversity reveals slight changes of the values by the increase of the t.s on the average; the diversity values are the highest for $t.s. = 10$ (Fig. 10). Otherwise,

the results of ACV_{rel} (Fig. 11) reveal that $t.s. = 10$ achieves the highest values and therefore the lowest quality non-dominated solutions on the average. In general, the optimal value for t.s. is a trade-off between a higher computational complexity in every iteration and the performance improvement. From this point of view, the optimal choice of t.s. is about 10 for the aggregate selection strategy.

Figures 12 to 14 present the ACV_{scaled}, diversity and ACV_{rel}-values of the test runs with the ACV-based selection and different t.s. The increase of the t.s. from 6 to 10 results in a slight decrease of ACV_{scaled} on the average (Fig. 12). A further increase to $t.s. = 15$, ACV_{scaled} almost stagnates. The increase of the t.s. reveals an improvement of the diversity, once a higher improvement is achieved by the increase of $t.s. = 6$ to 10 and a stagnation of the values by a further increase (Fig. 10). The increase of the t.s. from 6 to 10 reveals a significant increase of ACV_{rel} (Fig. 14) and therefore an improvement of the non-dominated solution quality. Otherwise, the results are skewed visibly by the position of the median. A further increase of the t.s. to 15 results only in a slightly decreased ACV_{rel}. The reason for these results is the low range of the solution quality in a tournament set of size 6. A further increase of the t.s. above 10 does not provide a further quality enhancement according to convergence and diversity, since the enhancement is restricted by the probability of selecting high-quality solutions from the entire population in the tournament set. The optimal value for the t.s. in the case of the ACV-based selection is easily accessible as $t.s. = 10$.

These performance results are compared to the performance of the traditional and character-encoded NSGA-II. The process of NSGA-II is slightly changed to preserve diversity: Binary tournament selection is used for parent selection, whereas each individual is allowed to be presented two times as parent. Further, single-point recombination is used and one-point mutation of each individual is performed. The individuals for the succeeding generation are selected by binary tournament selection with a preference for the solution with the largest crowding

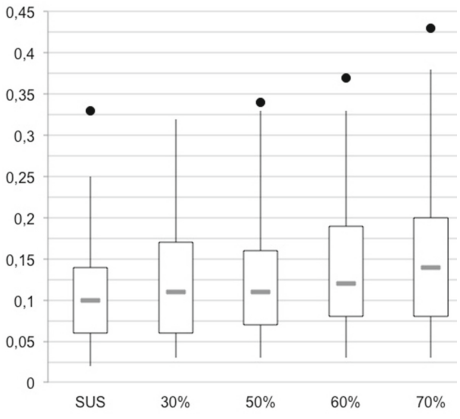


Fig. 7. ACV_{rel} of aggregate selection with different p_0-values and $t.s. = 10$.

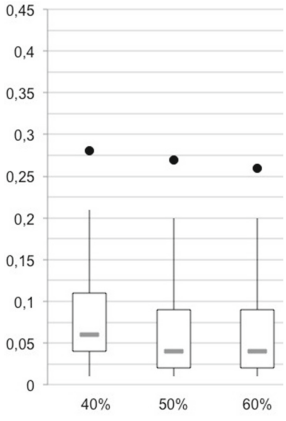


Fig. 8. ACV_{rel} of ACV-based selection with different p_0-values and $t.s. = 10$.

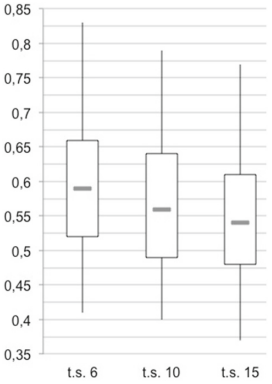


Fig. 9. ACV_{scaled} of aggregate selection with different t.s. values and $p_0 = 60\%$.

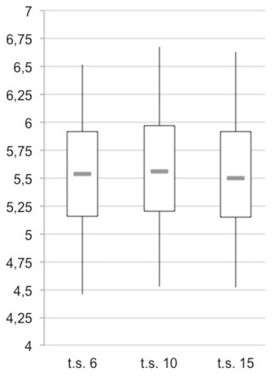


Fig. 10. Diversity of aggregate selection with different t.s. values and $p_0 = 60\%$.

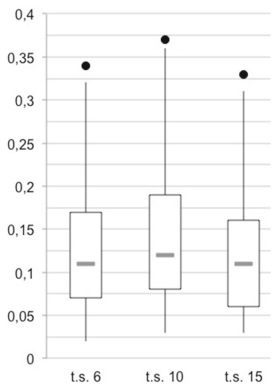


Fig. 11. ACV_{rel} of aggregate selection with different t.s. values and $p_0 = 60\%$.

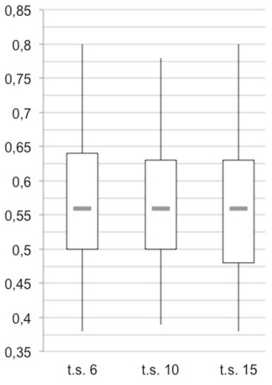


Fig. 12. ACV_{scaled} of ACV-based selection with different t.s. values and $p_0 = 50\%$.

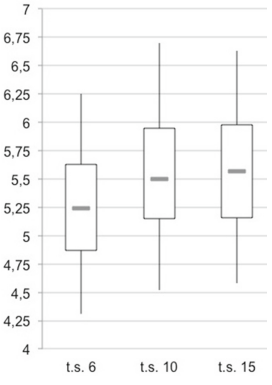


Fig. 13. Diversity of aggregate selection with different t.s. values and $p_0 = 50\%$.

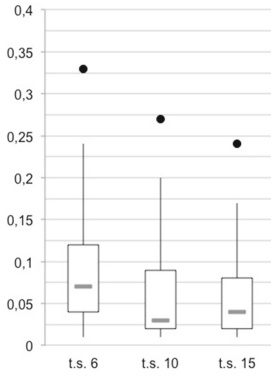


Fig. 14. ACV_{rel} of ACV-based selection with different t.s. values and $p_0 = 50\%$.

distance value. Figure 15 presents the performance results of this traditional NSGA-II configuration. Even with the diversity preserving method, the diversity values are the lowest compared to the performance results presented above (Figs. 10 and 13). Furthermore, the ACV_{scaled} values are remarkably higher and therefore, NSGA-II provides the worst convergence compared to the results of Figs. 9 and 12. The ACV_{rel} results are also the highest and therefore the non-dominated solutions provide are of a low quality compared to the entire population.

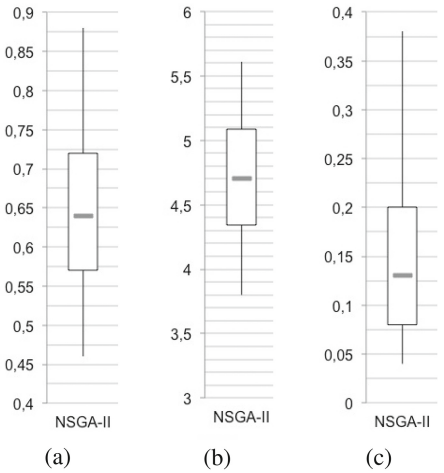


Fig. 15. Performance of NSGA-II with diversity preserving selection of the succeeding generation: (a) ACV_{scaled}, (b) Diversity and (c) ACV_{rel}.

7 Conclusion

This paper has presented two types of selection strategies for the determination of the succeeding generation in NSGA-II, a multi-objective genetic algorithm applied to biochemical optimization. The focus of the selection strategies is the guidance of the search process towards high-quality solutions while maintaining genetic diversity. Both strategies have in common that they are based on tournament selection and are a combination of SUS as fitness-proportionate selection and a discerning selection criterion. Aggregate selection uses a front-based strategy as the discerning selection criterion, whereas ACV-based selection makes use of the ACV indicator. In aggregate selection, the selection of a random individual from the first front does not guarantee the selection of the optimal solution with respect to each objective function. To overcome this problem, ACV-based selection is used to select the high-quality individuals in the multi-objective sense based on the objective function values. Aggregate selection is compared to ACV-based selection in terms of search behavior. In general, our evaluation reveals that both selection strategies are comparable according to the convergence behavior and the diversity, especially in the case of the optimum parameter setting: $p_0 = 60\%, t.s. = 10$ for aggregate selection and $p_0 = 50\%, t.s. = 10$. The advantage of ACV-based selection is the significantly higher quality, on the average, of the first front and therefore of the non-dominated solutions. The ACV indicator used as a discerning selection criterion guarantees a more focused search process in the direction of high-quality solutions.

The presented strategies have the potential to be applied to other molecular MOPs, since the proposed 3D-MOP is as generic as possible regarding the fitness assignment of physiochemical properties. In the case that the objective functions

are the absolute values of the difference between the fitness value of the candidate peptide and the fitness value of the reference peptide, the zero point is an advisable choice for the ACV indicator. Otherwise, the reference point has to be newly defined. The influence of the reference point has not been investigated in this work and is thus the limitation of this study.

In future work, the experiments will be performed on a higher-dimensional biochemical optimization problem. The investigation of the impact of selection strategies on the optimization performance of real-valued MOPs is also in interesting area of future work.

References

1. Otvos, L.: Peptide-Based Drug Design: Methods and Protocols. Humana Press Inc., Totowa (2000)
2. Dumitrescu, D., Lazzerini, B., Jain, L.C., et al.: Evolutionary Computation. CRC Press LLC, Boca Raton (2000)
3. Bäck, T.: Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In: First IEEE Conference on Evolutionary Computing, vol. 1, pp. 57–62 (1994)
4. Kumar, R.: Blending roulette wheel selection and rank selection in genetic algorithms. Int. J. Mach. Learn. Comput. **2**(4), 365–370 (2012)
5. Zhong, J., Hu, X., Gu, M., et al.: Comparison of performance between different selection strategies on simple genetic algorithms. In: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automations (2005)
6. Baker, J.: Adaptive selection methods for genetic algorithms. In: Proceedings of an International Conference on Genetic Algorithms, pp. 101–111 (1985)
7. Emmerich, M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. In: EMO 2005. LNCS, vol. 3410, pp. 62–76 (2005)
8. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature PPSN VIII, pp. 832–842 (2004)
9. Phan, D.H., Suzuki, J.: R2-Indicator based evolutionary algorithm for multiobjective optimization. In: Congress on Evolutionary Computation (CEC 2013), pp. 1836–1845 (2013)
10. Trautmann, H., Wagner, T., Brockhoff, D.: R2-EMOA: focused multiobjective search using R2-indicator-based selection. In: Learning and Intelligent Optimization, pp. 70–74 (2013)
11. Wagner, T., Trautmann, H., Brockhoff, D.: Reference articulation by means of the R2 indicator. In: Evolutionary Multi-criterion Optimization (EMO 2013), vol. 7811, pp. 81–95 (2013)
12. Rosenthal, S., El-Sourani, N., Borschbach, M.: Introduction of a mutation specific fast non-dominated sorting GA evolved for biochemical optimization. In: SEAL 2012. LNCS, vol. 7673, pp. 158–167 (2012)
13. Rosenthal, S., El-Sourani, N., Borschbach, M.: Impact of different recombination methods in a mutation-specific MOEA for a biochemical application. In: Vanneschi, L., Bush, W.S., Giacobini, M. (eds.) EvoBIO 2013. LNCS, vol. 7833, pp. 188–199 (2013)

14. Rosenthal, S., Borschbach, M.: A benchmark on the interaction of basic variation operators in multi-objective peptide design evaluated by a three dimensional diversity metric and a minimized hypervolume. In: Emmerich, M., et al. (eds.) EVOLVE - A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation IV, pp. 139–153 (2013)
15. Bäck, T., Schütz, M.: Intelligent mutation rate control in canonical genetic algorithm. In: Proceedings of the International Symposium on Methodology for Intelligent Systems, pp. 158–167 (1996)
16. BioJava: CookBook, release 3.0. http://www.biojava.org/wiki/BioJava
17. Fonseca C.M., Fleming P.J.: Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA 1993), pp. 416–423 (1993)
18. Deb, K., Pratap, A., Agarwal, S.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
19. Corne, D.C., Knowles, J., Oates, M.J.: The pareto envelope-based selection algorithm for multiobjective optimization. In: Proceedings of Parallel Problem Solving from Nature PPSN VI, vol. 1917, pp. 839–848 (2000)
20. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)
21. Knowles, J.D., Corne, D.W.: The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC 1999), pp. 98–105 (1999)
22. Hopp, T.P., Woods, K.R.: A computer program for predicting protein antigenic determinants. Mol Immunol **20**(4), 483–489 (1983)
23. Rosenthal, S., Borschbach, M.: Impact of population size and selection within a customized NSGA-II for biochemical optimization assessed on the basis of the average cuboid volume indicator. In: Proceedings of the Sixth International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies, BIOTECHNO 2014, Charmonix, April 2014

On Gradient-Based and Swarm-Based Algorithms for Set-Oriented Bicriteria Optimization

Wilco Verhoef(✉), André H. Deutz, and Michael T.M. Emmerich

Multicriteria Optimization, Design and Analytics (MODA) Group, LIACS, Leiden University, Niels Bohrweg 1, 2375 CA Leiden, The Netherlands
wilco@verhoef.nu, emmerix@gmail.com
http://moda.liacs.nl

Abstract. This paper is about the numerical solution of multiobjective optimization problems in continuous spaces. The problem is to define a search direction and a dynamical adaptation scheme for sets of vectors that serve as approximation sets. Two algorithmic concepts are compared: These are stochastic optimization algorithms based on cooperative particle swarms, and a deterministic optimization algorithm based on set-oriented gradients of the hypervolume indicator. Both concepts are instantiated as algorithms, which are deliberately kept simple in order to not obfuscate their discussion. It is shown that these algorithms are capable of approximating Pareto fronts iteratively. The numerical studies of the paper are restricted to relatively simple and low dimensional problems. For these problems a visualization of the convergence dynamics was implemented that shows how the approximation set converges to a diverse cover of the Pareto front and efficient set. The demonstration of the algorithms is implemented in Java Script and can therefore run from a website in any conventional browser. Besides using it to reproduce the findings of the paper, it is also suitable as an educational tool in order to demonstrate the idea of set-based convergence in Pareto optimization using stochastic and deterministic search.

Keywords: Hypervolume indicator · Pareto front · Set-oriented gradient · Particle swarm optimization · Multiobjective optimization · Algorithm animation

1 Introduction

Multi-objective optimization (MOO) is a class of optimization problems where multiple objective functions are optimized simultaneously. MOO problems are common in numerous fields including engineering, science, industry, drug discovery, finance, and logistics. Given this broad range of application areas, there is a big need for fast and reliable MOO algorithms.

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_2

In typical MOO problems, the objectives are conflicting. Thus, an optimal solution for one objective is not optimal for the others. Hence, with conflicting objectives, there is no single optimal solution for the problem. Instead, there is typically a whole set of solutions in the decision space that are non-dominated with respect to the Pareto dominance relation. We call this set the *efficient set*. The image of the efficient set obtained by the objective functions is the *Pareto front*. In continuous spaces they can be viewed as a trade-off curve (in 2-D) or a trade-off (hyper)surface (in higher dimensions).

The Pareto front of a function with m objectives is typically a manifold of at most $m - 1$ dimensions and it is not required to be connected. When approximating a Pareto front (and efficient set) by means of a finite approximation set it is a common strategy to search for sets that maximize the size of the dominated (hyper) space which is measured by the hypervolume indicator. In this paper two different strategies for finding hypervolume-maximal sets will be discussed.

For this we will introduce new algorithms for multi-objective optimization, namely a *multi-objective cooperative particle swarm optimization* MOCOPS algorithm and a *multi-objective gradient based optimization* MOGO algorithm (a modified version of a previously discussed set-based gradient strategy). Then we will provide a numerical analysis of the dynamics of these algorithms on bicriteria test problems.

The specific contribution of this research is three-fold. We will analyze the performance of the new algorithmic concepts by using the hypervolume indicator as a quality measure. Secondly this research will focus on an analysis and comparison of the algorithms based on their dynamical visualization (animation). Finally, we will provide a tool that the interested reader can use to further explore the proposed algorithmic concept in an interactive and easy to use web-application.

The structure of the paper as follows: In Sect. 2 we will first introduce the definitions and notation that are to be used in the remainder of the paper. Moreover we will provide a formal definition of the problem and review related work. In Sect. 3 the different multi-objective optimization algorithms are presented. In the Sect. 4 the dynamics of the algorithms on test problems will be compared. We sum up the main findings in Sect. 5 and provide some directions that will be interesting for future work. Instead of extensive statistical plots of repeated runs, we will provide the user with an easy to use javascript program which can be used in a web-browser to reproduce our results and gives the opportunity for self-study of the new algorithms. A description of this visualization tool in Appendices A will conclude the paper.

2 Background

2.1 Definitions and Notation

The space of candidate solutions is called the *decision space*. The space of objective function values with the decision space as domain is called the *objective space*.

For optimization, it is desirable to have an unambiguous way of determining whether an arbitrary vector is considered better than another. For this reason, you can define the relations *weakly-Pareto-dominance* and *strictly-Pareto-dominance* between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$. Weakly-Pareto-dominance is a relation between two vectors where one vector is considered better or equal than another. We will assume that our objective is minimization. Vector $\mathbf{x}$ weakly Pareto-dominates vector $\mathbf{y}$, if and only if Eq. (1) holds.

$$\forall_{i \in \{1,\ldots,m\}}\ \mathbf{x}_i \leq \mathbf{y}_i \tag{1}$$

A vector $\mathbf{x}$ is considered to *strictly Pareto-dominate* a vector $\mathbf{y}$ if and only if

$$\forall_{i \in \{1,\ldots,m\}}\ \mathbf{x}_i \leq \mathbf{y}_i \wedge \exists_{i \in \{1,\ldots,m\}}\ \mathbf{x}_i < \mathbf{y}_i$$

The strict Pareto-dominance is a strict order where strict dominance of $\mathbf{x}$ over $\mathbf{y}$ is denoted with $\mathbf{x} \prec \mathbf{y}$.

Let $\mathbb{S} \subseteq \mathbb{R}^d$ and $\boldsymbol{f} : \mathbb{S} \to \mathbb{R}^m$. In optimization problems $\mathbb{S}$ represents the decision space, and $\boldsymbol{f}$ the objective functions. The Pareto front of the minimization problem is the non-dominated subset of the image of $\mathbb{S}$ under $\boldsymbol{f}$. Formally, we define the Pareto front Y_{PF} in Eq. 2:

$$Y_{PF} := \{\boldsymbol{f}(\mathbf{x}) \mid \mathbf{x} \in \mathbb{S} \wedge \nexists_{\mathbf{x\prime} \in \mathbb{S}}\ \boldsymbol{f}(\mathbf{x\prime}) \prec \boldsymbol{f}(\mathbf{x})\} \tag{2}$$

The inverse image of the Pareto set with respect to $\boldsymbol{f}$ is called the *efficient set*.

The *hypervolume indicator* (HI) is a widely used measure for multi-objective optimization indicating how well the population approximates the Pareto Front [10]. More precisely, it is defined as the Lebesgue measure of the dominated subspace by a population of m- dimensional vectors in the objective space limited by a reference point (to keep it finite), in symbols:

$$HI(P) := \lambda_m \left(\bigcup_{\mathbf{y} \in P} [\mathbf{y}, \mathbf{r}] \right).$$

Here, λ_m denotes the Lebesgue measure in m dimensions, e.g. λ_1 is the length, λ_2 is the area, λ_3 is the volume, and so forth. The reference point is chosen such that it is dominated by all relevant objective vectors. The choice of the reference point should be large enough for it to be dominated by all points that are generated in the optimization.

In this work, we will often consider the contribution of a single point to the dominated hypervolume indicator. The *hypervolume contribution* of a vector in the objective space (objective vector) in a multiset of such vectors (population) is defined as the hypervolume of the total population of objective vectors, minus the hypervolume of the population without that objective vector. Objective vectors which are dominated by some vector in the population have a hypervolume contribution of 0. The concepts of (Pareto) dominance, hypervolume and hypervolume contribution are clarified for two dimensions in Fig. 1.

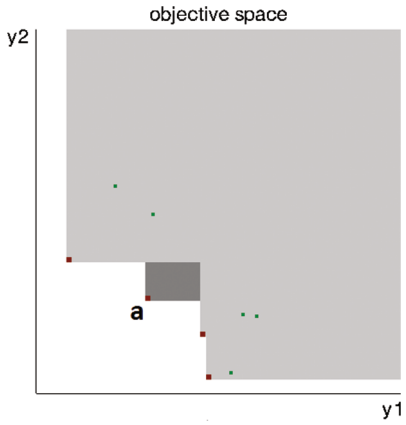


Fig. 1. This is an example of a population in the objective space. Objective vectors in the population are displayed as slightly bigger (as compared to dominated points), black, and filled circles in case they are not dominated by another objective vector in the population. Objective vectors in the population are displayed in dots when they are part of the dominated set. The set of points that is dominated by at least one point in the population is colored light gray. The hypervolume contribution of objective vector a is colored dark gray.

In the literature on multiobjective optimization algorithms, different terms with similar meanings are used depending on the field of research. Below is a short summary of interchangeable terms. With Table 1, an attempt has been made to match each field with each of their terms, however when reasoning in general about the algorithms, the terms used might be mixed together. However, we decided to use the terms *population*, *particles*, and *objective functions* to denote the essential entities in the discussed algorithms.

Table 1. Terminology in set-oriented optimization. The terms in bold font will be used throughout this paper.

EA	PSO	Gradient optimization
fitness (function)	fitness (function)	**objective function**
individual	**particle**	(search) point, (candidate) solution
population	swarm	approximation set, μd-vector (cf. [6])

Notation	Description
$\mathbb{S} \subseteq \mathbb{R}^d$	Decision space
I	Set of objective vectors, subset of $\mathbb{R}^m$
m	Dimension of the objective space
d	Dimension of the decision space
$\mathbf{x} \in \mathbb{S}$	Representation of a particle in the decision space
$\mathbf{y} \in \mathbb{R}^m$	Representation of a particle in the objective space
$\boldsymbol{f} : \mathbb{S} \to \mathbb{R}^m$	Vector of objective functions
μ	Population size
$\mathbf{a} \in \mathbb{S}$	Particle
$\Phi : \mathbb{S}^\mu \times \mathbb{S} \to \mathbb{R}$	Fitness contribution of an individual to a population
$HV : I^\mu \to \mathbb{R}$	Hypervolume indicator of a solution w.r.t. a population of size μ
$\Delta HV : I^\mu \times I \to \mathbb{R}$	Hypervolume contribution of a single solution in a population
$i \sim u(\{a, b, c\})$	Denotes a uniform random discrete sample $b \in \{a, b, c\}$
$x \sim u([0, 1])$	Denotes a uniform random continuous sample $x \in [0, 1]$
$\mathbf{z} \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$	Denotes a sample from the multivariate i.i.d. normal distribution with mean value 0 and variance 1.

2.2 Problem Definition

For optimization, minimization is assumed for all objectives throughout this work.

$$\boldsymbol{f}(\mathbf{x}) \to \min$$

The aspiration of the optimization algorithms in this paper is maximization of the hypervolume indicator over the set of all populations of size μ. That will be the measurement used for benchmarking:

$$HV(P) \to \max, P \in \mathbb{S}^\mu$$

For this the convergence dynamics of two different types of algorithms will be compared - swarm-based optimization and gradient-based optimization.

2.3 Related Work

The idea to maximize the hypervolume indicator using population based search was initiated first in the context of evolutionary multi-criterion optimization algorithms. For instance, the S-Metric Selection-EMOA (SMS-EMOA) is a popular algorithm in this field [5]. Here the term 'S-Metric' is an alternative name for the hypervolume indicator. In particular for small numbers of objective functions ($m = 2, 3$) this algorithm performs very well in comparison with other EAs, although recent research has shown that it does not always converge to the globally Pareto optimal front (which is also the case for most other EMOAs).

Swarm based hypervolume optimization has been suggested before by Mosthaghim et al. [13] in an algorithm oriented at the 'follow a leader' paradigm in classical particle swarm algorithm. In our paper we will completely

abandon this and present a cooperative search paradigm where each particle can contribute in equal proportions to the success of a population.

Especially in the 'fine tuning' phase of the set-based optimization on differentiable problems it is not very promising to use stochastic search algorithms with isotropic mutation operators. This is because the local tangent cone in which dominating solutions of a point can be found in the efficient space gets increasingly acute and the opening angles converge to zero. This is why sampling steps must be exactly in descent directions in order to get closer to the Pareto front. This is where gradient-based search presents itself as an interesting alternative to stochastic search.

Research about gradient optimization algorithms in MOO using the hypervolume indicator has been performed before [6]. In the research, computation complexity of the algorithms is reported for a varying numbers of dimensions. It was shown by Emmerich et al. [6] that essentially the hypervolume contribution gradients are the sub-gradient components of the entire population vector. While [6] focuses on the computation of the Hypervolume gradient and its general properties, its search dynamics were investigated on a couple of low dimensional problems in [6]. Here a simple set-based gradient and set-based Newton method was compared on bi-criterion problems. In our paper we will study an improved version of this algorithm that introduces a non-zero gradient for dominated points.

3 Optimization Algorithms

In this study we consider iterative (sequential) algorithms for Pareto optimization. They generate a series of populations P_0, P_1, ..., that (probabilistically) converge to the Pareto front. Such processes will be visualized and studied using a web-based interface.

On the website http://moda.liacs.nl/pareto-demo.html the two algorithms under investigation are implemented in `java script`, an interpreter language which features client side execution. The featured algorithms are a cooperative swarm based algorithm and a hypervolume gradient method.

As opposed to single objective optimization, in multiobjective optimization instead of a starting point a starting population needs to be provided for iterative optimization algorithms. Alternatively, we may initialize the population uniformly randomly within the search space or on points of a regular grid.

3.1 Multi-objective Particle Swarm Optimization Algorithm

In the interpretation we use in this paper a particle swarm optimization (PSO) algorithm is a randomized search heuristics where a swarm of particles moves gradually towards an optimal solution driven by randomized modification operators and interaction between the particles.

In conventional PSO algorithms, the swarm is driven by a leader, who is the currently best individual in a population, and by local memories of particles on

their so-far best positions. In single-objective optimization such processes will typically converge to local, or sometimes even to global optima. In multiobjective optimization such an approach could be easily used to find a single point on the Pareto front, but is not well suited to distribute points across the Pareto front, because the particles all strive to resemble the leader which is counterproductive when searching for a diverse set of solutions. To a certain extent this can be compensated by assigning local leaders, but this makes the algorithm quite complicated and adds parameters to the algorithm (i.e., number of leaders).

The use of traditional PSO for multi-objective optimization problems has been addressed already in the literature, both in the context of general multiobjective optimization [4], and for finding Pareto fronts that maximize the hypervolume indicator [13]. Both approaches let to algorithms that can produce good approximations to Pareto fronts.

In this paper, however, we consider another approach that we will term *cooperative particle swarm*. This algorithm will have the following properties that distinguishes it from previous swarm-based approaches:

- Leader-free: The particles in the population cooperate in covering the Pareto front, instead of competing with each other. There is no leader in the swarm; each particle strives to contribute to the global performance of the swarm.
- Indicator-based: The algorithms seeks to maximize an unary performance indicator. Here the hypervolume indicator is used, but also other unary indicators could be considered (e.g., *reference point free hypervolume* [8]).

The new approach is deliberately kept very simple. This is for two reasons: Firstly we want to demonstrate that only a few essential components are needed to steer a swarm towards a Pareto front. Secondly, simplicity will make the algorithms easier accessible to a rigorous theoretical analysis. It will also be easier to compare it on a conceptual level to the later discussed set-gradient based algorithm.

We will term the approach Multiobjective Optimization by Cooperative Particle Swarms (MOCOPS).

The pseudo-code for the proposed MOCOPS algorithm is given in Algorithm 1. It starts with randomly initializing a set of particles. Then, in each iteration of the algorithm, a particle is randomly selected and a small random variation of this particle is generated by adding a vector of normally distributed random numbers.

If the fitness contribution of the mutated particle relative to the population is better than for the original position then the particle will move to the new position: Firstly, it will be tested which one of the two positions leads to a better hypervolume indicator of the population. Secondly, if both positions are equally good (which will typically occur for dominated solutions), the point that has a better value in the aggregated linear objective function with equal weights is considered. Note that if one solution is dominated by the other solution it will also be considered better in the latter comparison (because of positive equal weighting). Therefore, eventually all solutions will strive towards the non-dominated front and then their hypervolume contribution will be considered.

The cycle continues with picking a random particle again. The MOCOPS algorithm is displayed in pseudocode in Algorithm 1. Care must be taken to ensure $\mathbf{x}_{new} \in \mathbb{S}$ (e.g. by rejecting infeasible vectors).

Algorithm 1. Multiobjective Optimization by Cooperative Swarms (MOCOPS)

Input initial population P_0
while termination criterion is not reached **do**
  $t \leftarrow t + 1$
  $s \sim u(\{1, 2, \ldots, n-1, n\})$
  $\mathbf{x}_{old} = \mathbf{x}^{(s)}$
  $P \leftarrow P_t \setminus \{\mathbf{x}^{(s)}\}$
  {Try to improve position of particle $\mathbf{x}^{(s)}$}
  $\mathbf{z} \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_{new} = \mathbf{x}_{old} + \sigma \cdot \mathbf{z}$
  if $HV(P \cup \{\mathbf{x}_{new}\}) > HV(P \cup \{\mathbf{x}_{new}\})$ **then**
    $P_t \leftarrow P \cup \{\mathbf{x}_{new}\}$
  else if $HV(P \cup \{\mathbf{x}_{new}\}) < HV(P \cup \{\mathbf{x}_{old}\})$ **then**
    $P_t \leftarrow P \cup \{\mathbf{x}_{old}\}$
  else if $f_1(\mathbf{x}_{new}) + f_2(\mathbf{x}_{new}) < f_1(\mathbf{x}_{old}) + f_2(\mathbf{x}_{old})$ **then**
    $P_t \leftarrow P \cup \{\mathbf{x}_{new}\}$
  else
    $P_t \leftarrow P \cup \{\mathbf{x}_{old}\}$
  end if
end while
Return P_t

One iteration of the bicriteria MOCOPS algorithm can be performed with a time complexity in $\mathcal{O}(\mu \log \mu)$ and its complexity is related to the problem of computing the hypervolume contribution of a point which is discussed in [9]. However, by implementing the algorithm as an online algorithm, that is using incremental update steps, we can compute a single iteration with time complexity in $O(\log \mu)$ (amortized over the number of iterations) [12]. Fast - linear time - hypervolume update schemes are also known for three objective functions [11]. The computational complexity is expected to grow exponentially in the number of objective functions [3], that is why the scheme does probably not lend itself very well for many-objective optimization.

3.2 Adaptive Mutation

A weakness of the algorithm design of Algorithm 1 is that the particles are always perturbed with the same distribution and average step-size. When a particle is far away from the optimum a relative big step size will be beneficial. When the solution is already very close to an optimal position then fine-tuning is required, and thus smaller steps.

To account for this we introduce a simple scheme for the adaptation of the standard deviation. In earlier research it has been found that adapting the mutation rate in a stochastic descend method a rate of $\frac{1}{5}$ is a good heuristic choice [1,2]. This has been called the $1/5^{th}$ *success rule*. In our approach we multiply σ or divide it by a scalar in order to keep the success rate of trial moves approximately $\frac{1}{5}$. Heuristically this scalar has been determined as $\sqrt[4]{1.04}$.

In earlier research it has been argued that self-adaptive step-size control does not work in the context of multiobjective optimization, because the boundary between the region where a point improves (Pareto dominates the original point) and the region where a point does not improve becomes cusp-like. However, in the context of hypervolume maximization the boundary between these two spaces is, in most relevant cases, differentiable. This means that in the limit of an infinitely small step-size always a success rate of 1/2 can be achieved, unless the point is exactly on its optimal position on the Pareto front. This is because the improvement region and non-improvement regions are locally separated by a $m-1$ dimensional hyperplane, and the probability to generate a trial point on either one side of the plane is the same.

3.3 Multi-objective Gradient Based Optimization Algorithm

The MOGO algorithm is a deterministic algorithm where each search point is simultaneously directed by a search point dependent subgradient of the total hypervolume gradient. For example, the search point $\mathbf{x}^{(i)}$ could be directly directed by the gradient of $\boldsymbol{f}$. This will however lead to the convergence to a local optimum. However, it is expected that using the gradient this way will lead to little diversification. Therefore, instead we will be use a search direction that is derived from the gradient of the entire population, which was derived in [7] and discussed in more detail in [6]. This strategy leads to convergence as well as to diversification (see for instance [14]).

In [6] it was shown that following the set-gradient of the hypervolume indicator is equivalent to simultaneously moving the particles of a population in a direction of steepest descent of their hypervolume contributions to the population. This direction we will denote with $\nabla \Delta HV(\mathbf{x}, P)$ for some particle $\mathbf{x} \in P$.

The MOGO algorithm we will propose next will basically follow the lines of the gradient flow algorithm in [14], but some important modification will be made in order to eliminate problems with losing dominated solutions and 'creepyness' – a term used in [14] to describe problems caused by large differences in the length of local gradient components.

We use the same definition of $\nabla \Delta HV(\mathbf{x}, P)$ as in [6] and, for the sake of brevity, restrict our discussion on the bicriteria case.

Let $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^2$ denote the vector valued objective function. The Jacobian Matrix of $\mathbf{f}$ at $\mathbf{x}$ is shown below[1].

$$\mathbf{J}_\mathbf{f}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) \cdots \frac{\partial f_1}{\partial x_d}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) \cdots \frac{\partial f_2}{\partial x_d}(\mathbf{x}) \end{pmatrix} \tag{3}$$

The values in a Jacobian indicate how the objective function values respond on a change of the input. The hypervolume contribution derivative vector for the mapping $\mathbb{R}2 \to \mathbb{R}^0_+$ is defined below:

$$\nabla HV(\mathbf{y}) := \begin{pmatrix} \frac{\partial \Delta HV}{\partial y_1} \\ \frac{\partial \Delta HV}{\partial y_2} \end{pmatrix} \tag{4}$$

Its computation can be done by computing the lengths of the line segment in the attainment surface (or 'staircase') that are adjacent to $\mathbf{f}(\mathbf{x})$. For a derivation, see [6].

Multiplication of the matrix and vector yields a gradient of the ΔHV.

$$\nabla \Delta HV(\mathbf{x}, P) = \left(\mathbf{J}_\mathbf{f}(\mathbf{x})^\top \cdot \nabla HV(\mathbf{f}(\mathbf{x}))\right)^\top. \tag{5}$$

In [14] an algorithm was studied that follows the flow of this gradient. It was found to suffer from a strong discrepancy in the length of the subgradients for different points which let to problems in convergence. In this study we counteract this problem by using instead of $\nabla HV(\mathbf{y})$ the normalized subgradient $\nabla HV(\mathbf{y})/||\nabla HV(\mathbf{y})||$:

$$\nabla_N \Delta HV(\mathbf{x}) := \left(\mathbf{J}_\mathbf{f}(\mathbf{x})^\top \cdot \frac{\nabla HV(\mathbf{f}(\mathbf{x}))^\top}{||\nabla HV(\mathbf{f}(\mathbf{x}))||}\right). \tag{6}$$

This way the gradient direction does not change, but the difference in length of sub-gradients is compensated for. Yet, the length of the total gradients decreases in the proximity of hypervolume maximal solutions, which is desirable and makes an additional step-size adaptation mechanism no longer needed.

Moreover, in order to not 'lose' points that are Pareto dominated within the population – they have zero gradients – we propose to move them in the following direction, which is guaranteed to point into the dominance cone in case of locally non-dominated solutions

$$\nabla\left(-\frac{1}{2}f_1 - \frac{1}{2}f_2\right)(\mathbf{x}) = -\frac{1}{2}\left(\nabla f_1(\mathbf{x}) + \nabla f_2(\mathbf{x})\right). \tag{7}$$

Instead of $\frac{1}{2}$, it is also possible to use a small positive step-size σ. This will be done here and which also stresses the analogy to the step-size used in the swarm based search.

[1] In this paper we restrict ourselves to bicriteria optimization but the introduced principles are applicable also in higher dimensions.

The MOGO algorithm follows directly from these preliminaries. It starts with initializing a randomly distributed population. For every search point the descent vector is computed and added to the current solution. The algorithm terminates if stagnation sets in because all descent directions are zero vectors. The MOGO algorithm is displayed in Algorithm 2.

Algorithm 2. Multiobjective Gradient-based Optimization (MOGO)

```
Initialize randomly distributed population
P_0 = (x^(1), ..., x^(μ))
repeat
  c ← 0
  for i ∈ {1, ..., μ} do
    if ||∇ΔHV(x^(i))|| ≠ 0 then
      q ← x^(i) + σ · ∇_N ΔHV(x^(i))
    else if ∇(−f_1 − f_2)(x) ≠ 0 then
      q ← x^(i) + σ∇(−f_1 − f_2)(x)
    else
      q ← x^(i)
    end if
    x^(i) ← q
    c ← c + 1
  end for
until c = μ
```

The computational time complexity of the MOGO algorithm is determined by the time complexity for computing the full hypervolume gradient, the components of which are used as descent directions. It was shown in [6] to be in $\Theta(d\mu + \mu \log \mu)$ (provided the number of objective functions is lower than 4).

4 Evaluation

4.1 Test Problems

Problem 1. The objective functions for problem 1 are depicted below. The reference point used for the hypervolume indicator will be the maximal point $(1.25, 1.25)$ (Fig. 2).

$$f_1(\mathbf{x}) = (x_1)^2 + (x_2 - 0.5)^2 \tag{8}$$

$$f_2(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 0.5)^2 \tag{9}$$

Problem 2. The objective functions for problem 2 are depicted below. The reference point used for the hypervolume indicator will be the maximal point $(1, 1)$ (Fig. 3).

$$f_1(\mathbf{x}) = 1 - ((x_1)^2 + 1)(x_2)^2 \tag{10}$$

$$f_2(\mathbf{x}) = 1 - ((x_2)^2 + 1)(x_1)^2 \tag{11}$$

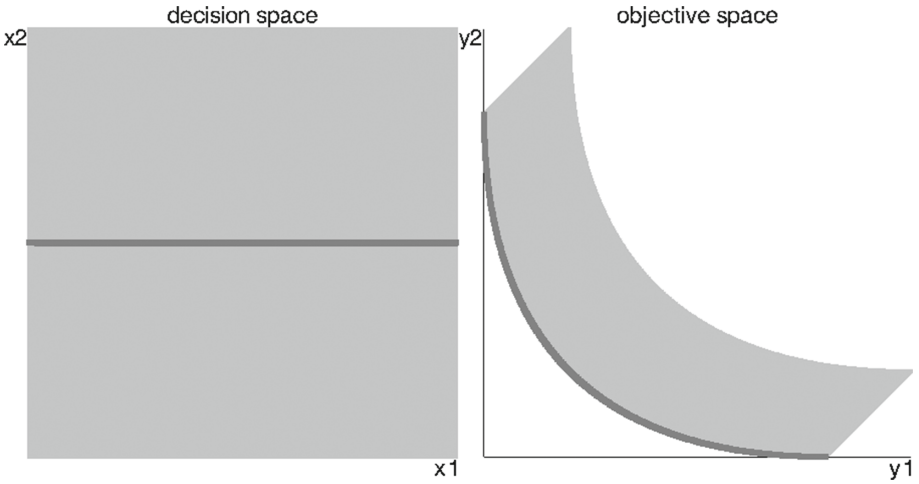


Fig. 2. The decision space and objective space of test problem 1 are shown using a uniformly distributed population of 250000 particles. The area covered with dark gray particles denote the efficient and Pareto set among the population. The area covered with light gray particles resembles the subset which is dominated by the Pareto set of the population. The efficient set is horizontal.

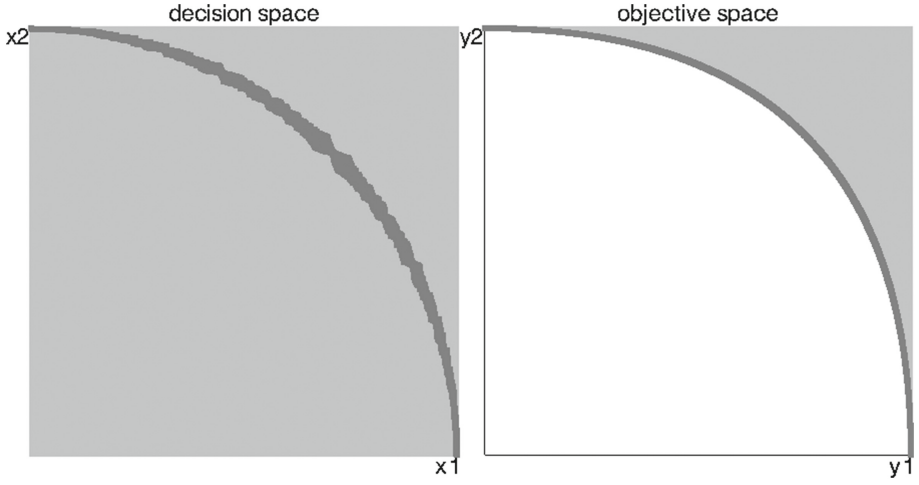


Fig. 3. The decision space and objective space of test problem 2 are shown using a uniformly distributed population of 250000 particles. The area covered with dark gray particles denote the efficient and Pareto set among the population. The area covered with light gray particles resembles the subset which is dominated by the Pareto set of the population. The efficient set is curved.

4.2 Experiments Setup

In this paper the number of reported experiments is kept relatively small. Instead of focussing on statistics we invite the reader to experiment with the implementation which is made available on the website http://moda.liacs.nl/pareto-demo.

With $\mu = 100$ we are going to compare the different algorithms based on the hypervolume indicator with varying amount of iterations. The amount of iterations that will be used is 10, 100 and 1000. Each of these results will be sampled from 100 tests. The 10, 100 and 1000 iterations tests will be performed on both test problems. The MOCOPS will be benchmarked with and without adaptive mutation. The MOCOPS algorithms will have their mutation rate initialized with 0.2. The step size of the MOGO is initialized with 0.0008.

Also we will have a look at the dynamics of the algorithms in a visual way. We will compare the converged populations of the different algorithms.

4.3 Description of Results

The results of the benchmark on problem 1 and 2 can respectively be found in Tables 2 and 3. MOCOPS is the MOCOPS algorithm without adaptive mutations and MOCOPS A is the MOCOPS algorithm with adaptive mutation.

In Figs. 4 and 5 the converged populations of respectively the MOCOPS and MOGO algorithm are shown of test problem 2. It seems the algorithms converge with a slightly different alignment. Both alignments are diverse and approximate the real Pareto set well.

4.4 Discussion of Results

Looking at the benchmarks, it seems the performance of the algorithms is more or less the same. The MOGO seems to perform slightly better than the MOCOPS

Table 2. Test problem 1

Algorithm	10	100	1000
MOCOPS	1.3602	1.3713	1.3879
MOCOPS adaptive	1.3609	1.3696	1.385
MOGO	1.3632	1.3812	1.3909

Table 3. Test problem 2

Algorithm	10	100	1000
MOCOPS	1.3604	1.3709	1.3879
MOCOPS adaptive	1.3606	1.37	1.3851
MOGO	1.3639	1.3807	1.3909

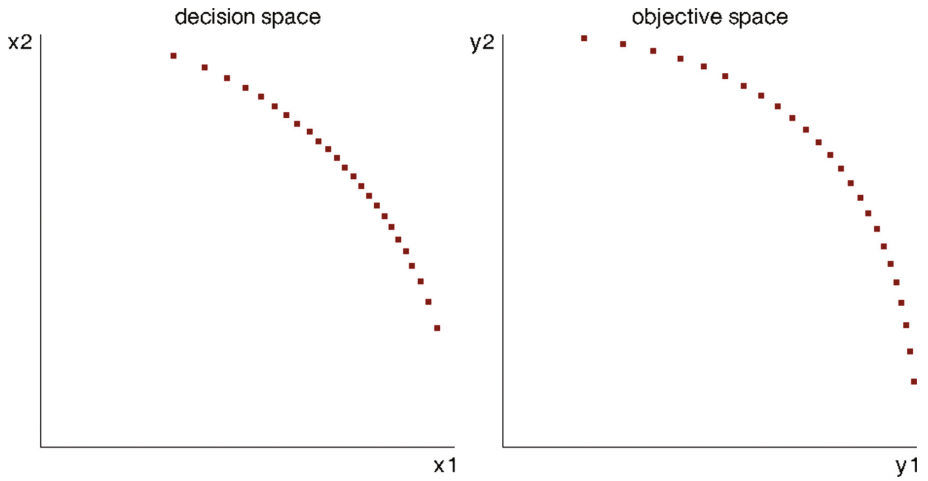


Fig. 4. A screenshot of the converged population with the MOCOPS algorithm on test problem 2.

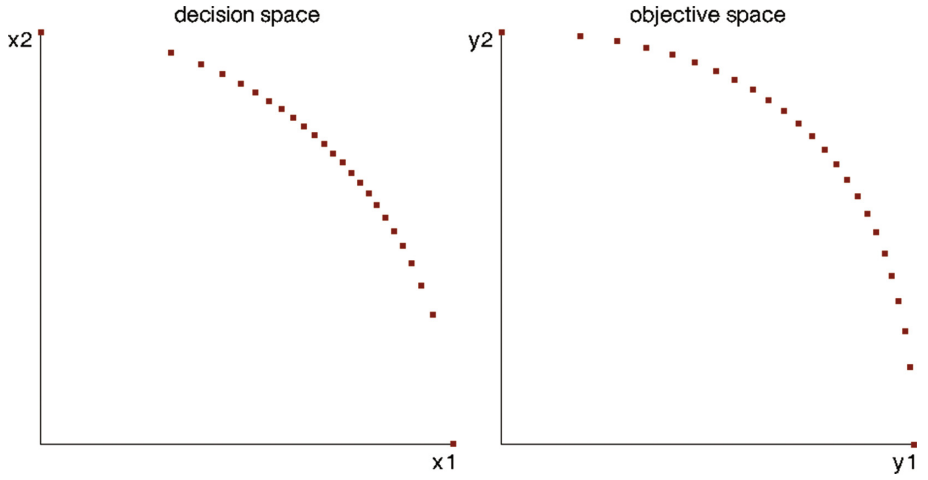


Fig. 5. A screenshot of the converged population with the MOGO algorithm on test problem 2.

variants. The visualization seems to offer more convenient information. Experimenting with the algorithms show that nearly the whole population quickly finds a spot on the Pareto front with the MOGO algorithm. In the MOCOPS algorithm some particles tend to lag behind. Since the MOGO algorithm updates the whole population simultaneously, it is much faster with larger populations.

5 Conclusion

It turns out both the algorithms perform well on the test problems in the sense that they are capable to deliver precise approximations of the Pareto front.

However, the MOGO algorithm outperforms the MOCOPS algorithm. This especially holds with a big population. The MOGO tends to get a nearly perfect diversification. Notably, using the adaptations (gradient length normalization, using the direction as in (7) for dominated points) that were introduced in this paper the MOGO algorithms does not exhibit the problems that were observed for earlier hypervolume-gradient descent schemes. At the same time, it needs to be remarked here that the MOCOPS approach is more robust and does not require differentiability. It is also more promising, when multimodal problems should be considered, though further adaptation would certainly be required for a competitive performance in the multimodal case as compared to other state-of-the-art strategies developed for this problem domain.

For future research it will be interesting to see how the MOCOPS will perform when the adaptive mutation is performed individually instead of globally. The analysis in this paper was on very simple optimization problems and at most can serve as a proof of concept study. Experiments with more challenging test problems and comparisons to state-of-the-art methods will be required before the strategies proposed in this paper can be recommended for practical usage. Based on the first results, we think both approaches to be interesting approaches for further assessment and development.

A Appendices

A.1 Manual of the Application

The application and code is available online [15]. Using the application is straightforward and does not need any installation or configuration. The application can be started with any modern browser. The application is shown in Fig. 6.

1. This is the sidebar with the interaction parameters.
   (a) Pressing the *Printable* button sets the background color to white. Pressing the *Regular* button will set the color scheme regular again.
   (b) In the problem section you can choose a test problem.
   (c) In the initialization section you can select the population size. You can initialize the population with the set population size by pressing one of the buttons. Pressing the *Initialize randomly* button will position the particles at random in the decision space. Pressing the *Initialize uniformly* button will try to position the particles uniformly in the decision space.
   (d) In the algorithm section you can choose the optimization algorithm by pressing *Particle swarm optimization* or *Gradient based optimization*. Deselecting the *Enable dominated set* will enable the use of the whole population. Pressing *Adaptive mutation* makes the MOCOPS algorithm

use the $1/5^{th}$ success rule. The mutation rate for the MOCOPS algorithm can be adjusted by using the slider. The step size of the MOGO can be adjusted similarly with the other slider.

(e) In the optimization section you can select how many milliseconds delay every iteration should have using the slider. A bigger delay can make the dynamics of the algorithm clearer. The button *Start* will start the selected algorithm. Pressing the button *Stop* will stop the algorithm again. The *Benchmark* button will run some benchmarks and outputs the statistics in

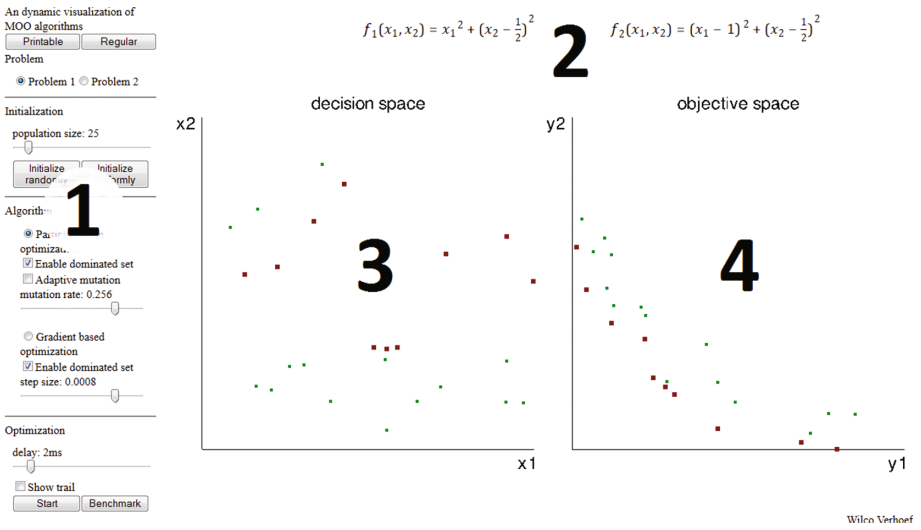


Fig. 6. A screenshot of the interactive multi-objective optimization application.

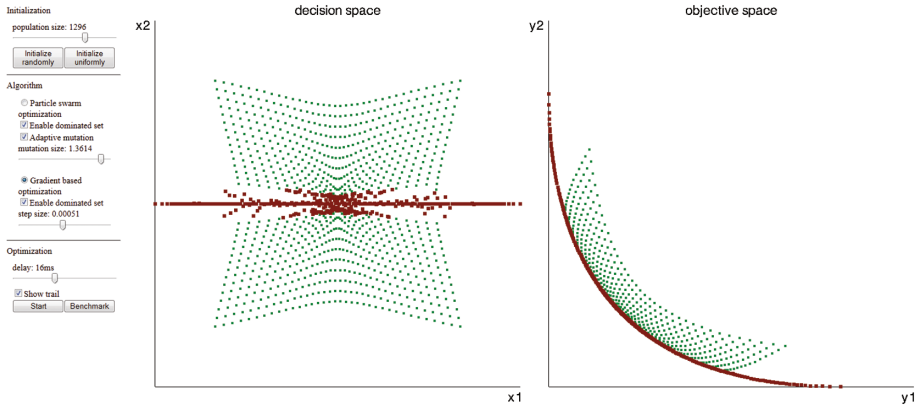


Fig. 7. A screenshot during convergence of the population with the MOGO algorithm on test problem 1.

the browser console. In most browsers the console is accessible by pressing `F12`.

(f) The application automatically adapts to the window size. Full screen mode is available with `F11`.

2. The objective functions of the chosen test problem are shown here.
3. This part of the screen shows the decision space. Points of the efficient set of the population are displayed in red squared dots. The particles which are dominated by the particles in the efficient set are displayed as green dots.
4. This part of the screen shows the objective space. Points of the Pareto set of the population are displayed in red squared dots. The particles which are dominated by the Pareto set are shown as green dots.

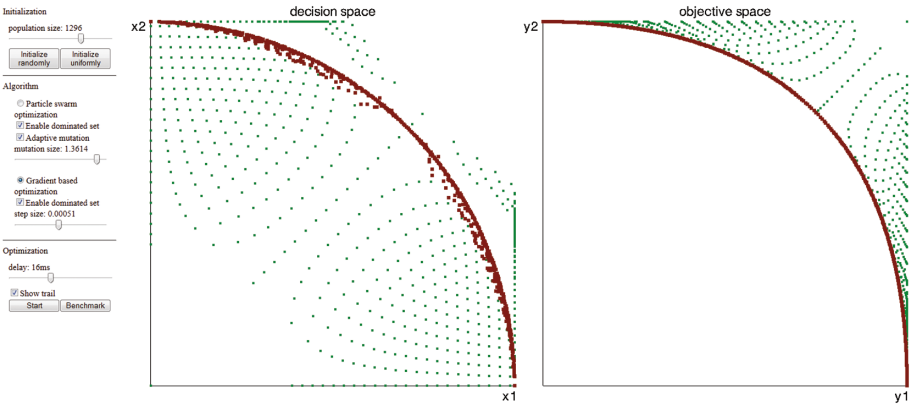


Fig. 8. A screenshot during convergence of the population with the MOGO algorithm on test problem 2.

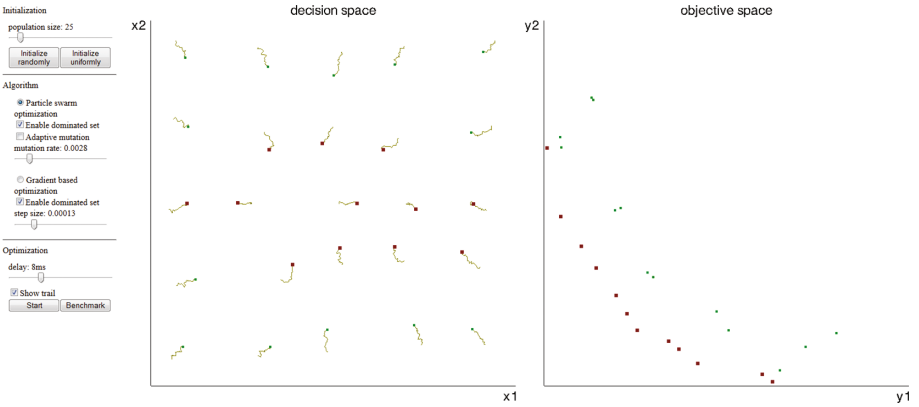


Fig. 9. A screenshot with path tracing during the convergence of a population with the MOCOPS algorithm on test problem 1.

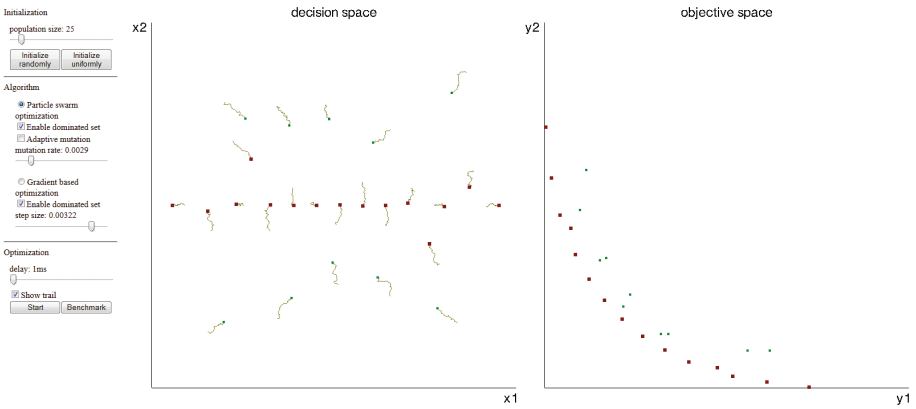


Fig. 10. A screenshot with path tracing during the convergence of a population with the MOCOPS algorithm on test problem 1 at a late stage.

Finally, we exhibit some example screenshots:

- Figure 7 shows a large population that is in the process of converging towards the Pareto front starting from a uniformly distributed sample. The MOGO algorithm is applied on Problem 1. Green, small points are dominated and red squared dots are non-dominated with respect to the other points in the population.
- Figure 8 shows a large population that is in the process of converging towards the Pareto front starting from a uniformly distributed sample. The MOGO algorithm is applied on Problem 2.
- Figure 9 shows a small population of particles moved by the MOCOPS algorithm on Problem 1. Also, the traces of the recent moves of the particles are visualized. Note, that points on the efficient set move sideways in order to find their optimal position with respect to diversity (hypervolume contribution).
- Figure 10 shows the same population as shown in Fig. 9 at a later stage of the convergence process.

References

1. Auger, A.: Benchmarking the (1+1) evolution strategy with one-fifth success rule on the bbob-2009 function testbed. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, GECCO 2009, pp. 2447–2452. ACM, New York (2009)
2. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies–a comprehensive introduction. Nat. Comput. **1**(1), 3–52 (2002)
3. Bringmann, K., Friedrich, T.: Approximating the least hypervolume contributor: Np-hard in general, but fast in practice. In: Evolutionary Multi-Criterion Optimization, pp. 6–20. Springer (2009)

4. Coello Coello, C.A., Lechuga, M.S.: Mopso: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, vol. 2, pp. 1051–1056. IEEE (2002)
5. Emmerich, M., Beume, N., Naujoks, B.: An emo algorithm using the hypervolume measure as selection criterion. In: Evolutionary Multi-Criterion Optimization, pp. 62–76. Springer (2005)
6. Emmerich, M., Deutz, A.: Time complexity and zeros of the hypervolume indicator gradient field. In: Schuetze, O., Coello Coello, C.A., Tantar, A.-A., Tantar, E., Bouvry, P., Del Moral, P., Legrand, P. (eds.) EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation III. Studies in Computational Intelligence, vol. 500, pp. 169–193. Springer International Publishing (2014)
7. Emmerich, M., Deutz, A., Beume, N.: Gradient-based/evolutionary relay hybrid for computing Pareto front approximations maximizing the S-metric. Springer (2007)
8. Emmerich, M.T.M., Deutz, A.H., Yevseyeva, I.: On reference point free weighted hypervolume indicators based on desirability functions and their probabilistic interpretation. Procedia Technol. **16**, 532–541 (2014)
9. Emmerich, M.T.M., Fonseca, C.M.: Computing hypervolume contributions in low dimensions: asymptotically optimal algorithm and complexity results. In: Evolutionary Multi-Criterion Optimization, pp. 121–135. Springer (2011)
10. Fleischer, M.: The measure of pareto optima applications to multi-objective metaheuristics. In: Evolutionary Multi-Criterion Optimization, pp. 519–533. Springer (2003)
11. Guerreiro, A.P., Fonseca, C.M., Emmerich, M.T.M.: A fast dimension-sweep algorithm for the hypervolume indicator in four dimensions. In: CCCG, pp. 77–82 (2012)
12. Hupkens, I., Emmerich, M.: Logarithmic-time updates in sms-emoa and hypervolume-based archiving. In: EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV, pp. 155–169. Springer (2013)
13. Mostaghim, S., Branke, J., Schmeck, H.: Multi-objective particle swarm optimization on computer grids. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007, pp. 869–875. ACM, New York (2007)
14. Hernández, V.A.S., Schütze, O., Emmerich, M.: Hypervolume maximization via set based Newton's method. In: Tantar, A.-A., Tantar, E., Sun, J.-Q., Zhang, W., Ding, Q., Schtze, O., Emmerich, M., Legrand, P., Del Moral, P., Coello Coello, C.A. (eds.) EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V. Advances in Intelligent Systems and Computing, vol. 288, pp. 15–28. Springer International Publishing (2014)
15. Verhoef, W.: Interactive demo on multi-objective optimization, liacs, leiden university, nl, wilco.verhoef.nu/projects/moo, bsc project (2015)

Quadcriteria Optimization of Binary Classifiers: Error Rates, Coverage, and Complexity

Vitor Basto-Fernandes[1,2], Iryna Yevseyeva[3], David Ruano-Ordás[4], Jiaqi Zhao[5], Florentino Fdez-Riverola[4], José Ramón Méndez[4], and Michael T.M. Emmerich[6(✉)]

[1] Instituto Universitario de Lisboa (ISCTE-IUL), University Institute of Lisbon, ISTAR-IUL, Av. Das Forcas Armadas, 1649-026 Lisboa, Portugal
vitor.basto.fernandes@iscte.pt

[2] School of Technology and Management, Computer Science and Communications Research Centre, Polytechnic Institute of Leiria, 2411-901 Leiria, Portugal
vitor.fernandes@ipleiria.pt

[3] School of Computer Science and Informatics, Faculty of Technology, Cyber Technology Institute, De Montfort University, Gateway House, The Gateway, Leicester LE1 9BH, UK
iryna@dmu.ac.uk

[4] Informatics Engineering School, University of Vigo, Campus As Lagoas S/N, 32004 Ourense, Spain
{drordas,riverola,moncho.mendez}@uvigo.es

[5] The School of Computer Science and Technology, China University of Mining and Technology, No 1, Daxue Road, Xuzhou 221116, Jiangsu, P.R. China
jiaqizhao88@126.com

[6] Multicriteria Optimization, Design, and Analytics Group, LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
emmerich@liacs.nl

Abstract. This paper presents a 4-objective evolutionary multiobjective optimization study for optimizing the error rates (false positives, false negatives), reliability, and complexity of binary classifiers. The example taken is the email anti-spam filtering problem.

The two major goals of the optimization is to minimize the error rates that is the false negative rate and the false positive rate. Our approach discusses three-way classification, that is the binary classifier can also not classify an instance in cases where there is not enough evidence to assign the instance to one of the two classes. In this case the instance is marked as suspicious but still presented to the user. The number of unclassified (suspicious) instances should be minimized, as long as this does not lead to errors. This will be termed the *coverage objective*. The set (ensemble) of rules needed for the anti-spam filter to operate in optimal conditions is addressed as a fourth objective. All objectives stated above are in general conflicting with each other and that is why we address the problem as a 4-objective (quadcriteria) optimization problem. We assess the

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_3

performance of a set of state-of-the-art evolutionary multiobjective optimization algorithms. These are NSGA-II, SPEA2, and the hypervolume indicator-based SMS-EMOA. Focusing on the anti-spam filter optimization, statistical comparisons on algorithm performance are provided on several benchmarks and a range of performance indicators. Moreover, the resulting 4-D Pareto hyper-surface is discussed in the context of binary classifier optimization.

Keywords: Binary classification · Three-way classification · Parsimony · Evolutionary multi-objective optimization · Parallel coordinates

1 Introduction

An email anti-spam system consists of a set of boolean filtering rules, that operate jointly and support spam messages detection. Discovering the relative importance of these rules and assigning the corresponding scores (weights) of each rule is a complex setup and maintenance process.

The need of frequent scores reassignment for existing rules and score settings for new rules, to keep the anti-spam filter updated and running, requires the adoption of machine learning and optimization techniques. In rule-based spam filtering, several binary classification techniques are combined in a filter, allowing for flexible creation and deployment of highly customized spam filtering. These techniques include intelligent analysis of email content, collaborative querying and information sharing on senders, deliveries and legitimacy verification. Each technique individually is not able to provide efficient classification, but their joint usage provide acceptable levels of classification quality.

Each rule corresponds to a logical test and has a score assigned for the filter operation. An email message is checked (binary classified) by each rule, and the final classification is done by comparing the sum of all the matching rule scores with a required threshold value. When the sum is above an upper threshold the message is classified as spam, when the sum is below a lower threshold the message is classified as legitimate. Otherwise it is not classified and marked as a boundary case.

Continuous creation of new ways to distribute spam, leads to the need of continuous creation of new anti-spam filtering rules and corresponding scores setting, in addition of scores updating of existing rules. Rules creation and scores setting is mainly performed manually by system administrators based on experience, applying a try-and-error approach. It is a complex process that has to take into account the existing rules knowledge base and the relative importance of rules to assign individual scores.

This process involves the analysis of thousands of rules and scores to create a complex highly customized anti-spam filter. The anti-spam filter is highly dependent on the type of organization, business or leisure domains, location, language, culture and other user specific criteria. The importance of different objectives can change depending on the context. For instance, in leisure domains false positive classifications may have less serious consequences than they would have in business domains.

A survey of literature on assisted/automatic configuration proposals is found in the literature for the SpamAssassin anti-spam filtering in [3].

The general framework of the evolutionary multiobjective optimization approach for automatic anti-spam filtering scores tuning presented here was presented by some authors of this paper in recent works [4–6]. So far the framework was tested with at most three objective functions at a time. Now, the approach is extended to a four-objective problem formulation, including three-way classification (instead of binary classification) and parsimony (or complexity reduction of the rule base system), in addition to false positive and false negative minimization objectives.

This work is structured as follows. Section 2 presents the anti-spam filter multiobjective optimization problem formulation and its relation to the machine learning perspective. In Sect. 3, the 4-objective multiobjective optimization problem formulation is presented. Section 4 details the experiments design and protocol. Then Sect. 5 presents the results analysis and discussion. Finally, in Sect. 6 the authors present the conclusions and future research directions.

2 Multiobjective Problem Formulation

Machine learning problems often can be formulated as multiobjective optimization problems. Maximizing classification performance metrics such as true positive rates and true negative rates, parsimony at the same time, trade-off new information and forgetting outdated one and learning details while performing model abstracting, are examples of typical machine learning trade-offs to be achieved [7].

In multiobjective optimization problems m objective functions $f = (f_1, f_2, \ldots, f_m)$ must be optimized simultaneously, such that f_k, $k \in \{1, ..., m\}$ are real-valued functions evaluated for points in some decision/input space (e.g., the weights of the rules and values of thresholds to be found). Each point in the decision space maps to an m dimensional vector in the so-called objective space, containing the objective function values (e.g., false positive rate, false negative rate, complexity and coverage of a classifier).

Solutions in the objective space are only partially ordered, that is two solutions can either be in a dominance relation or they can be incomparable to each other. A solution Pareto dominates another solution if it is better (lower for minimization) or equal on all, and better on at least one objective. If for two solutions neither the first dominates the second nor the second dominates the first they are said to be incomparable. The solutions in the decision space that are not dominated by other solutions are called *efficient solutions* and together form the efficient set. The projections of these solutions into the objective space are called Pareto optimal points and together form the *Pareto front*. In general, the Pareto front of a problem with m objective functions is at most of dimension $m-1$. This means that for the 4-objective problems considered in this work, the Pareto fronts will be at most three dimensional.

Multiobjective optimization following the *a posteriori approach* results in the *efficient set* and *Pareto front* – or an approximation to these sets. Selection of a

single solution among Pareto optimal ones is done by or on behalf of the decision maker according to his/her preferences [15]. The knowledge of the Pareto front also reveals insights into the structure of the essential conflict between different objective functions. This information can be useful for the classification system's designer in order to assess problem inherent trade-off or to see limitations on what can possibly be achieved with a certain classification software by tuning parameters.

In the following we seek to find approximations to Pareto fronts that arise in the context of binary classifier tuning, and, more specifically, in tuning spam classifiers. The following objective functions will be considered:

- *Minimize false negative rates:* A spam detection system's major purpose is to detect all spam messages. If a spam email is not detected this error is called a false negative.
- *Minimize false positive rates:* A spam detection system should not mistakenly classify legitimate emails, also called *ham*, as being spam emails. If an email is classified as spam but it is not a spam email, we call this error a false positive.
- *Minimize the number of unclassified samples:* Ideally a spam filter should classify all instances. However, as discussed in [8,14], binary classification of instances as positive or negative is sometimes too strict and can result in high misclassification costs. Three-way classification can also leave an email unclassified in case of low confidence in classification. In this case only a warning or suspicious flag is provided and it is assumed that the user will then correctly classify the email. This way, especially in badly supported cases significant improvements on error rates can be achieved, albeit at the cost of additional work for the user. Advantages of the so called three-way classification approach are described in the literature as the ability to provide a more complete feedback to the users, and in this sense reducing qualitatively and/or quantitatively the misclassification rate.
  Difficult instances to be classified (boundary cases) are marked as *unclassified* and forwarded to the user for further examination. This way the maximization of coverage is formulated as the minimization of the number of unclassified samples.
- *Minimize complexity:* In previous work on anti-spam filter optimization [8], it was observed that many rules were not participating in the classification process and those with very small scores only marginally influenced the classification results. This observation suggests that in addition to minimizing the occurrence of false positives and false negatives, the complexity of the anti-spam filter (or its parsimony) can also be optimized. For the anti-spam filtering case, we measure parsimony as the minimum number of rules with score different from zero, that support a specific classification quality.

Several conflicts between these four objectives can be identified: Firstly, there is a conflict between the false positive rate and the false negative rate. In extreme cases of anti-spam systems tuning/configuration the system can always have a zero false positive rate, that is a zero rate of legitimate messages lost.

This rate is obtained for instance if no instance is marked as spam. Usually this comes at the expense of higher false negative classifications. In the other extreme, system tuning may classify a large number of instances as being spam. In the most extreme case it might even classify all emails as spam. This will minimize the false negative rate but comes at the expense of more legitimate emails being dismissed (high false positive rate).

Secondly, there is typically a conflict between a high coverage of the classifier and the error rates. Obviously, every instance that is not examined by the user but instead automatically classified can potentially lead to misclassification costs (increment of false positive and false negative rates). On the other hand the effort of the user and his/her exposure to suspicious emails should be minimized.

Finally, there is a conflict between the error rates and the complexity of a classifier. It goes without saying that extremely simple classifiers might have a low accuracy while more complex classifiers can capture more complex rules and therefore potentially yield classifiers with lower error rates. Here, it should be noted that from a certain level onwards adding complexity to the classifiers might not anymore yield to improvements in terms of error rates and even can be counterproductive due to overfitting. To identify this critical complexity level which will be obtained as an upper bound of the Pareto front, can be a valuable output of multiobjective optimization.

The conflict between coverage and complexity seems to be of a somewhat more complex nature and we will assess it by means of empirical results. In our multiobjective problem formulation, 4-objectives are considered to be minimized, false positive rate, false negative rate, unknown classifications rate and parsimony.

3 Quadcriteria Optimization Methods

The optimization or tuning of anti-spam filtering systems using rule ensembles as classifiers is difficult to be accomplished with derivative-based deterministic optimization techniques, because the behaviour of rules can be highly complex, non-smooth and non-linear. This is why instead metaheuristics are used for this task. Among the metaheuristics used for multiobjective optimization, evolutionary multiobjective optimization algorithms (EMOAs) are allegedly the most frequently and best studied approaches.

In our study, state-of-the-art EMOAs were selected for the anti-spam filtering optimization problem, namely, NSGA-II [12], SPEA2 [16] and SMS-EMOA [17]. These methods are representative for the recently most relevant generational Pareto-based, steady state Pareto-based and indicator-based EMOA approaches. Testing representative methods from different optimization strategies groups, allow the study of their behaviour and performance for the anti-spam filtering problem.

In the current study the following formulation for the anti-spam filtering optimization is adopted. A four-objective (*fnr* - false negative rate, *fpr* - false positive rate, *ur* - unclassified rate, *cr* - complexity rate) binary-real representation decisions variable formulation (a real valued scores vector and a binary string/vector

for rules activation/deactivation), aiming at classifier performance and complexity rate improvement is proposed. Minimizing these objectives, means reducing the number of spam messages not identified by anti-spam filtering techniques, reducing the number of legitimate messages classified as spam by mistake, reducing the number of unclassified messages, and reducing the number of rules used in the classifier.

The anti-spam filtering problem is formulated here as a multiobjective optimization problem on a mixed-integer decision space and with normalized objective function values in the range $[0;1]$ that are all to be minimized. These are false negative rate (f_1), false positive rate (f_2), number of unclassified instances divided by the number of instances (f_3) and number of rules used by the classifier divided by the total number of available rules (f_4).

Minimization is assumed for all objectives, evaluated in decision space with a vector of decision variables, $w = (w_1, w_2, \ldots, w_n)$, with n being the total number of rules. The output for all n rules of a filter is weighted by these in order to compute a final score. It is also possible to assign a negative score to a rule. The individuals of initial population are generated randomly with scores in the $[-5;5]$ range. New individuals are also generated by variation operations in the same range.

While a real decision variables vector is used for the representation of the anti-spam rules scores settings in the interval $[-5;5]$, a binary vector $b = (b_1, \ldots, b_n) \in \{0,1\}^n$ of decision variable, with each bit representing one rule for the algorithm to activate (or deactivated) rules according to variation and selection operators along the evolutionary process, allows the algorithm to activate/deactivate rules and assess their relevance in the classification process. In addition a lower and an upper threshold t_1 and t_2 are optimized by the algorithm. Both thresholds are set in the interval $[0;1]$, with lower threshold being always lower than the upper threshold. If the sum of rules' score that match an email message is below t_1 it is classified as ham, if it is above t_2 it is classified as spam, and otherwise it remains unclassified. Therefore, the values of t_1 and t_2 will have a direct impact on the number of unclassified samples but also on the misclassification costs.

A normalized counting of false negatives, false positives, unknown messages and number of active rules is adopted, leading all four objectives to assume values in the range of [0;1], as described in Eqs. 1, 2, 3 and 4.

$$fnr(w,b,t_1,t_2) = fn(w,b,t_1,t_2)/TotalNumberOfSpamMessages \rightarrow \min \quad (1)$$

$$fpr(w,b,t_1,t_2) = fp(w,b,t_1,t_2)/TotalNumberOfHamMessages \rightarrow \min \quad (2)$$

$$ur(w,b,t_1,t_2) = \#unclassified(w,b,t_1,t_2)/TotalNumberOfMessages \rightarrow \min \quad (3)$$

$$cr(w,b,t_1,t_2) = \sum_{i=1}^{n} b_i/TotalNumberOfRules \rightarrow \min \quad (4)$$

with $w \in [-5,5]^n$ being the rules weights, $b \in \{0,1\}^n$ being the rule activation variables ($n = TotalNumberOfRules$), and (t_1, t_2) being the lower and upper

threshold. Finally, in order to ensure that the thresholds are feasible we introduce the constraints:

$$0 \leq t_1 \leq t_2 \leq 1 \tag{5}$$

4 Experimental Setup

In the next section, we describe the experimental setup and the performance evaluation metrics used for the experiments results analysis. SpamAssassin is the anti-spam filtering system adopted in our experiments due to wide adoption by the open source community, the research community on anti-spam systems, its wide commercial usage, and available email corpora. The SpamAssassin corpus used in our experiments is composed of 9349 email messages, 2398 spam and 6951 legitimate messages [9].

The experiments phase was performed following the default spam filter configuration present in the Debian GNU/Linux Squeeze distribution running SpamAssassin 3.3.1 [10]. Filtering rules scores range fall under the interval $[-5; 5]$.

From the 2440 rules available in SpamAssassin distribution, only those fitting at least one message in the dataset are considered in the optimization experiments. Actually, only 330 rules fit email messages and only those have been considered in the optimization process.

Experiments were performed with jMetal [11] version 4.5, an optimization framework for the development of multiobjective metaheuristics in Java. A jMetal RealBinary encoding decision variables scheme was used where the chromosome is constituted by an array of real values in the interval $[-5;5]$ and a bit string. The length of the chromosome is determined by the number of anti-spam filtering rules effectively used (330). Each rule is associated with a real value score in the $[-5;5]$ interval and a one bit in the chromosome. If the i^{th} bit is 0 the i^{th} rule is ignored, and otherwise the rule is active and considered by the spam classifier with the i^{th} corresponding real value score. Messages are classified as spam when the sum of scores of active rules that match the message is equal or greater than the defined threshold value.

NSGA-II, SPEA2 and SMS-EMOA algorithms stopping criteria are set to a maximum of 25000 function evaluations. The SBX single point crossover and polynomial bit flip mutation operators are used as the variation operators in the experiments, with crossover probability 0.9 and mutation probability $1/n$, where n is the number of anti-spam filtering rules. Population size is 100 individuals for all algorithms, archive size 100 for SPEA2 and offset set to 100 for SMS-EMOA. In order to obtain robust performance statistics, all stochastic algorithms perform 30 independent runs.

Although performance assessment of multiobjective optimization algorithms constitutes a complex task, involving outcome quality assessment, computing resources usage, analysis of several runs of the stochastic based algorithms, in addition of not having an absolute optimal for comparison purposes, a set of well established performance indicators can be used and provide guidance for multiobjective optimization algorithms performance assessment. General performance

criteria include accuracy (or convergence) which measures how close the solutions are to some known optimal, coverage (or uniformity) which measures how many non-dominated solutions are generated and how well they are distributed, and variance (or spread) which measures, for every objective, the maximum range of non-dominated front covered by the generated solutions.

In order to evaluate and compare approximation sets from multiple runs of two or more stochastic multiobjective optimizers, complementary techniques must be combined.

Firstly, we use the 4-D hypervolume (HV) indicator as a measure for the size of the subspace that is dominated by the Pareto front approximation. This dominance compliant quality indicator has favorable theoretical properties (Pareto compliants) and high values indicate a diverse set of solutions located close to the true Pareto front. For the HV indicator a reference point is required. This choice is problem specific and it is important that all points in the Pareto front approximation are always dominated by the reference point.

The hypervolume indicator is accompanied by the SPREAD indicator. This indicator shows how far Pareto front, spreads in objective space, or decision space. The larger the spread of the Pareto front is, the wider range of values on objectives it covers. This indicator is not Pareto compliant but is useful to understand the geometry of the obtained Pareto fronts, in particular whether also extreme parts are well covered.

A graphical representations of the 4D Pareto front is done by means of graphical parallel coordinates. The objective functions determine the four parallel axis of this diagram. Each solution in the Pareto front representation is represented as a polyline. The diagram we used was made available by the statistical software `System R`. The Pareto front that we visualized is the reference Pareto front, that is the Pareto front obtained from the non-dominated subset of the union of all Pareto front approximations obtained in the independent runs. The main purpose of this graphic is to provide insights into the trade-offs of the anti-spam filtering problem and to obtain a good compromise solution.

5 Results Analysis

The results presented in this section outline NSGA-II, SPEA2 and SMS-EMOA algorithms behavior and performance on the 4-objectives anti-spam filtering problem formulation.

Figure 1 shows the reference Pareto front, in other words, the best individuals of all algorithms of all runs. It is clear from this figure that the anti-spam filtering system may have very low fnr, fpr and ur, while using less than 38.5% of the 330 rules that match some message in the email corpus. Some solutions could reach fnr, fpr, and ur, all falling under 1%, using around 16% of the anti-spam filtering rules. The finding is confirmed by the plot in the scatterplot matrix in Fig. 2, showing clearly the range in which there is a conflict between fnr and *complexity*.

The experiments results confirm the trend of increasing the number of rules in the anti-spam filtering system (due to the need of dealing with new spam

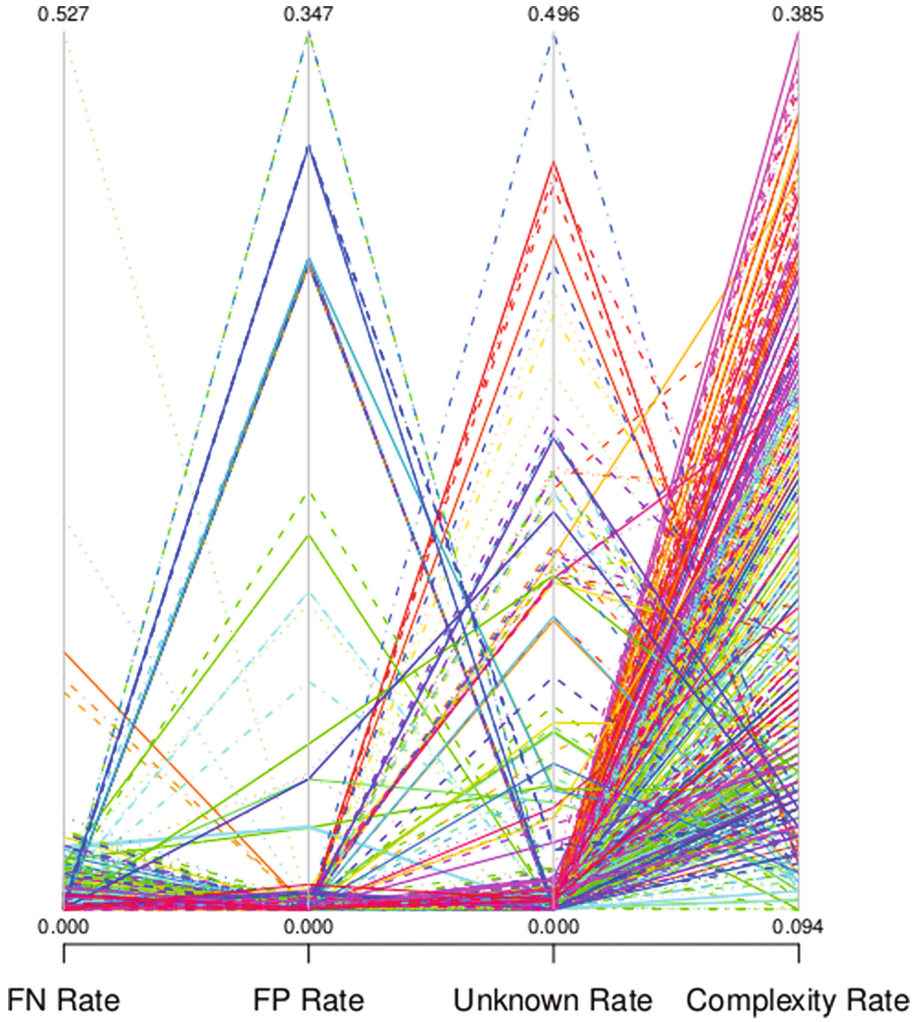


Fig. 1. Reference front parallel coordinates plotting for four objectives anti-spam problem formulation.

message), has a marginal effect on progressively decreasing the relevance of the rules set in the classifier quality, while having a considerable impact on classifier complexity, and therefore on the computational effort. This suggests that in order to introduce new rules, multiobjective optimization should be used to avoid redundancy or near-redundant rules in a rule ensemble.

From the experiments it is also possible to conclude that the 4-objectives classifier is able to reach high levels of accuracy with respect to the false negative rate, false positive rate and unclassified rate, even when only a small fraction of filtering rules are activated.

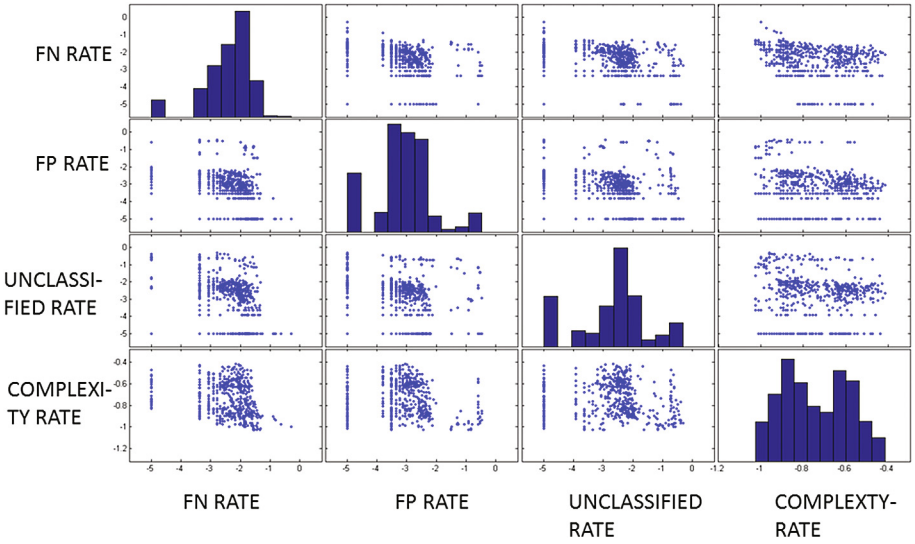


Fig. 2. Scatterplot matrix of combined set of results. Of all values the logarithm with basis 10 is taken.

Besides, our results provide first trends on the question which algorithms are most suitable for the quadcriteria optimization of binary classifiers. Boxplots depicting statistics on the multicriteria performance indicators hypervolume and spread, are shown for the algorithms under consideration, namely NSGA-II, SPEA2 and SMS-EMOA (Figs. 3 and 4). The boxplots represent graphically the

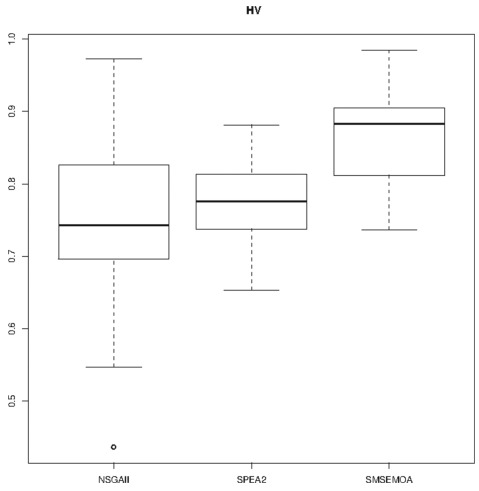


Fig. 3. Hypervolume boxplot for four objectives anti-spam problem formulation.

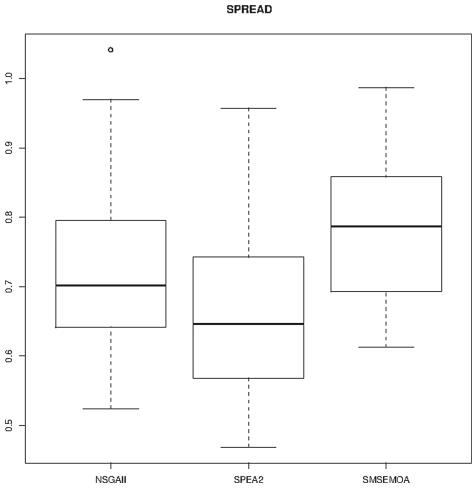


Fig. 4. SPREAD boxplot for four objectives anti-spam problem formulation.

median, quartiles, and outliers of the statistics on each algorithm (30 runs, each). The comparison of NSGA-II, SPEA2 and SMS-EMOA is done with respect to the reference Pareto front.

From the boxplots we conclude that SMS-EMOA is clearly the best performing algorithm of all with respect to both hypervolume and spread indicators. The second best performing algorithm with respect to hypervolume indicator is SPEA2, and the second best algorithm with respect to spread indicator is NSGA-II. It should be noted, however, that SMS-EMOA is also the most demanding algorithm in terms of required CPU-time. For the same computation resources used in our experiments to run all algorithms, a SPEA2 run duration could be measured in a scale of seconds, NSGA-II in a scale of minutes and SMS-EMOA in a scale of dozens of minutes. The hypervolume calculation of jMetal SMS-EMOA implementation used in our experiments, follows traditional algorithms approaches, which are relatively slow for 4-D hypervolume [18]. Recent faster implementations are now available [19,20] and will be adopted in the future for 4-D hypervolume calculations.

6 Conclusions and Future Work

Introducing three-way classification and parsimony as additional and simultaneous optimization objectives, in addition to false negatives and false positives minimization, revealed important improvements that can be achieved in modern anti-spam filtering systems.

For the proposed 4-objectives problem formulation, it was found that a good performance can be achieved with a small number of rules being used by the anti-spam filtering classifier. It was observed that from 330 rules that match

messages in the SpamAssassin data corpus, only 16% to 38.5% of rules are needed to achieve classification error rates on all the other three objectives (fnr, fpr, ur) under 1%.

The authors will take this results as the basis for further anti-spam filtering and classifier optimization research, heading in two main directions. First, other multiobjective and many-objectives optimization algorithms of high potential for the anti-spam filtering type of problems will be studied and explored (e.g. MOEA/D and NSGA-III), and also tailor made approaches for classification such as CH-EMOA [1,8] or mixed integer optimization [2]. Secondly, analysis of the rules that reveal highest contributions for the classification process will be addressed, in order to assess not only quantitative classifier complexity, but also to explore the nature of the rules most frequently present in the best solutions. Knowledge exploration raised by this analysis was introduced by the authors in [13]. There the authors point research hypothesis and directions with respect to rules relative relevance analysis, and anti-spam rules automatic generation guided by knowledge extracted by the means of multiobjective optimization techniques.

Finally, it will be of interest to consider newly proposed visualization techniques for 4-D Pareto fronts byTuša and Filipič [21], in order to gain additional insight into the structure of the Pareto front data.

Acknowledgements. This work was partially funded by the [14VI05] Contract-Programme from the University of Vigo. Iryna Yevseyeva acknowledges Engineering and Physical Sciences Research Council (EPSRC), UK, and Government Communications Headquarters (GCHQ), UK, for funding Choice Architecture for Information Security (ChAISe) project EP/K006568/1 as a part of Cyber Research Institute.

References

1. Wang, P., Emmerich, M., Li, R., Tang, K., Bäck, T., Yao, X.: Convex hull-based multi-objective genetic programming for maximizing receiver operating characteristic performance. IEEE Trans. Evol. Comput. **19**(2), 188–200 (2015)
2. Li, R., Emmerich, M.T., Eggermont, J., Bäck, T., Schütz, M., Dijkstra, J., Reiber, J.H.: Mixed integer evolution strategies for parameter optimization. Evolu. Comput. **21**(1), 29–64 (2013)
3. Basto-Fernandes, V., Yevseyeva, I., Méndez, J.R.: Anti-spam multiobjective genetic algorithms optimization analysis. Int. Resour. Manage. J. **26**(1), 54–67 (2012)
4. Yevseyeva, I., Basto-Fernandes, V., Méndez, J.R.: Survey on anti-spam single and multi-objective optimization. In: Cruz-Cunha, M.M., Varajo, J., Powell, P., Martinho, R. (eds.), ENTERprise Information Systems. Communications in Computer and Information Science, vol. 220, pp. 120–129. Springer, Heidelberg (2011)
5. Basto-Fernandes, V., Yevseyeva, I., Méndez, J.R.: Optimization of anti-spam systems with multiobjective evolutionary algorithms. Int. Resour. Manage. J. **26**, 54–67 (2012)
6. Yevseyeva, I., Basto-Fernandes, V., Ruano-Ordás, D., Méndez, J.R.: Optimising anti-spam filters with evolutionary algorithms. Expert Syst. Appl. **40**(10), 4010–4021 (2013)

7. Jin, Y.: Multi-objective Machine Learning. Studies in Computational Intelligence. Springer, Heidelberg (2006)
8. Zhao, J., Basto-Fernandes, V., Jiao, L., Yevseyeva, L., Maulana, A., Li, R., Bäck, T., Emmerich, M.T.M.: Multiobjective optimization of classifiers by means of 3-d convex hull based evolutionary algorithm, ARXIV Computer Science abs/1412.5710 (2014). http://arxiv.org/abs/1412.5710
9. The Apache SpamAssassin Project - SpamAssassin public corpus (2005). http://spamassassin.apache.org/publiccorpus
10. SpamAssassin Team: The apache spamassassin project (2011). http://spamassassin.apache.org/
11. Durillo, J.J., Nebro, A.J.: jMetal: a java framework for multi-objective optimization. Adv. Eng. Softw. **42**, 760–771 (2011)
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
13. Basto-Fernandes, V., Yevseyeva, I., Frantz, R.Z., Grilo, C., Daz, N.P., Emmerich, M.: An automatic generation of textual pattern rules for digital content filters proposal, using grammatical evolution genetic programming. Procedia Technol. **16**, 806–812 (2014)
14. Yao, Y.: The superiority of three-way decisions in probabilistic rough set models. Inf. Sci. **181**(6), 1080–1096 (2011)
15. Miettinen, K.: Nonlinear Multiobjective Optimization. Springer, New York (1999)
16. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength Pareto evolutionary algorithm. In: Proceedings of EUROGEN 2001, Athens Greece. CIMNE, Barcelona (2001)
17. Emmerich, M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005)
18. While, L., Bradstreet, L., Barone, L.: A fast way of calculating exact hypervolumes. IEEE Trans. Evol. Comput. **16**(1), 86–95 (2012)
19. Emmerich, M.T.M., Fonseca, C.M.: Computing hypervolume contributions in low dimensions: asymptotically optimal algorithm and complexity results. In: Evolutionary Multi-Criterion Optimization. Springer, Heidelberg (2011)
20. Guerreiro, A.P., Fonseca, C.M., Emmerich, M.T.: A fast dimension-sweep algorithm for the hypervolume indicator in four dimensions. In: CCCG, pp. 77–82 (2012)
21. Tušar, T., Filipič, B.: Visualizing 4D approximation sets of multiobjective optimizers with prosections. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 737–744. ACM (2011)

Parameter Identification of Stochastic Gene Regulation Models by Indicator-Based Evolutionary Level Set Approximation

Alexander Nezhinsky and Michael T.M. Emmerich(✉)

LIACS, Leiden University, Niels Bohrweg 1, Leiden, The Netherlands
m.t.m.emmerich@liacs.leidenuniv.nl
http://moda.liacs.nl

Abstract. Continuous level set approximation seeks to find a set of points (parameter vectors) that approximates the set of sets of parameters that for a given function map to a given output value. In this work we will look at a class of difficult to solve level set problems with innumerably many solutions and show how their solution sets can be approximated by robust evolutionary search methods.

In particular, this paper seeks to solve noisy parameter identification problem from biology where the task is to find the set of parameter settings of a stochastic gene regulatory network, simulated by Gillespie's algorithm with delays, that match existing observations. In this context the necessity of active diversity maintenance and adaptation of search operators to find all feasible subspaces is studied. As a result a robust implementation and default setting of evolutionary level set approximation (ELSA) for noisy parameter identification problems will be developed and validated. The validation uses two classical gene regulatory networks and it is demonstrated that a larger set of reaction parameters can be found that potentially could explain the observed stochastic dynamics.

Keywords: Stochastic simulation · Evolutionary level set approximation · Parameter identification · Stochastic gene regulatory networks

Given some function $f : \mathbb{R}^n \to \mathbb{R}$, a level set (approximation) problem (LSP) is the problem of finding (or approximating) the set of input vectors that are mapped to a function value below or equal to a certain threshold τ, i.e. approximating the set

$$L = \{x \in \mathbb{R}^n | \ f(x) \leq \tau\}$$

In the remainder L will be called the *level set* and f the *target function*.

We will also consider a variation of this problem, which is to approximately solve equation systems:

$$L = \{x \in \mathbb{R}^n | \ ||f(x) - T|| \leq \epsilon\}$$

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_4

where $T \in \mathbb{R}$ is some target value, and $\epsilon \geq 0$ is the desired minimal accuracy. Structurally the two problems are very similar, although in the second formulation the user needs to also define a desired minimal accuracy.

LSPs occur in various application domains. For instance the problem of finding solution sets of under-determined non-linear equation systems, finding solutions to constraint satisfaction problems, finding the basins of attraction for an attractor of a dynamical system, and parameter identification problems in systems modeling can be formulated as LSPs.

In the literature there are many methods described for solving such models by means of analysis and differential calculus. For differentiable functions, among these, level set continuation methods are commonly used. They start with a single feasible point that is for instance found by minimization of $||f(x) - \tau||$, and then construct new candidate points by using linear approximations to the level curves (the local tangent space). Due to linearization these approximations are not exactly on a desired position on the level curve and a correction step corrects for this error. By smart bookkeeping strategies, the level set is gradually covered by subsequent applications of such predictor corrector steps [6].

In this work we are interested in a method that also works for non-differentiable and noisy f. Moreover, the methods that are discussed in this paper are direct methods, meaning that they do not require knowledge about the internal structure of f and consider f as a black box.

The work is structured as follows: Sect. 1 introduces the evolutionary level set approximation (ELSA) algorithm and discusses how it differs from algorithms from diversity optimization, such as NOAH. Then, in Sect. 2 we discuss the application problem, the parameter identification of stochastic gene regulatory networks (SGRNs) and provide two concrete problems in Section sec:problems. In Sect. 4 we discuss the experimental setup and preliminary experiments on determining the noise level, which is required to set an appropriate accuracy. The results from the application of ELSA on the SGRNs are described in Sect. 5. In this section we also describe modifications to the standard ELSA algorithm that were required to make it solve the problem. The paper concludes with a summary of the main findings and highlighting of some important topics for future research (Sect. 6).

1 Evolutionary Level Set Approximation

Evolutionary level set approximation (ELSA) has been recently proposed as a heuristic solution method for LSPs [2]. ELSA is a population-based stochastic search algorithm that, if successful, converges to a set that is evenly distributed across the level set. In brief, one iteration of this algorithm can be described as the stochastic process on the state space $(\mathbb{R}^n)^\mu$ (populations of size μ):

The pseudo code of the basic ELSA method is provided with Algorithm 3. First a random start population of μ individuals is generated using stochastic operators. Then a new solution is created by stochastic variation based on a solution in P_t (or, in case of recombination, based on multiple solutions).

In the default case, it is created by uniformly randomly selecting an individual in P and mutating it by means of a small normally distributed perturbation. The selection method selects the subset of size μ that is optimal with respect to an augmented diversity measure. The algorithm is kept simple, because this makes it easier to analyze its dynamics. However, later in this work we will motivate a number of modifications of the basic schemes that will be required to solve the application problem.

Algorithm 3. Evolutionary Level Set Approximation

1. Initialize population P_0 of size μ.
2. Evaluate all solutions in P_0 using f and mark all infeasible solutions (i.e. solutions with $f(x) > \tau$).
3. While convergence is not reached
   (a) Generate a new solution $x_{new} \in \mathbb{R}^n$ by copying (some) solution(s) in P and applying an mutation (or recombination) operator.
   (b) $Q_t \leftarrow P_t \cup \{x_{new}\}$
   (c) Evaluate the solution using f.
   (d) Determine the minimal contributor x_{mc} to an *augmented diversity metric* in P_t.
   (e) $P_{t+1} \leftarrow Q_t \setminus \{x_{mc}\}$.
   (f) $t \leftarrow t + 1$.
4. Return P_t

1.1 Augmented Diversity Indicators

The augmented diversity measure considers with first priority the number of feasible solutions in a solution set. To evaluate feasible subsets, their diversity metric (indicator) will be assessed. Formally the augmented diversity indicator is constructed as follows:

$$Q^+(P) = \text{Diversity}(P \cap L) + \text{Penalty}(P - L),$$

where $P - L = \{x \in P |\ f(x) > \tau\}$ and $L \cap P$ can be computed by removing the infeasible solutions from P, because it holds

$$L \cap P = P - (P - L).$$

Now, given a set of infeasible solutions, say A, and an upper bound for the longest distance between two points in the feasible set (Diameter), the penalty is computed by

$$\text{Penalty}(A) := \mathit{Diameter} \cdot |A| + \sum_{x \in A} f(x) - \tau.$$

This way, for the diversity indicators that we consider in this work it is guaranteed that the a replacement of a infeasible solution by a feasible solution always

leads to an improvement of the indicator. Eventually, this will lead to a convergence of the ELSA algorithm to a diverse, feasible subset.

Different diversity and representative measures were suggested for guiding ELSA:

- Solow Polasky Diversity [9,10]
- Averaged Distance to Nearest Neighbor (ADNN), and
- Averaged Distance Indicator (ADI) [2]

The first two indicators are computed based on a distance matrix D_{ij}, that is the distance of all solutions in the approximation set P, $i = 1, \ldots, n$, $j = 1, \ldots, n$, $n = |P|$.

The *Solow Polasky Diversity* [9] is an indicator that is used in conservation biology for measuring biodiversity. It was used for diversity optimization by Ulrich and Thiele [10] and computes the sum of entries in the inverse of the matrix M with $m_{ij} = e^{-\theta D_{ij}}$. In detail, let $R = M^{-1}$ then $SP(P) = \sum_{i=1} \sum_{j=1} r_{ij}$. It is straightforward to conclude that the Solow Polasky diversity metric requires an appropriate choice of θ which is sensitive to the conditioning of M. This makes it difficult to apply in degenerate or noisy cases. The sensitivity to settings of θ was confirmed in the experiments reported in Emmerich et al. 2013.

The ADNN indicators comprise different types of averages of the distances to nearest neighbors in a set. Let $D(p, P)$ denote the distance of p to its nearest neighbor in $D \setminus \{p\}$. Then we define:

$$\mathrm{ADNN}_{Max}(P) = \max_{p \in D} \{D(p, P)\}$$

$$\mathrm{ADNN}_{\Sigma}(P) = \sum_{p \in D} D(p, P)$$

$$\mathrm{ADNN}_{\Pi}(P) = \prod_{p \in D} D(p, P)$$

Among these the $\mathrm{ADNN}_{\Pi}(P)$ turns out to be robust over a wide range of problems, why we will consider it in this work. As compared to the other two indicators, this indicator due to its product form favors solution sets where the variance of the distances to the nearest neighbor is relatively small. The result are sets that are more evenly spread across the search space.

In Emmerich et al. 2013 the ADI indicator was introduced as a third indicator. It is especially well suited to find representative finite approximations of point sets and is related to the averaged Haussdorff distance [8]. However, it requires Voronoi cell partitionings and is computationally more involved, especially in high dimensional search spaces. The approximation sets achieved with the computationally much simpler $ADNN_{\Pi}$ look very similar to those achieved with ADI, why we will consider only the $ADNN_{\Pi}$ indicator in this work.

Note, that as opposed to the ADI and SP diversity the ADNN indicators do not have the monotonicity property. The monotonicity property states that when

adding a point to a set the diversity should grow. This property is important when measuring diversity of sets of different size, but is not required in the context of ELSA, which features a constant population size and a penalty term that enforces that adding points to the feasible set (and therby removing a infeasible point) improves the indicator.

There is another conceptual downside with the ADNN indicator as compared to the previously mentioned indicators, which was not discussed in the previous literature. The following example will illustrate it:

Example 1. Assume two solution sets on the real line, that is $A = \{1, 2, 5, 6\}$ and $B = \{1, 2, 3, 4\}$ with all solutions feasible. A natural choice for a distance between two points on the real line would be $D_{ij} = |x_i - x_j|$, $i = 1, \ldots, 4$. Then $\mathrm{ADNN}_\Pi(A) = \mathrm{ADNN}_\Pi(B)$, although intuitively one would find A to be more diverse than B.

However, in the empirical studies that were conducted so far these cases did not affect convergence to well distributed sets. A reason for this could be that the globally optimal set requires solutions to be evenly spread. In the above example, for instance, if the usable part of the real line is the interval $[1, 7]$ the evenly spaced sets $A = \{1, 3, 5, 7\}$ is the unique maximum of ADNN_Π over all sets of size 4.

1.2 Theoretical Discussion

The process of ELSA can be described as a Markovian Process governed by the stochastic recursion:

$$P_0 = \mathrm{Init}(); P_{t+1} = \mathrm{Select}(P_t \cup \{\mathrm{Variate}(P)\}), t = 1, 2, \ldots$$

In theory, given a mutation operator with sufficient support, ELSA converges in probability to a local optimum of a diversity indicator and it is ascertained that all points will eventually reach the feasible set. The reason for this is that replacing an infeasible solution by a feasible solution always leads to the improvement of the augmented diversity indicator, and in case of equal sized feasible sets the diversity indicator of a set can only stay equal or increase.

2 Stochastic Simulation of Gene Regulatory Networks

A gene regulatory network is constituted by a collection of DNA segments within a cell which govern gene expression levels and protein concentrations over time. The DNA segments dynamically interact with each other through their RNA and protein expression products. Whereas classically gene regulatory networks were modeled deterministically by means of systems of ordinary differential equations, more recently stochastic simulation receives increased attention as biologists want to study adaptations in which stochasticity plays a crucial role. One such example is bet-hedging, a strategy where in a homo-genetic population different

phenotype traits are developed according to some evolved distribution in order to make the population robust to environmental fluctuations [11].

In this paper we are less concerned with the adaptive value of stochasticity but rather want to devise robust simulation and optimization tools to study these systems. The stochastic simulation of gene regulatory networks has been studied in past work and the common method is to use extensions of Gillespie's algorithms for reaction systems with delay [1,3] which was originally developed as a statistically sound and efficient simulation algorithm for chemical reaction systems with known reactions and reaction rates. It is typically used for modeling systems with a small number of molecules. To use these algorithms for simulating gene regulatory networks it is required to introduce delays, for instance the time it takes from a factor docking to a promotor to the start of translation, or small delays for the transport of messenger RNA fragments from the DNA to the ribosome.

We have used an example SGRN as shown in [7]. The network is a stochastic simulation of transcription and translation. It is modeled as a delayed chemical reaction. Each reaction i has a variable k_i which is the reaction's rates and proportional to the probability of a reaction to occur in a certain time interval. We have simulated the network over time t. As result we have the output data that consists of concentrations of certain proteins as each time stamp. We want to retrieve a feasible collection of input values k for the network, based on output concentration of certain molecules under the assumptions that the initial k values are not known during output analysis.

An overview of this process is given in Fig. 1. First we simulate a GRN with known k values. Resulting concentrations of certain proteins are represented

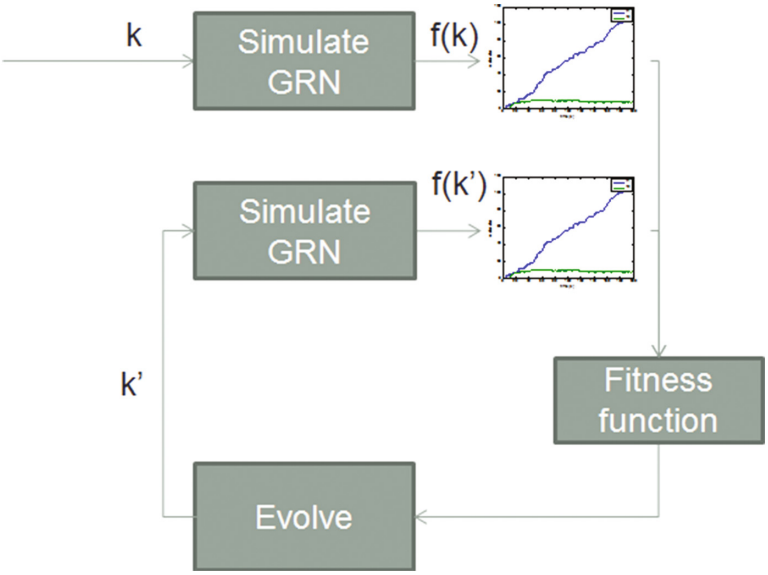


Fig. 1. Overview of the proposed framework.

as $f(k)$. Then we run an Evolutionary Algorithm to predict k' based on the output $f(k)$. The first iteration k'will not be predicted but randomly set. The values of k' are then again fed to the GRN simulator and concentrations of proteins are gained as $f(k')$. $f(k')$ is then compared to $f(k)$ based on some fitness function. The more the concentrations match each other the better the result of the fitness function. Based on the fitness function the quality of predicted $k'$is evaluated and the Evolutionary Algorithm is rerun.

2.1 Simulation of Gene Regulatory Network by Gillespie Algorithm

For simulation our GRN we use a stochastic model, since deterministic models may not accurately describe such systems when the number of molecules is small. Stochastic model simulation is done by the use of the Gillespie algorithm. Gillespie's algorithm consists of the following steps.

1. Initialization: Molecules, reaction rates (k) and random number generators are initialized during this step.
2. Monte Carlo sampling: Randomly choose next reaction to execute (multinominal distribution) and the time interval to the next reaction (exponential distribution). The probability of executing a reaction is proportional to the number of molecules and the reaction rates. Moreover, the time to the next reaction depends on these parameters, too.
3. Update: Increase the time by the chosen time interval and update the molecule count based on the chosen reaction outcome.
4. Iterate: Unless the simulation time has been exceeded, loop back to the Monte Carlo step.

3 Problem Definitions

We have chosen to simulate a problem that uses a GRN as described in *Ribeiro et al. 2006* [7]. The problem involves measurements of monomers. Problem simulations involve the possibility to select the presence or absence of two inducers. As a result of a single run of the network the amounts of two proteins r_1 and r_2 are of interest.

3.1 Monomer Problem by *Ribeiro et al.*

From the network that is proposed by [7] we chose the initial settings for the following reaction elements: $RNAP$, Pro_1, Pro_2, Pro_1r_2, Pro_2r_1, r_1, r_2, Ind_1, Ind_2. The following initial element amounts are assumed: $RNAP = 50$, $Pro_1 = 1$, $Pro_2 = 1$. The values τ_i denote delays after reactions. These are the times before a certain element again becomes available to the system. Before this time the element is locked (deactivated) in the stochastic simulation. In order to simulate the network with the presence of only the first inducer we set $Ind_1 = 1$ and $Ind_2 = 0$, to simulate the presence of both inducers we set $Ind_1 = 1$ and $Ind_2 = 1$. All other elements are set to 0.

Regular Reactions. Proposed reactions are taken from [7]. The stoichiometric reaction equations are given by:

1. $RNAP + Pro_1 \xrightarrow{k_1} Pro_1(\tau_1) + RNAP(\tau_2) + n_1 \times r_1(\tau_3)$
2. $RNAP + Pro_2 \xrightarrow{k_2} Pro_2(\tau_1) + RNAP(\tau_2) + n_2 \times r_2(\tau_3)$
3. $r_2 + Pro_1 \xrightarrow{k_3} Pro_1 r_2$
4. $r_1 + Pro_2 \xrightarrow{k_4} Pro_2 r_1$
5. $Pro_1 r_2 + Ind_1 \xrightarrow{k_5} Pro_1 + r_2 + Ind_1$
6. $Pro_2 r_1 + Ind_2 \xrightarrow{k_6} Pro_2 + r_1 + Ind_2$
7. $r_1 \xrightarrow{k_7}$
8. $r_2 \xrightarrow{k_8}$

Additionally some reactions do not instantly produce resulting elements after execution, but resulting elements are created with a certain delay. To account for this we introduce delayed elements: $RNAP(\tau_2)$, $Pro_1(\tau_1)$, $Pro_2(\tau_1)$, $r_1(\tau_3)$, $r_2(\tau_3)$. Delayed elements are treated as reactions. The delayed processes are then:

9. $RNAP(\tau_2) \xrightarrow{k_9} RNAP$
10. $Pro_1(\tau_1) \xrightarrow{k_{10}} Pro_1$
11. $Pro_2(\tau_1) \xrightarrow{k_{11}} Pro_2(\tau_1)$
12. $r_1(\tau_3) \xrightarrow{k_{12}} r_1$
13. $r_2(\tau_3) \xrightarrow{k_{13}} r_2$

The array n_i comprises constants that represent the rate of translation of mRNA. We assume $n_i = 1$. Stochastic rate reactions are assumed to be $1s^{-1}$, $k_1, \ldots, k_6 = 1.0$, with an exception of the decay reactions $k_7, k_8 = 0.001$. All values are taken from [7].

As described in [4] (page 3) the half lives τ can be converted from corresponding rate constants by $\tau_i = \frac{\ln 2}{k_i}$. And thus $k_i = \frac{\ln 2}{\tau_i}$. For $\tau_2 = 20.0$ we set $k_9 = \ln 2/20.0$, for $\tau_1 = 1.0$ we set $k_{10}, k_{11} = \ln 2$, and for $\tau_3 = 10.0$ we set $k_{12}, k_{13} = \ln 2/10.0$. for $\tau_3 = 10.0$ we set $k_{12}, k_{13} = ln2/10.0$.

Propensities of Reactions. Propensities are the probability that the reaction given in equation i occurs and are depending on availability of elements on the left side of the equation and stochastic rate reaction k_i for the reactions 1 to 13:

$$
\begin{aligned}
P_1 &= k_1 \times RNAP \times Pro_1 \\
P_2 &= k_2 \times RNAP \times Pro_2 \\
P_3 &= k_3 \times r2 \times Pro_1 \\
P_4 &= k_4 \times r_1 \times Pro_2 \\
P_5 &= k_5 \times Pro_1 r_2 \times Ind_1 \\
P_6 &= k_6 \times Pro_2 r_1 \times Ind_2 \\
P_7 &= k_7 \times r_1
\end{aligned}
$$

$$
\begin{aligned}
P_8 &= k_8 \times r_2 \\
P_9 &= k_9 \times RNAP(\tau_2) \\
P_{10} &= k_{10} \times Pro_1(\tau_1) \\
P_{11} &= k_{11} \times Pro_2(\tau_1) \\
P_{12} &= k_{12} \times r_1(\tau_3) \\
P_{13} &= k_{13} \times r_2(\tau_3)
\end{aligned}
$$

In order to validate the implementation of the new MATLAB simulator, we compared its result with that of the GRN simulator by Ribeiro et al. [7]. The amounts of two proteins r_1 and r_2 are measured over time.

3.2 Objective Function

The objective function allows us to compare how well a proposed solution looks like the original solution. In our case we compare the graphs that represent concentrations of the resulting proteins over time: r_1 and r_2 for the monomer problem. An often used measurement in case of comparing two graphs would be a root mean square error (RMSE). In order to be able to perform RMSE an interpolation step of the resulting data is needed since the two sets can have a different time distribution.

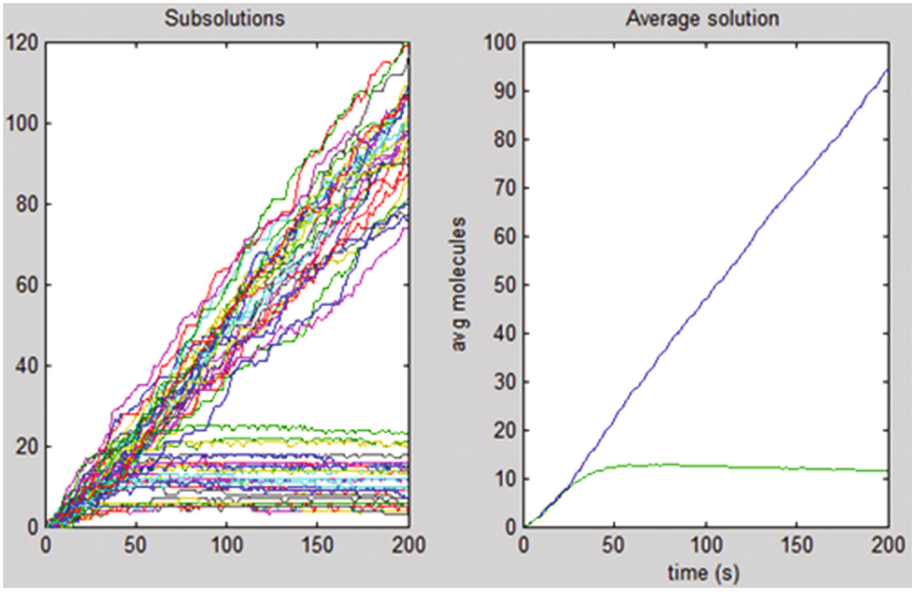


Fig. 2. Sampled trajectories (left), Known average trajectory of molecular concentrations (right)

Calculating the fitness by considering a single run RMSE however is not satisfactory. Due to the stochasticity of the simulation, RMSE with same k give different results. Therefore it is unrealistic to demand a zero RMSE. The bandwidth of the simulation results can be assessed from Fig. 2 (left). A solution would be to average the $f(x)$ from over multiple runs. We would like to estimate what the amount of runs would be to retrieve a trustworthy RMSE while keeping the amount of rerunning the network to a minimum as shown in Fig. 2 (right).

4 Experimental Setup

Next, experiments are described for testing the ability of ELSA to find correct and alternative settings for k. For both considered problems we compute time series for the given k. Then we use ELSA to find back the parameters.

4.1 Monomer Problem

To calculate a trustworthy $RMSE(A, B)$ for model A and model B of we compute A m times, then take an average of the runs $\bar{A} = \frac{1}{m}\sum_{i=1}^{m} A_i$. In this case this is the average of concentrations of $r_1$and r_2 at each time stamp. Averaging is done over the runs. Then the same for solution B is done, B is recalculated n times into $\bar{B}$. Then we consider the value of $RMSE(\bar{A}, \bar{B})$ instead of $RMSE(A, B)$ for the calculations.

In order to determine optimal number of m and n when comparing two solutions we assume that A and B are actually created from the same input values, and thus the error between them should be minimal. To accomplish for that we recalculate A $m = 30$ times. After that we try different values for n and compare the $RMSE(\bar{A}, \bar{B})$ values. All of the experiments were in addition rerun 100 times (for each experiment $RMSE(\bar{A}, \bar{B})$ was recalculated 100 times and average a and standard deviation σ computed. The following results were acquired:

1. For $m = 30$, $n = 1$: $a = 115$, $\sigma = 67$
2. For $m = 30$, $n = 10$: $a = 36$, $\sigma = 15$
3. For $m = 30$, $n = 20$: $a = 29$, $\sigma = 11$

From these results we can see that a and σ do not change after $n = 20$ (values are the same for $n = 20$ and $n = 30$). Therefore it is safe to assume that for correct $rmsa(\bar{A}, \bar{B})$ can be calculated with $n = 30$, $m = 20$.

4.2 Gene Expression Problem

In a similar way as for the monomer we need to determine a sufficient number of m_c and n_d when comparing the two solutions C and D for the gene expression problem. To accomplish for that we recalculate C $m_c = 30$ times. After that we try different values for n and compare the $RMSE(\bar{C}, \bar{D})$ values. The following results were acquired.

1. For $m = 30$, $n = 1$: $a = 75$.
2. For $m = 30$, $n = 2$: $a = 56$.
3. For $m = 30$, $n = 10$: $a = 27$, $\sigma = 4$.
4. For $m = 30$, $n = 15$: $a = 22$, $\sigma = 2$.
5. For $m = 30$, $n = 20$: $a = 19$, $\sigma = 2$.
6. For $m = 30$, $n = 30$: $a = 19$, $\sigma = 2$.

From these results we can see that a and σ does not change after $n = 20$, anymore. Therefore we assume that for correct $RMSE(\bar{C}, \bar{D})$ can be calculated with $n = 30$, $m = 20$;

5 Level Set Approximation Results

Evolutionary algorithms are population based optimization algorithms. Population evolves by making use of such concepts such as mutation, crossover and fitness-based selection. The root mean square error fitness function determines the quality of solutions (= candidate parameter sets). We have evaluated CMA-ES and ELSA approaches. CMA-ES provides a single solution, is quasi parameter-free and is optimized for noisy data. CMA-ES can be extended with niching which makes sure multiple optima being maintained in niches [5].

5.1 CMA-ES Evaluation

We ran CMA-ES for the monomer problem with the following settings: $m = 30$ $n = 1$ $f_{min} = 115$ $eval_{max} = 5000/n$. We have chosen $n = 1$ for speed performance, we have chosen $f_{min} = 115$ since $a = 115$ when two sets are equal and compared with $m = 30$ $n = 1$. The results that were obtained during 5 runs: $f < f_{min}$ was reached at: 4160 evaluations, 2810 evaluations, > 5000 evaluations, 1040 evaluations and 1080 evaluations. In Fig. 3 (Right, Middle) some resulting graphs are shown. Additionally we have tried running CMA-ES with niching enabled with the following settings: $m = 30$ $n = 1$ $f_{min} = 115$ $eval_{max} = 5000/n$ and the number of niches $q = 2$. The algorithm was rerun 5

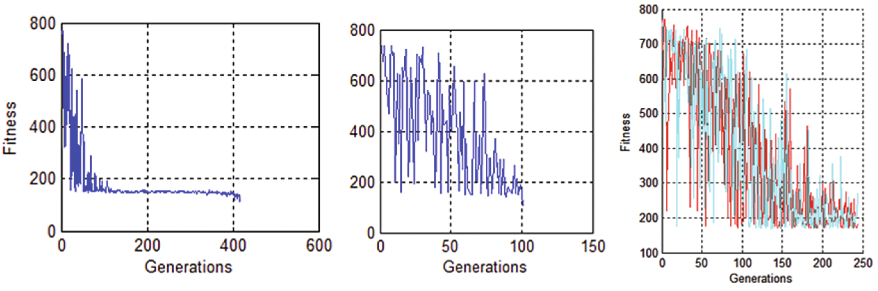


Fig. 3. Left, Middle: CMA-ES convergence history. In the cases above $f < f_{min}$ was reached at: 4160 evaluations (416generations) and 1040 evaluations (104 generations). Left: CMA-ES with niching (2 niches) evaluation results.

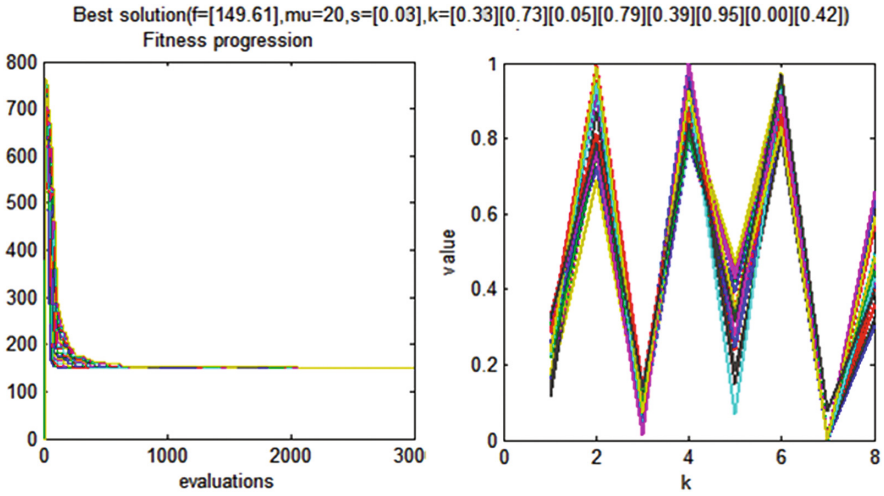


Fig. 4. ELSA evaluation results. In the case above $f < f_{min}$ was never reached within 5000 evaluations (left graph). In the right graph we see that all the possible solutions collapse to almost the same values.

times, but $f < f_{min}$ was never reached within 5000 evaluations. However, again the lacking ability of the CMA-ES to find a feasible solution was preventing in this case to find any feasible solution. In Fig. 3 (Right).

5.2 ELSA Evaluation

We ran ELSA with the following settings: $m = 30$ $n = 1$ $f_{min} = 115$ (error threshold), $eval_{max} = 5000/n$, and 14 points to approximate the level set. The algorithm was rerun 5 times, but $f < f_{min}$ was never reached within 5000 evaluations. A notable problem was that all the solutions would collapse to same pattern as shown in Fig. 4. In order to improve on these results we have considered minor modification of the default ELSA algorithm. Within the default ELSA algorithm child solutions are generated from parents by multi-point mutation. In our ELSA modification child solutions are created either by random generation (ratio 0.25), random recombination (ratio 0.25), one-point mutation (ratio 0.25) or generated from parent by multi-point mutation (ratio 0.25).

We ran this ELSA modification with the same settings as previously: $m = 30$ $n = 1$ $f_{min} = 115$ $eval_{max} = 5000/n$ with 14 levelsets. The algorithm was rerun 5 times, $f < f_{min}$ was reached at the following amount of evaluations:

1. First point $< f_{min}$ at iteration 173, all points $< f_{min}$ at iteration 775;
2. First point $< f_{min}$ at iteration 907, all points $< f_{min}$ at iteration 1448;
3. First point $< f_{min}$ at iteration 1452, all points $< f_{min}$ at iteration 2059;
4. First point $< f_{min}$ at iteration 2118, all points $< f_{min}$ at iteration 3316;
5. First point $< f_{min}$ at iteration 657, all points $< f_{min}$ at iteration 1164;

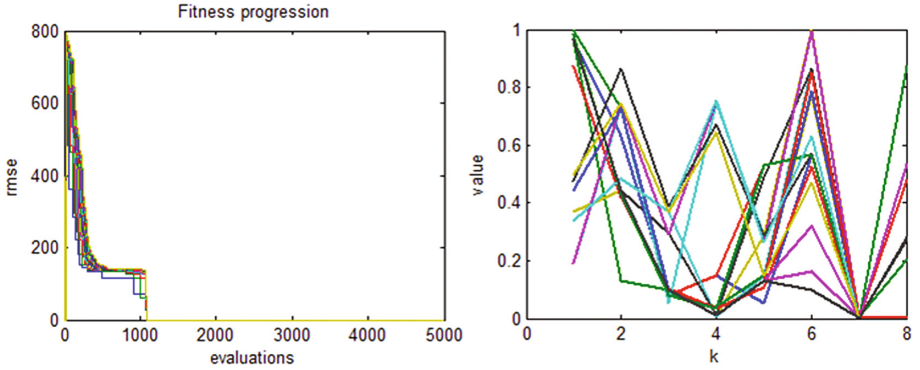


Fig. 5. Modified ELSA evaluation results. In the case above $f < f_{min}$ was never reached just after 1000 evaluations (left graph). In the right graph we see that the possible solutions show a high variety of values.

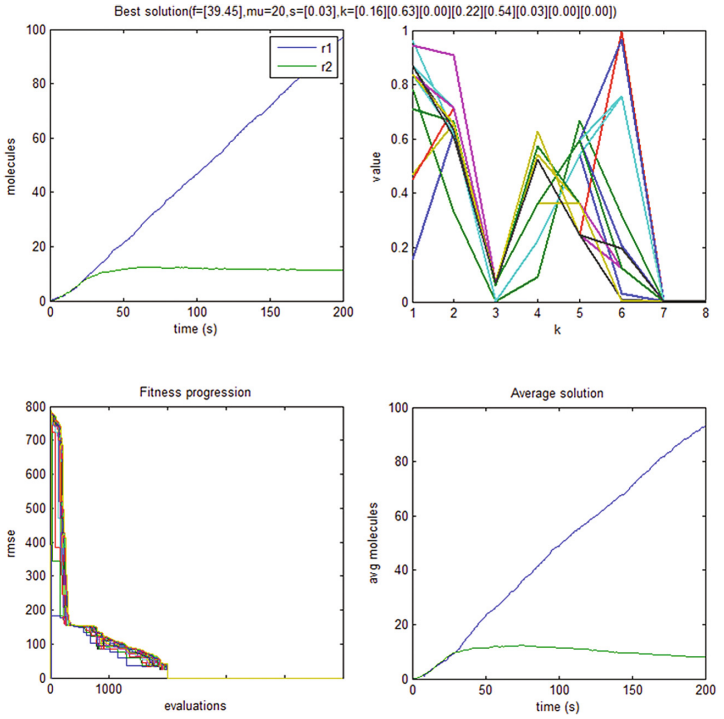


Fig. 6. Top left graph is the ground truth graph or r_1 and r_2 concentrations. Bottom left graph is one of the runs is shown with convergence of the results. Top right graph shows the 14 levelsets found by the algorithm. Bottom right graph is ther_1 and r_2 concentrations created from the best solution.

We can see that all the levelsets reached f_{min} within $eval_{max} \approx 2000$ evaluations which makes the modified ELSA a better algorithm then the standard approach CMA-ES for this problem. Additionally, the solution space now shows a high variety of different solutions, as can be seen in Fig. 5.

Now that we have found a good approach for the problem we want to create even less noisy solutions. In order to do so we adjust values of n during runtime. Initial run settings:$m = 30$ $n = 1$ $f_{min_1} = 115+67$ $eval_{max} = 5000/n$. When the solutions with these settings are found, $f < f_{min_1}$ for all levelsets, we continue running but with the following settings: $n = 2$ $f_{min_2} = 78+48$. When $f < f_{min_2}$ for all levelsets, we continue running but with the following settings: $n = 10$ $f_{min_3} = 36+15$. When $f < f_{min_3}$ for all levelsets, we continue running but with the following settings: $n = 20 f_{min_4} = 29 + 11$. Results show fast convergence (Fig. 6) with fitness values around 30.

6 Conclusions and Outlook

ELSA was used as a method to calibrate parameters in stochastic gene regulatory networks (SGRN). Stochastic simulation by Gillespie's algorithm is used to compute a stochastic trajectory. The SGRN (Ribeiro et al. 2006) was used as a test case. The system has 8 reaction rates (prospensities). The 'task' for ELSA was to find the reaction rates that give rise to an observed average trajectory (see Fig. 1, right). The systems behaviour was given by the averaged trajectory of the number of certain molecules. As for ELSA the true reaction rates were unknown it has to find them back by minimizing the RMSE until it is below the threshold and to find a diverse set of solutions with RMSE below threshold. The study showed that the standard version of ELSA and a restart method based on the uncertainty handling CMA-ES (Hansen et al. 2009) did not manage to generate diverse sets, whereas, after some modification of the mutation operator, introducing a random switch between large and small mutation, ELSA was able to identify a diverse set of reaction parameter settings (see Fig. 1, left). Each one of the found parameter settings can be a possible explanation of the system's behaviour and further knock out experiments would have to be conducted to narrow down the search space to a single solution. The main conclusion from the method is that not only the selection method is crucial for obtaining a sufficient performance but also the variation operators. The interplay between the two need to be in the focus of future scientific investigations.

Acknowledgements. The authors gratefully acknowledge financial support by the Netherlands Science Organization (NWO) within the Computational Life Science/ BETNET Project.

References

1. Cai, X.: Exact stochastic simulation of coupled chemical reactions with delays. J. Chem. Phys. **126**(12), 124108 (2007)
2. Emmerich, M.T.M., Deutz, A.H., Kruisselbrink, J.W.: On quality indicators for black-box level set approximation. In: EVOLVE-A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation, pp. 157–185. Springer (2013)
3. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. **81**(25), 2340–2361 (1977)
4. Hayot, F., Jayaprakash, C.: Nf-κb oscillations and cell-to-cell variability. J. Theor. Biol. **240**(4), 583–591 (2006)
5. Li, R., Eggermont, J., Shir, O.M., Emmerich, M.T.M., Bäck, T., Dijkstra, J., Reiber, J.H.C.: Mixed-integer evolution strategies with dynamic niching. In: Parallel Problem Solving from Nature–PPSN X, pp. 246–255. Springer (2008)
6. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. AMS, vol. 153. Springer, New York (2003)
7. Ribeiro, A., Zhu, R., Kauffman, S.A.: A general modeling strategy for gene regulatory networks with stochastic dynamics. J. Comput. Biol. **13**(9), 1630–1639 (2006)
8. Schutze, O., Esquivel, X., Lara, A., Coello Coello, C.A.: Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. IEEE Trans. Evol. Comput. **16**(4), 504–522 (2012)
9. Solow, A.R., Polasky, S.: Measuring biological diversity. Environ. Ecol. Stat. **1**(2), 95–103 (1994)
10. Ulrich, T., Thiele, L.: Maximizing population diversity in single-objective optimization. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 641–648. ACM (2011)
11. Veening, J.-W., Smits, W.K., Kuipers, O.P.: Bistability, epigenetics, and bet-hedging in bacteria. Annu. Rev. Microbiol. **62**, 193–210 (2008)

Evolution in ICT Security

On Using Cognition for Anomaly Detection in SDN

Emilia Tantar(✉), Alexandru-Adrian Tantar, Miroslaw Kantor, and Thomas Engel

Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, 4 Rue Alphonse Weicker, 2721 Luxembourg, Luxembourg
{emilia.tantar,alexandru-adrian.tantar, miroslaw.kantor,thomas.engel}@uni.lu

Abstract. Through this position paper we aim at providing a prototype cognitive security service for anomaly detection in Software Defined Networks (SDNs). We equally look at strengthening attack detection capabilities in SDNs, through the addition of predictive analytics capabilities. For this purpose, we build a learning-based anomaly detection service called Learn2Defend, based on functionalities provided by Opendaylight. A potential path to cognition is detailed, by means of a Gaussian Processes driven engine that makes use of traffic characteristics/behavior profiles e.g. smoothness of the frequency of flows traversing a given node. Learn2Defend follows a two-fold approach, with unsupervised learning and prediction mechanisms, all in an on-line dynamic SDN context. The prototype does not target to provide an universally valid predictive analytics framework for security, but rather to offer a tool that supports the integration of cognitive techniques in the SDN security services.

1 Introduction

Security in Software Defined Networks (SDNs) spans over different layers, following mainly two perspectives: (i) the security of an SDN infrastructure in itself and (ii) enhancing the security of already existing networks brought to the SDN context. The SDN architecture offers the advantage of a centralized holistic perspective over the entire network and a clear separation between the data plane, the control plane and the applications layer. This opens the path for service improvement and implicitly for security as a service, built on top of the controller offered functionalities.

Different security aspects are under investigation, ranging from policy enforcement, to traffic anomaly detection [19] and mitigation [10] or DoS attack detection and mitigation [4]. The global network perspective enables also coping with aspects as malicious administrators behavior, including misconfigurations [17]. Tools were also developed for enabling configuration integrity checks [1], or going even further and enabling virtualization [25]. Initial efforts

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_5

have been made [14] also in proposing good practice guidelines to protect SDN networks.

Although progress is continuously being reported, one major issue requiring further investigation relates to the efficiency and scalability of security mechanisms, in front of the big data challenge, as flows are continuously being generated. On a more practical note, security applications, built on top of the controller, depend on the efficiency of the south-bound protocol statistics collection and processing [10]. Although several protocol alternatives were developed, e.g. ForCES [28], OpenFlow [18] imposes as a *de facto* standard, being both supported and promoted by the Open Networking Foundation(ONF) and providing the needed communication collaborative ground that enables the integration of different vendors equipment for example. Technical solutions are being developed, e.g. by making use of combined sFlow and OpenFlow capabilities, for raising the data collection burden. This empowers anomaly detection of large scale malicious events, measured through metrics such as the entropy. Nevertheless the data analysis still remains a major concern and when looking at the cognitive dimension emerged in SDN for security purposes, one cannot help in noticing that control loops and autonomic network management are amongst the ancestors of cognitive techniques for SDN [27]. Active security prototypes for SDN [13] make use for example of feedback control loops. Security control loops have been reviewed [27] and their advantages and drawbacks highlighted.

Since the eighties [6], statistical approaches and machine learning or cognition oriented approaches remain the driving forces in providing security in the networking context, and this remains valid for SDN also. Neural networks, K-means clustering or Markov models are only some of the application examples in the SDN context for anomaly detection [16]. Naive techniques make use of fixed confidence bounds in detecting anomalous behavior, with variations such as moving average [23] to adapt to the dynamically fed traffic flows. However, when considering these paths, one should be cautious, as no universal approximator exists and a good understanding of the network characteristics can provide valuable results [26].

What catches the eyes, from the implementation perspective, is the open-source development of controllers, with candidates as Pox [22], Nox [11], Floodlight [8], Beacon [7], Opendaylight [21] and the need of a prototype platform enabling fast and easy integration of predictive analytics behavior in a highly controllable environment. Through this paper we address this issue by proposing an open-source prototype for Cognitive security in SDN, for the purpose of anomaly detection, called Learn2Defend.

The remainder of this paper is organized as follows. We start with a brief overview of anomaly detection, in Sect. 2. Subsect. 2.1 provides an outline of Gaussian Processes, followed by a description of the data collection and training algorithms in Subsect. 2.2 (extension of the Defense4All framework to include cognition mechanisms). Section 3 provides a structural view of the prototype solution we built, including a description of the testbed and brief preliminary showcase running results. We end with conclusions in Sect. 4.

2 Prototyping Cognitive Behavioral Security: Anomaly Detection

Following the classification provided in [6], various types of statistical profiling can be put in place based on collected data. The Learn2Defend prototype, proposed hereafter, can be considered as an intrusion detection system, according to the taxonomy of [5] as historical attack data can be made available from the controller, while topological network information and statistics regarding the current network status can be also provided from the controller.

Behavioral security prototypying in SDN was previously considered for attack detection and mitigation in the context of data centers; considering statistics targeting both switches and routers [15]. Following the emergence of virtualization, the proposed perspective was built on top of an NFV/SDN context. In the follow up we provide prototypying an attack detection/mitigation system by means of open-source platforms. Do note that a thorough knowledge of the network particularities is essential in fine tailoring and choosing the appropriate tools for performing online data analysis, as in the case of classical networks [26].

We start by depicting, in Fig. 1 the prototype cognitive security module, together with controller functionalities of interest and inherited communication

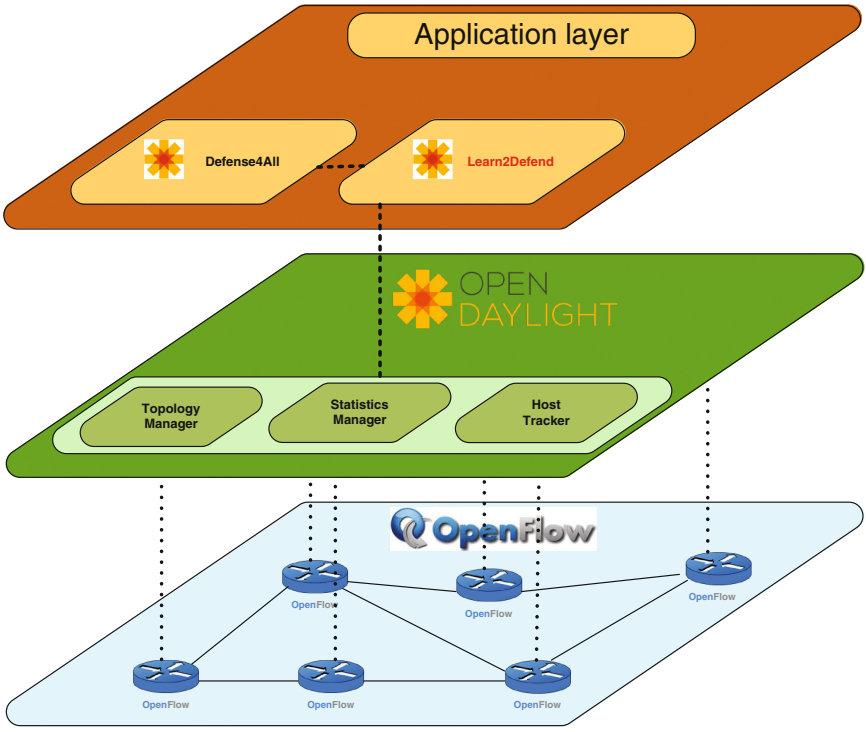


Fig. 1. SDN Holistic abstraction, including cognitive security third party application.

module. This is followed by an architectural view of the road to cognition, from the Gaussian processes modeling perspective.

2.1 A Road to Cognition: Gaussian Processes Modeling

The machine learning domain, in general, and anomaly detection, in particular, include a large array of methods. Examples include decision trees, artificial neural networks, support vector machines or Bayesian methods [2,3,24]. Some of these methods are intended for clustering, some for classification while others look at regression problems. However, one is generally determined to select one class over another based on criteria such as the nature of the data to work with, expected output, flexibility or, more importantly, trade-off between training and prediction/classification time. With respect to the area we deal with, we chose to use Gaussian Processes (GPs) [24] mainly due to their non-parametric nature, high flexibility and interesting training-prediction trade-off (at the scale of our instances). A strong advantage GPs have is their ability to easily integrate any prior knowledge with respect to data by selecting or adapting some specific kernel function (details to be provided later). Moreover, GPs where shown, for some specific classes, to be a direct equivalent of a neural network with infinitely many hidden units (limit case) [20]. Another aspect we are interested in is being able to give confidence bounds for our predictions.

Gaussian Processes (GP) can be thought of as a generalization of the Gaussian distribution [24]. If samples are visualized as functions (spanning an infinite number of points), if any linear combination of any finite number of points along these functions has a multivariate Gaussian distribution, the process is said to be Gaussian.

A GP is a non-parametric stochastic model capable of providing predictions given a subset of 'observed' points. It makes use of random variables with a joint multivariate Gaussian distribution. The model in itself can be seen as an unknown function that maps any input point $x \in \mathbb{R}^d$ to an output $y = f(x)$, $y \in \mathbb{R}^n$, with the implicit underlying assumption that $f(x)$ is a (multivariate) Gaussian random variable. The values $y_1, \ldots, y_n$ can be thus seen as being drawn from $\mathcal{N}(\mu, \Sigma)$, where $\Sigma_{i,j}$ is the covariance of the outputs (that correspond to the inputs x_i and x_j). For a GP, the covariance matrix is modeled by a kernel, denoted in the follow-up as $K(x_i, x_j)$; this is in fact the core of the GP model.

A GP model is fully described by its mean $\mu(x)$ and covariance function $K(x_i, x_j)$; the covariance can also be interpreted as a distance metric between two input points. Not surprisingly, the quality of a GP strongly depends on the kernel. As the kernel models the interactions and overlap between the various components, it should consider, if possible, the nature of the data to deal with, e.g. smoothness, cyclic or repeating patterns. This is however not always possible. In our case, specifically, the way data looks is strongly influenced by the environment it originates from (hardware components, network topology, users, usage, etc.), and we can not consequently make any before-hand assumptions. Some examples of kernel functions include constant, linear, Gaussian noise or squared exponential functions, among many others.

For the scope of our study and for prototyping, we decided to make use of a Squared Exponential (SE) invariant kernel (also known as a Radial Basis Function or Gaussian kernel), defined as follows:

$$K(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|^2}{2\ell^2}\right) \tag{1}$$

where ℓ is a *characteristic length-scale* [24]; such a kernel implies a smooth process. If this assumption is thought not to hold given a specific e.g. network or hardware resources setup, one can also refer to using a Matérn class kernel; despite being more computationally intensive (not directly fit for our case), the Matérn class of covariance functions has the advantage of being able to infer smoothness from the data. For details on GPs or kernels, please refer to [24]. Also, please do note that while a Gaussian assumption may (indeed) not hold in all (network traffic) cases, it was preferred working with a Squared Exponential kernel given that it offers (i) a sufficient tradeoff in terms of descriptive coverage and computational constraints (w.r.t. this specific application and setup); and (ii) a sufficient approximation of the different stochastic processes that combine into a single signal, i.e. as observed in dense traffic patterns.

Using the SE provides the equivalent of a Bayesian linear regression where the covariance is defined as a linear combination of an infinite number of Gaussian basis functions [9,24]. Among others, the SE definition assumes a high correlation among close points while distant points are uncorrelated. This fits our setup and study framework, while being however easily inter-changeable to fit any other (more complex or significantly different) case.

A simple approach, if such a kernel refinement is needed, is to extend the anomaly detection engine (described later more in detail) to make use not of one but of a set of differently configured GPs; one can then simply select a best-fit kernel or use a voting-like approach. Note that it is also possible to construct a completely new kernel by combining existing ones (according to some well defined rules, not detailed as out of scope for our study), mixing, for example, patterns for cyclic and increasing trends or smoothness, if some prior knowledge about the data is available.

We identify first the basic needs of such a cognitive security protoype: (i) enable real-time data gathering and (ii) provide support for out-of-the-box machine learning techniques. The cognitive behavioral security pipeline is as follows: a counter is instantiated on a specific port for a specific protocol, multiple protocols can be considered for the same port. Build a training set based on historical data, create the initial behavioral model based on the training data. Online dynamically adapt the model corresponding to the regular profile based on newly feed data (the re-training phase). Use only labeled data for the online training.

Also, as with using a regression-based approach, it is possible in a similar manner to interpose a classification component. The main constraint one has to consider at this point is however the incurred computational load. An e.g. random forest based approach may, in specific case, be more effective but, at the

same time, may pose a too significant load when in dense traffic. An extensive comparative analysis is considered for a future study where several network setups and load cases need to be surveyed. For the limit of this study we only referred to the (already deployed in live traffic) baseline moving average based detection.

2.2 Learning and Building Profiles

The framework is broken into several components, e.g. filtering, pre-processing, classification and/or regression-based detection. We namely make use of two independent processes, one for building a training set and the second one for identifying anomalous flows, pseudo-code in Algorithm 1 and Algorithm 3, respectively. The aim is to allow a flexible design where different algorithms can be inter-connected in a (relatively) black-box manner. For our scope and w.r.t. the fixed aim of study, we address the problem of detecting outliers via a regression-based approach. A significant deviation from a pre-trained model, in this setup, is reported as a potential sign of e.g. potential intrusion/abnormal traffic.

Both the training set construction and the anomaly detection algorithms make use of what we will refer to as 'instances', i.e. fixed length arrays of data flow bytes. Algorithm 1 uses a sliding window approach to construct the training set. The algorithm is deployed in live traffic conditions. For our concept we rely on raw data (assembled into instances), e.g. size in bytes associated to (a subset of) keys to monitored sampled based on time of arrival (irregular); it is equally possible using, for example, traffic patterns or online (running) statistics. Traffic logs, i.e. packet size for a given (subset of) key(s), are assembled as samples and then used to build a baseline model. Logs, w.r.t. our configuration and in most of our interest cases, show a stable profile and relatively high correlation degree. Note that the current (real-life deployed) detection approach, nonetheless sufficiently effective, relies on a moving average statistics.

For anomaly detection we build an 'observed' instance in a similar manner (sliding window at the level of single readings and not over full instances). The size of an instance (number of attributes) as well as the number of instances required for training are fixed in advance. The main reason for using such an approach is to remain coherent even if trend shifts appear in the data. While we consider a Gaussian Process prediction model, other similar models or classifiers can be used as well, either independently or simultaneously, e.g. making use of differently sized windows, for short- and long-term views, or using offline screened data exclusively.

The training set construction algorithm is called in iterative manner (online dataset construction) and, the algorithm by itself, iterates as long as flow data is available; for this example we assume that we look at bytes. All flow data not labeled as being for training is skipped (lines 3–5). This allows one to control when model updates are (or should not be) made. A traffic key is generated next, based on the protocol and port pair (line 6). If the key is monitored, then we enter the training set construction segment of the algorithm (lines 8–22).

Algorithm 1. Online training set and Gaussian Processes predictor construction.

```
while flow data available do
  data ← flow data

  if data is not for training then
    continue;
  end if
  key ← { protocol, port }

  if key is monitored then
    if fIndex ¡ numAttributes then
      trainingInst{ fIndex } ← data.bytes
      fIndex ← fIndex + 1
    end if

    if fIndex = numAttributes then
      if ‖ trnset ‖ = learningThreshold then
        trnset ← trnset / trnset{ 0 }
      end if

      trnset ← trnset ∪ trainingInst

      instance ← {empty instance}
      for j ∈ [0, numAttributes − 1] do
        instance{ j } ← trainingInst{ j+1 }
      end for
      trainingInst ← instance
      fIndex ← numAttributes-1

    end if

    if ‖ trnset ‖ = learningThreshold then
      detector ← build { trnset }
    end if
  end if
end while
```

Please do note that it is possible to specify all or only a limited set of keys to be monitored, with no restriction, as required. If we did not yet fill a full training instance (line 8), we append the value read to the instance and increment the field/attribute 'fIndex' value correspondingly (lines 9–10); 'fIndex' is assumed to be a global variable, first initialized to 0. If now the training instance is complete (line 12), we can add it to the training set. If we reached the learning threshold (number of instances to trigger the construction of the detector component), we

first discard the oldest instance in the training set (lines 13–15), and then we add the training instance (line 16) constructed at the previous step. All values in the instance used to collect flow data (for this key) are shifted to the left (lines 17–22); this step makes place for a future flow data value. Last, inside the 'key is monitored' branch, if the learning threshold was reached, we build the detector (lines 24–26).

Algorithm 2. Multiobjective Optimization by Cooperative Swarms (MOCOPS)

Input initial population P_0
while termination criterion is not reached **do**
  $t \leftarrow t + 1$
  $s \sim u(\{1, 2, \ldots, n-1, n\})$
  $\mathbf{x}_{old} = \mathbf{x}^{(s)}$
  $P \leftarrow P_t \setminus \{\mathbf{x}^{(s)}\}$
  {Try to improve position of particle $\mathbf{x}^{(s)}$}
  $\mathbf{z} \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_{new} = \mathbf{x}_{old} + \sigma \cdot \mathbf{z}$
  if $HV(P \cup \{\mathbf{x}_{new}\}) > HV(P \cup \{\mathbf{x}_{new}\})$ **then**
    $P_t \leftarrow P \cup \{\mathbf{x}_{new}\}$
  else if $HV(P \cup \{\mathbf{x}_{new}\}) < HV(P \cup \{\mathbf{x}_{old}\})$ **then**
    $P_t \leftarrow P \cup \{\mathbf{x}_{old}\}$
  else if $f_1(\mathbf{x}_{new}) + f_2(\mathbf{x}_{new}) < f_1(\mathbf{x}_{old}) + f_2(\mathbf{x}_{old})$ **then**
    $P_t \leftarrow P \cup \{\mathbf{x}_{new}\}$
  else
    $P_t \leftarrow P \cup \{\mathbf{x}_{old}\}$
  end if
end while
Return P_t

For an outline of the data collection process, please see Algorithm 3. We assume that index is a 'globally' visible variable, initially set to 0 (first element of an instance). The algorithm iterates while there is still flow data available. We first retrieve the flow key (line 2) and, if the key is among the ones being monitored, we enter the data collection/prediction segment (lines 3 to 25). Next, we collect flow data (line 4). If 'index' is less than 'numAttributes' (number of attributes/size of an instance), we add the newly read data to 'observed' at the position currently indicated by 'index'. The result is that we construct the instance as new data is read. Then, 'index' is incremented to indicate the next position available in the 'observed' instance. If we have a complete instance ('index' = 'numAttributes', line 9), we can screen 'observed' using detection (Gaussian Processes for our case, pre-trained using Algorithm 1). Namely, if the learning threshold, i.e. minimum number of instances for training, was reached, we determine the expected value μ and the variance σ of the flow (lines 11 and 12). If the observed (actual) value is outside the 95% interval ($[\mu - 2\sigma, \mu + 2\sigma]$), then we add the '{protocol, port}' pair to the set of anomalous

Algorithm 3. Outline of the process used to read data and that runs predictions with a confidence bounds check.

```
while flow data available do
  key ← flow key
  if key is monitored then
    data ← flow data { key }
    if index ¡ numAttributes then
      observed{ index } ← data.bytes
      index ← index + 1
    end if
    if index = numAttributes then
      if reached learning threshold then
        μ ← detector.classify{ observed }
        σ ← detector.stdev{ observed }
        if data.bytes ∉ [μ − 2σ, μ + 2σ] then
          idPair ← { protocol, port }
          anomalous ← anomalous ∪ idPair
        end if
      end if
      instance ← {empty instance}
      for j ∈ [0, numAttributes − 1] do
        instance{ j } ← observed{ j+1 }
      end for
      observed ← instance
      index ← numAttributes-1
    end if
  end if
end while
```

flows. Note that the variance is calculated with respect to the context data stored in 'observed' and is not some fixed value; it is also determined by the nature of the data that was used for training. Having filled a full instance, i.e. 'observed', we now need to accommodate a new flow data reading. Lines 18–23 discard the first reading in 'observed' and shift all other values to the left; the 'index' is set accordingly (the last array entry in 'observed' can now receive a new value). The loop terminates when no more flow data is available. The algorithm by itself is called in iterative manner whenever data becomes available.

3 Learn2Defend: Cognitive Security in SDN Preliminary Prototype

The Learn2Defend cognitive behavioral security prototype has been developed by having as base architecture the open-source Defense4all [23] communication mechanisms and interfacing them with the Weka [12] machine learning open platform capabilities. A general layered system architecture of Learn2Defend is depicted in Fig. 2.

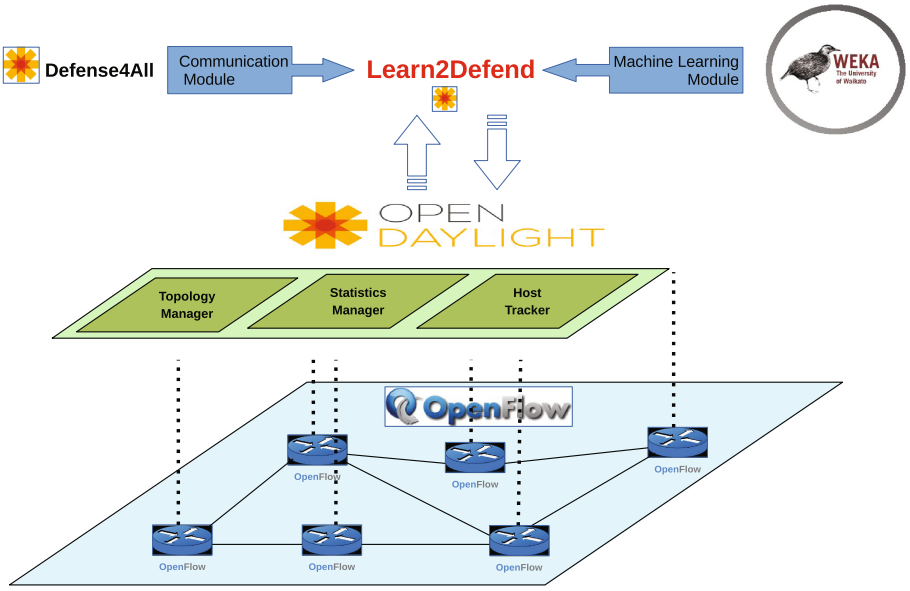


Fig. 2. Learn2Defend architecture and module inheritage.

It should be noted, that although built on top of the Defense4all communication and data collection modules, the Learn2Defend service runs as an independent service. No dependencies are required from the Defenese4all, as the needed modules have been integrated entirely in the service.

3.1 Testbed Description

In order to test Learn2Defend functionalities in a realistic scenario, we prepared a testbed consisting of 8 TP-LINK TL-WR1043ND v1.8 wireless routers (see Fig. 4) with an underlying network topology as depicted in Fig. 3.

This commodity hardware was updated with Pantou firmware which supports OpenFlow and OpenWrt5, which is a common Linux-based operating system allowing embedded devices to route network traffic. Such a modified firmware supports version 1.0 of OpenFlow, and version 10.03.1 (Backfire) of OpenWrt.

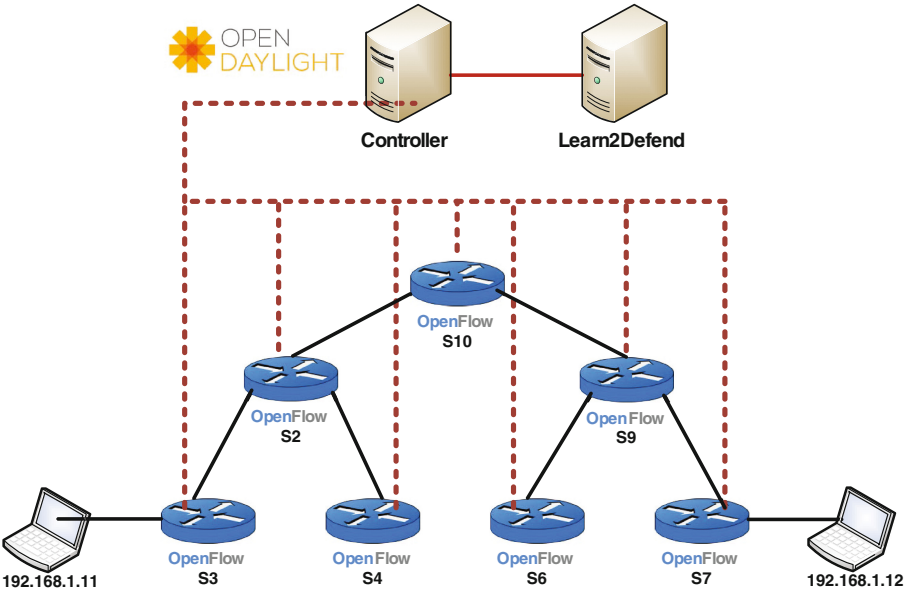


Fig. 3. Testbench infrastructure, including a separate machine hosting the Learn2Defend service.

Fig. 4. Image of the physical testbed.

As an SDN controller we used Hydrogen release of the OpenDaylight [21] controller (ODL). The OpenDaylight controller communicates with OpenFlow switches and exposes a unified interface to program OpenFlow switches. The communication with the Learn2Defend application is performed using the ODL REST API. Two additional hosts with an Ubuntu 12.04 64-bit OS were used to demonstrate different attack and fault scenarios in the network. A snapshot

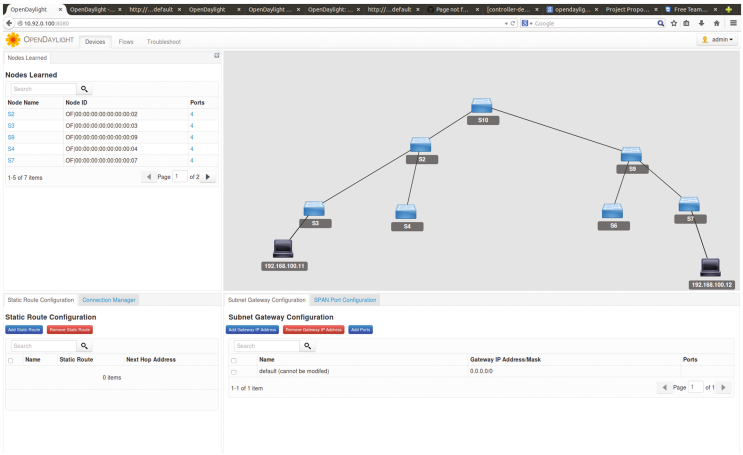


Fig. 5. View of the testbed depicted by the OpenDaylight GUI.

Fig. 6. ICMP testing - using the Defense4all service (top) and Learn2Defend service (middle). TCP testing using Learn2Defend (bottom).

of OpenDaylight GUI with implemented physical topology has been presented in Fig. 5.

Through Learn2Defend we aim in providing enhanced attack detection capabilities, by bringing the power of predictive analytics and incorporating machine learning capabilities for the attack/anomaly detection in SDN. Some captured attacks are illustrated hereafter, obtained employing confidence bounds in Defense4all for TCP flows (Fig. 6), respectively per switch behavioral profiling by means of Gaussian processes (TCP flows - Fig. 6 and ICMP traffic Fig. 6).

Although promising concerning the time required to perform attack detection, we aim further in covering long-lived/short-lived, small flows/large flows and following on the parameters tunning proposed in [15]. In parallel, it is also envisaged a real-life data feeding of the network in order to approach with a higher fidelity a production scenario.

4 Conclusion

Through this paper we provide a first view at a prototype solution, open-source platform for anomaly detection. Learn2Defend brings the scalability and flexibility of predictive analytics into the SDN attack detection world. In addition to extending Defense4All, the main achievement we have with respect to the proposed prototype solution, is that we offer a first view of a potential cognitive Gaussian Processes based architecture. Among other features, it allows including prior knowledge about the data to monitor via specific or custom designed kernel functions. Furthermore, the solution can be easily extended to simultaneously look at short- and long-term correlations, capture different patterns or use a set of classifiers in a voting-like scheme. No substantial change, in terms of implementation, is required as we exploit the 'unified' interface Weka provides, i.e. moving from one predictive method/classifier to a different approach is merely a question of changing only a few lines of code (declaration and parameter specification).

The entire design makes use of online data collection, and thus allows to easily switch from one type of monitoring to another, as required (or perform a full new retraining). Ongoing work is made for an extensive testing on real-life data, with future refinements on mitigation mechanisms, e.g. via virtualized honeynets.

Acknowledgment. This publication is based in parts on work performed in the framework of the IDSECOM project, INTER/POLLUX/ 13/6450335, and CoSDN project, INTER/POLLUX/12/4434480, both funded by the Fonds National de la Recherche, Luxembourg.

References

1. Al-Shaer, E., Al-Haj, S.: Flowchecker: configuration analysis and verification of federated openflow infrastructures. In: Sager, T., Ahn, G.-J., Kant, K., Lipford, H.R. (eds.) SafeConfig, pp. 37–44. ACM (2010)
2. Bishop, C.M.: Pattern recognition and machine learning. In: Information science and statistics. Springer, New York (2006)
3. Bishop, C.M., Nasrabadi, N.M.: Pattern recognition and machine learning. J. Electron. Imaging **16**(4), 049901 (2007)
4. Braga, R., Mota, E., Passito, A.: Lightweight ddos flooding attack detection using nox/openflow. In: IEEE 35th Conference on Local Computer Networks (LCN), 2010, pp. 408–415, Oct 2010
5. Debar, H., Dacier, M., Wespi, A.: Towards a taxonomy of intrusion-detection systems. Comput. Netw. **31**(8), 805–822 (1999)
6. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. **13**(2), 222–232 (1987)
7. Erickson, D.: The beacon OpenFlow controller. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013, pp. 13–18. ACM, New York (2013)

8. Floodlight project. http://www.projectfloodlight.org
9. Genton, M.G.: Classes of kernels for machine learning: a statistics perspective. J. Mach. Learn. Res. **2**, 299–312 (2002)
10. Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., Maglaris, V.: Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. Comput. Netw. **62**, 122–136 (2014)
11. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S.: Nox: towards an operating system for networks. SIGCOMM Comput. Commun. Rev. **38**(3), 105–110 (2008)
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
13. Hand, R., Ton, M., Keller, E.: Active security. In: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, HotNets-XII, pp. 17:1–17:7. ACM, New York (2013)
14. Kreutz, D., Ramos, F.M.V., Veríssimo, P.J.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
15. Krishnan, R., Krishnaswamy, D., Mcdysan, D.: Behavioral security threat detection strategies for data center switches and routers. In: IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2014, pp. 82–87, June 2014
16. Kukliński, S., Wytrebowicz, J., Dinh, K.T., Tantar, E.: Application of cognitive techniques to network management and control. In: Tantar, A.-A., et al. (eds.) EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V, pp. 79–93. Springer, Cham (2014)
17. Matsumoto, S., Hitz, S., Perrig, A.: Fleet: defending sdns from malicious administrators. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN 2014, pp. 103–108. ACM, New York (2014)
18. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. In: Proceedings of the ACM SIGCOMM 2008 conference, vol. 38(2), pp. 69–74 (2008)
19. Mehdi, S.A., Khalid, J., Khayam, S.A.: Revisiting traffic anomaly detection using software defined networking. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) Recent Advances in Intrusion Detection. Lecture Notes in Computer Science, vol. 6961, pp. 161–180. Springer, Heidelberg (2011)
20. Neal, R.M.: Bayesian Learning for Neural Networks. Springer, New York (1996)
21. OpenDaylight project, 01 May 2015. http://www.opendaylight.org
22. POX controller. http://www.noxrepo.org/pox/about-pox
23. Radware. Defense4All, User Guide (2014) https://wiki.opendaylight.org/view/Defense4All:Main
24. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). MIT Press, Cambridge (2005)
25. Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., Parulkar, G.: FlowVisor: A Network Virtualization Layer. Technical report , Deutsche Telekom Inc. R&D Lab, Stanford, Nicira Networks (2009)
26. Sommer, R., Paxson, V.: Outside the closed world: On using machine learning for network intrusion detection. In: IEEE Symposium on Security and Privacy (SP), 2010, pp. 305–316, May 2010

27. Tantar, E., Palattella, M.R., Avanesov, T., Kantor, M., Engel, T.: Cognition: a tool for reinforcing security in software defined networks. In: Tantar, A.-A., et al. (eds.) EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V, Advances in Intelligent Systems and Computing, vol. 288, pp. 61–78. Springer, Cham (2014)
28. Yang, L., Dantu, R., Anderson, T.A., Gopal, R.: Forwarding and Control Element Separation (ForCES) Framework, RFC 3746. The Internet Engineering Task Force, April 2004

Feature Creation Using Genetic Algorithms for Zero False Positive Malware Classification

Razvan Benchea(✉), Dragos Gavrilut, and Henri Luchian

Faculty of Computer Science, Iaşi, Romania
{rbenchea,gdt,hluchian}@infoiasi.ro
http://www.infoiasi.ro

Abstract. This paper presents a Genetic Programming approach to feature extraction in the frame of the perceptron algorithm described in [1]. While feature extraction has the potential of increasing the accuracy of classification, fully exploring the huge space of possible combinations of the initial 45150 features would make the approach infeasible; Genetic Programming provides a proper way of tackling the search for relevant features. In turn, the extracted features are used to train an algorithm - One Side Class Perceptron - designed to minimize the number of false positives; accuracy is increased. In the experiments, the classifier using the extracted features was run on a dataset consisting of 358,144 files. The results show that our overall approach and implementation is fit for real-world malware detection.

Keywords: Genetic programming · Genetic algorithms · Malware detection · Feature creation · Feature extraction

1 Introduction

Due to increase in computing power in the last decade we are now able, in the training phase, to explore more of the feature space and combine existing features into newer ones in order to achieve a better separability. In this paper we will try to develop new features from existing ones in order to achieve a better classification accuracy on large datasets.

We work with a modified version of the perceptron algorithm, called the One Side Class Perceptron (OSCP) [1] and we try to modify its features in order to achieve a better classification. Even though we know that by combining each feature with the others through a method called mapping we can increase the accuracy significantly, this method has some serious drawbacks when it comes to using it on an ordinary computer due to the large memory space needed to store this huge feature set. Our purpose is to find a smaller set of features, obtained through a series of transformation from the original feature set, that can produce a similar result with the mapped version of the perceptron but using a smaller amount of memory. In order to achieve this we will use genetic programming to evolve features that are good at separating records from two classes.

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_6

In this paper we only limit the operators for the genetic feature creation to the boolean set of operator $\{and\ ,\ not\}$; *and* is used for crossover operations while *not* is used for mutation. We choose to work only with this limited set because we want to achieve a similar result to the mapped version of the OSCP algorithm which only uses the *and* operator. This way, we can compare the results in a more accurate way. In a future paper, the operator set will be expanded in order to see if this improves the accuracy.

Using the methods described in this paper we create features that when used with the OSCP algorithm achieve the same accuracy as selecting the best 600 features among the 358144 created by the mapped version of the OSCP. We take the experiments one step further and try to find the minimum number of features needed to achieve the same results as the mapped version. As it turns out, we only need 4000 features, 15 times less required by the OSCP mapped version.

Our domain of application is malware detection and more exactly the process of separating clean executable files from malicious ones. Since we want this algorithm to be integrated into an antivirus product there are some important restrictions that limit the algorithm classification ability.

1. The first one refers to the number of false positives: since marking a non-malicious file as infected can lead to system corruption or important document loss it is very important that the algorithm performs at a very low false positive rate. Even though this may lead to a lower detection rate, this is preferable since in practice, the rest of the detection rate is assured by other mechanisms (i.e. hand written heuristics).
2. The second restrictions is about speed: since an antivirus product works in the background and scans almost every file being accessed by almost every process, it is very important to not bring an overhead to the system that will affect the user experience. For this reason, the testing phase of the algorithm must be very fast. The training speed is not very restrictive, but of course, since we are testing the algorithm on large datasets it is important that the training will finish in a reasonable amount of time.

The paper is further organized in the following sections: Sect. 2 describes the methods used by other authors of applying genetic programming in order to create features or classify malicious and benign files. Section 3 describes our methodology, the problems we encountered when testing the algorithms as well as the results. In Sect. 4 we bring our conclusions and plans for future work.

2 Related Work

In this section we discuss different approaches taken by other authors in order to solve similar problems. Even though the application of these methods is different than ours they all have in common feature creation using genetic programming.

Two distinct patterns can be observed when trying to combine features extracted using genetic algorithms with a classifier. The difference lies in the construction of the fitness function:

For the first case, the feature creation process and the classification process are well divided. The first process, which is the feature creation, uses a fitness function that increases the chance that the resulted features will be good at discriminating different classes of records. After the first process is finished, the classifier can use the original features in order to achieve a good accuracy. Even though the feature creation process is fast it only increases classifier accuracy if this is dependent on the fitness function used in the genetic algorithm. This is also the approach that we take in this paper. Examples of algorithms that use this approach can be found in [2] and [3]. The authors of [2] achieved a 75% increase of the maximum available speed of a program by trying to correctly predict loop unrolling in a compiler. This was obtained by using a decision tree as a classifier while the fitness function evaluated how close to the right unroll factor did the features get. In [3] authors try to detect faults in mechanical bearings by creating features that are a function of the original attributes. The resulted features are based on trigonometric functions and arithmetic operators. The fitness function used is the Fisher criterion which is very similar to our fitness function. The authors test these features using a multi layer perceptron as well as a support vector machine, both of them achieving a 6% increase in accuracy. However, in both of these cases, the dataset is very small, of only 160 samples in [3] and 57 records in [2].

For the second case, in order to develop features that are best suited for a certain classifier, the fitness function uses the classifier algorithm to decide which features survive to the next generation. While this method has the aforementioned advantage is not well suited for large datasets since a training procedure must be carried on for each generation. Such examples can be found in [4–8]. Even though it is applied in different way, the k-NN algorithm, used by [4,6] and [5] seems to produce good results when used in relation to the fitness function. The authors of [6] used it to evaluate the weight adjusted using a genetic algorithm. By also using a genetic algorithm for feature selection they achieved a 79% accuracy. Weight adjusting using genetic programming for a k-NN classifier was also adopted by [5]. Like in the approach we use in this paper, the authors also combined original features between them using the *and* operator. This reduced the error rate from 88% to 0%. Finally, the authors of [4] tested features created using mathematical operators that are able to find relations between close pixels by using both a k-NN classifier as well as a decision tree. The decision tree provided better results.

A more generic chromosome was described by authors of [7] in a framework called Eprep. Each chromosome stores one of the three possible classification algorithms (Generalized Linear machine, k-NN and Maximum Likelihood), as well as a mathematical combination of the original features created using non linear functions. The fitness function decides the best chromosomes by computing the number of misclassified samples from a validation set. From the 9 public datasets tested, on 2 of them the error rate has dropped by half when classification was done using these new features. A reduction rate from 11.9783 to 0.0887

was achieved by authors of [8] by using a svm in the fitness function for feature extraction and selection.

When it comes to malware detection using genetic programming, the results obtained by two different papers, [9] and [10], are quite different. Both of the authors tested the classification task using evolutionary and non-evolutionary algorithms. In the first paper, most of the evolutionary algorithms (XCS, UCS, Gassist-ADI, GAssist-Intervalar and Slave) were outperformed by non evolutionary ones (iterative rule learners: RIPPER, SLIPPER and PART, decision tree: C4.5 and an instance based rule learner: RIONA) in terms of accuracy. In the second comparison between genetic and non-genetic algorithms, good results were obtained by UCS (85.28%) as well as the SVM. However, the support vector machine completely missed a malware family. The differences between these result could be due to the way features are being constructed, since the authors of [10] used network traffic as attributes. An attempt to compare a classifier that uses weights adjusted using a genetic algorithm and non evolutionary algorithms can be found in [11]. The features are created using n-grams of function calls. When using n-grams of size 6, the classifier achieved 0% false positives and outperformed other non-evolutionary algorithms (svm, decision tree, propositional rule learner (ripper) and naive bayes). However, the dataset is too small (280 files) to consider the malware classification problem solved.

The authors of [12] implemented a system called Realgo that tries to mimic the human immune system. It works by trying to evolve "antigens" that are similar in structure to known virus signatures. For false positives reduction, the antigens are validated on a dataset of clean files and if files are detected, the antigen is removed. Even though this system seems interesting, validating it on a very large dataset might result in poor antigens due to possible noise in the database.

3 Methodology

3.1 Motivation

As briefly detailed in the introduction, we use the OSC perceptron algorithm for classification. This algorithm works as a normal perceptron algorithm that has a separate training phase after each iteration where the model generated by the previous iteration is trained with the items that belong to a certain class (in our case items that represent clean files). This phase is repeated until every item from a specific class is correctly classified. While this algorithm has the advantage of obtaining 0 false positive in the training phase it also decreases the detection rate significantly.

The experiments performed in this paper are carried on using a database of 358,144 records that correspond to 42940 malware files and 315204 clean files collected over the last 2 month. The idea is to use only fresh malware files (malware that appeared in the last couple of month) as this is more relevant for the malicious attacks landscape of these days. For each of these files, 300 boolean features are extracted. These features represent behavior characteristics

extracted using an emulator (i.e. if it copies itself into different location, if it modifies certain registry keys, if it connects to different services or Internet addresses, etc.) and static information that is extracted from the file (i.e. if it is linked to a certain library, it if has certain strings, if it uses certain x86 assembly instructions and so on). The initial database was larger than this one containing more than 500000 files. In practice these databases have different files that are considered noise. One example would be a file infector. While a file infector is a malware file, the features extracted from this file are likely to be very similar if not identical to the one of the original-uninfected host file. This can result in two records that belong to different classes to have the same set of features (or very similar ones). As the OSP algorithm is design to make sure that all of the record that belong to a specific class are correctly classified (in our case the benign files), the detection rate decrease dramatically in this case-that specific records will act as an out-lier in the database. As file infectors are not suited for this kind of detection, we decided to remove this kind of files (and similar ones like keygens, adware, spyware) from the database.

Once the dataset is created we perform a 3-fold cross validation using the OSC perceptron algorithm in the following way:

- The first approach is to use the methodology presented above, where we use the features as they are present in the dataset.
- For the second one we create a new set of attributes by combining each feature with all the others.

Given the original set of features $F = \{F_1, F_2, F_3, F_n\}$, where in our case $n = 300$, the new set of features, $F' = \{F'_1, F'_2, F'_3, F'_k\}, where\ k = \frac{n \times (n+1)}{2}$, are generated in the following way:

Algorithm 1. Mapping method for OSC algorithm

```
F' ← {}
for i = 1 → n do
  for j = i → n do
    F' ← F' ∪ {F_i ∧ F_j}
  end for
end for
```

We refer to this method of obtaining new features as mapping. The results can be seen in Table 1. The detection rate is significantly improved by the usage of the mapping method. However, the memory footprint is increased exponentially in this case. If we consider that each feature is stored into one bit of memory, the original set consisting of 300 features occupies approximately 38 bytes for each record. Since the database consists of 358144 records its memory foot print is around 14 MBytes. However, when the mapping method is used, the number of features is 45150; this means 5644 bytes for each record and the entire database needs approximately 2 Gb of memory (more than 150 times bigger than the

initial one). While 2 Gb is acceptable for current hardware, when using larger databases, it may be unfeasible to use this method.

The next logical step is to figure out a method for reducing the number of new features while preserving the detection rate. The first idea is to sort all of the 45150 features after the F2 score and select the best 600 features and use only them. The F2 function is presented in Eq. 1.

$$F2_i = \frac{(\mu_i^+ - \bar{\mu_i})^2 + (\mu_i^- - \bar{\mu_i})^2}{(\sigma_i^+)^2 + (\sigma_i^-)^2} \tag{1}$$

Where μ_i^+/μ_i^- represent the means of the positive and negative sets, and σ_i^+/σ_i^0 represent the standard deviations of the positive and negative sets. For more information about the F2 measure the reader is invited to consult [13].

The results of using the first 600 features according to the F2 function can be seen in Table 1.

Table 1. The results of the 3-fold cross-validation for OSP, OSP-MAP algorithms

Algorithm	Detection	False positive	Accuracy
OSP	51.12%	17	94.96 %
OSP-MAP	85.35%	44	98.21 %
OSP-MAP-BEST-600-F2	65.55%	15	95.86 %

Even though the results obtained from using the best 600 features are better it is important to notice that these were obtained only after generating all 45150 features. So, even though we do not solve the memory problem, we can try to use these results as a threshold. More exactly, we will try to achieve similar results without going through the process of generating all the features.

3.2 Feature Creation Using Genetic Algorithms

In order find the best features that can solve our problem we use genetic algorithms.

Since the mapping method achieved very good results by just using the *and* operator, we decide to use the same operator in order to produce new chromosomes. So, every chromosome is actually a series of features each one being connected with the others using the *and* operator. More precisely, given the original set of features $F = F_1, F_2, ..., F_n$, we consider a chromosome $C = G_1, G_2, ...G_k$, where G_i is a gene within this chromosome. G_i is represented as an integer, non-null number, that respects the following condition: $1 <= |G_i| <= n$, where G_i refers to the i-th feature from F.

The crossover operation is implemented by randomly choosing a splitting point of two chromosomes and exchanging genes between them. The splitting

Algorithm 2. Convert Chromosome to Feature

function $ConvertChromosomeToFeatureF$,C
$F' \leftarrow \{\}$
for all $G \in C$ **do**
  if $G > 0$ **then**
    $F' \leftarrow F' \cup \{F_{G-1}\}$
  else
    $F' \leftarrow F' \cup \overline{\{F_{G-1}\}}$
  end if
end for
end function

point can be any value from 0 to the length of the chromosome. This way, chromosomes of just one gene can be combined without any other necessary validation. Since the *and* operator is commutative, in case a gene is used multiple times in a chromosome, only one instance is kept.

When it comes to the mutation operation, this is implemented using the *not* boolean operator. However, using the not operator can lead to some chromosomes achieving the lowest fitness score if the same gene is used in both the positive and the negative form (resulted from negation operator). To prevent this, we eliminated both genes, thus considering that the genes are canceling each other. After the evolution process is finish, each chromosome is converted into a new feature according to Algorithm 2. Note that, internally, the features that are mutated (they have a not operator applied), are represented using negation.

On each generation, a total of 1000 crossover operations are being performed by randomly selecting a pair of chromosomes and choosing a crossover point. In order to make the algorithm use as many of the original features as possible, on each generation the original features are also added. This also brings more diversity into each generation. The mutation is being applied with a probability of 3% on the chromosomes resulted after the crossover operation.

The fitness function used is the F2 function. This is computed on each of the 1000 offspring as well as their parents (300 chromosomes) and only the best 300 are being kept.

The Evolution process is described in the algorithms bellow. Considering P, a population of individuals, P is initialized according to Algorithm 3.

Algorithm 3. Initial Population

function $InitialPopulationF$
$P \leftarrow \{\}$
for all $F_i \in F$ **do**
  $C \leftarrow \{i + 1\}$
  $P \leftarrow P \cup \{C\}$
end for
end function

The steps taken to evolve one generation to another are detailed in Algorithm 4.

3.3 Problems and Solutions

In the attempts to get more powerful features, several problems were encountered. These problems, as well as the solutions applied are presented in the following paragraphs.

In our first attempts, even though the chromosomes score kept getting better with every generation, the classification results were decreasing. After analyzing the chromosomes an interesting aspect can be observed: many of the features are actually prefixes for the others. Even though each of them has a very high fitness score, when used together in the classifier they are actually treated as a single feature since they separate the same set of files.

An example of the resulted features can be seen in Table 2. The first and the second columns represent the number of benign and malicious files, respectively, where the chromosome is being activated, while the last column represent the chromosome. The numbers that make up the chromosome are the original feature index. If it is a negative number, then a not operator (resulted from mutation) is being applied on that feature. As it can be seen, all of the 4 chromosomes actually separate the same number of benign and malicious files since they all contain the genes $-237, -52, 239, 245, 246$. A classification score computed on different iterations can be seen in Table 3. Using these chromosomes in a classifier will bring the same result as using only one of them.

Table 2. Problematic features with high F2 score

Benign records	Malicious records	Feature chain
26023	583	−237, −52, 239, 245, 246, 254,
26023	583	−237, −52, 239, 245, 246, 252, 266,
26023	583	−237, −52, 239, 245, 246, 251, 252, 254
26023	583	−237, −52, 239, 245, 246, 251, 252, 254, 255, 266

When investigating the results we observe that more than half of the resulted chromosomes are created in this way. This explains why the detection rate decreases with every iteration.

In order to fix this we limit the length of the resulted chromosomes to a constant. Even this does not fix the problem it blocks the chromosomes from getting too big after the first iterations.

Another problem that we observe is that the linear separability between certain records decreases after using the new features. This is because there are many features that are being activated in both of the benign and malicious files. In order to solve this problem the original features are also added,

Algorithm 4. Evolve Population

function *EvolvePopulationP*
combination $\leftarrow 0$
newPop $\leftarrow \{\}$
while *combination* $<$ *combinationsThreashold* **do**
$C1 \leftarrow P[Random(|P|)]$
$C2 \leftarrow P[Random(|P|)]$
$cut1 \leftarrow Random(|C1|)]$
$cut2 \leftarrow Random(|C2|)]$
$C1_{left} = \bigcup_{i=1}^{cut1} C1_i$
$C1_{right} = \bigcup_{i=cut1+1}^{|C1|} C1_i$
$C2_{left} = \bigcup_{i=1}^{cut2} C2_i$
$C2_{right} = \bigcup_{i=cut2+1}^{|C2|} C2_i$
$CLL \leftarrow \{\}$
$CRR \leftarrow \{\}$
$CLR \leftarrow \{\}$
$CRL \leftarrow \{\}$
if $Random(100) > 50$ **then**
$CLL \leftarrow C1_{left} \cup C2_{left}$
$CRR \leftarrow C1_{right} \cup C2_{right}$
if $Random(100) > mutationThreashold$ **then**
$index1 = Random(|CLL|)$
$CLL_{index1} = -CLL_{index1}$
$index2 = Random(|CRR|)$
$CRR_{index2} = -CRR_{index2}$
end if
else
$CLR \leftarrow C1_{left} \cup C2_{right}$
$CRL \leftarrow C1_{right}t \cup C2_{left}$
if $Random(100) > mutationThreashold$ **then**
$index1 = Random(|CLR|)$
$CLR_{index1} = -CLR_{index1}$
$index2 = Random(|CRL|)$
$CRL_{index2} = -CRL_{index2}$
end if
end if
if *CLL not empty* **then**
$newPop \leftarrow newPop \cup \{CLL\}$
end if
if *CRR not empty* **then**
$newPop \leftarrow newPop \cup \{CRR\}$
end if
if *CLR not empty* **then**
$newPop \leftarrow newPop \cup \{CLR\}$
end if
if *CRL not empty* **then**
$newPop \leftarrow newPop \cup \{CRL\}$
end if
end while
return *newPop*
end function

Random(n) generates a random number from 0 *to n using an uniform distribution*

Table 3. Results on different iteration for arbitrary length features

Iteration number	Detection
1	64.67 %
2	64.83 %
5	66.15 %
10	61.83 %
20	58.4 %
40	59.6 %

thus obtaining a total of 600 features. We test this new method selecting the best 300 chromosomes from the generic algorithm, adding the 300 original features and limiting the size of a chromosome to a constant. The results can be seen in Table 4.

Table 4. Results on different iterations for limited length features

Iteration number	Detection
1	63.82 %
5	68.31 %

As it can be observed in Table 4, after 5 iterations the detection rate is bigger than the one obtained using the best 600 features provided by the mapping method. As this method provides good results (with only 600 features), we decide to increase the number of features that will be generated by the genetic algorithm and see if we can get close to the detection rate provided by the mapping method (85%). We also increase the size of the population from 1000 to 6000 chromosomes in order to give the algorithm a chance to produce better individuals that will translate in relevant features. Table 5 shows that with 3000 or 4000 features we get close to the detection rate obtained by the mapping method. (85.35% detection rate) but with 15 times less features.

All of the tests in this paper were carried out using a computer with Intel Xeon CPU E5-2440 (2.4 ghz, 24 cores) with 24 Gb of ram and running Windows

Table 5. Results using different number of features

Number of features	Population size	Detection rate	False positives	Accuracy
1000	3000	73.01%	21	96.74%
2000	5000	77.16%	30	97.24%
3000	6000	80.66%	31	97.65%
4000	8000	81.16%	34	97.71%

Server 2008 SP1. The OSCP algorithm was tested in a 3-fold cross validation limited to 500 iterations.

4 Conclusions and Future Work

In the beginning of the paper we proposed to achieve a good detection rate, similar to the one provided by the OSCP algorithm but with a smaller memory foot-print. We achieved this by using a genetic algorithm and generating 15 times less features than our target. We also presented some pitfalls that one should avoid when using this kind of strategy - mainly the creation of multiple features derived from the same predecessors that will only isolate a limited amount of records from the database

For the future we plan to test the same method using multiple Boolean operators as well as bringing some parallelism to the future creation. This will allow us to test our method on larger datasets. We also plan to analyze and test different techniques for controlling bloat in the variable length chromosome. Another aspect that we are currently interested is the proactivity provided by the genetic generated features.

References

1. Gavrilut, D., Benchea, R., Vatamanu, C.: Optimized zero false positives perceptron training for malware detection. In: SYNASC, pp. 247–253 (2012)
2. Leather, H., Bonilla, E., O'Boyle, M.: Automatic feature generation for machine learning based optimizing compilation. In: Proceedings of the 7th Annual IEEE/ACM International Symposium on Code Generation and Optimization, CGO 2009, Washington, DC, USA, pp. 81–91. IEEE Computer Society (2009)
3. Guo, H., Jack, L., Nandi, A.: Feature generation using genetic programming with application to fault classification. IEEE Trans. Syst. Man Cybern. Part B Cybern. **35**(1), 89–99 (2005)
4. Kowaliw, T., Banzhaf, W., Kharma, N., Harding, S.: Evolving novel image features using genetic programming-based image transforms. In: 2009 IEEE Congress on Evolutionary Computation, CEC 2009, pp. 2502–2507, May 2009
5. Pei, M., Goodman, E.D., Punch, W.F.: Feature extraction using genetic algorithms. In: Proceedings of International Symposium on Intelligent Data Engineering and Learning'98 (IDEAL98), Hong Kong, p. 98 (1997)
6. Punch, W., Goodman, E., Pei, M., Chia-Shun, L., Hovland, P., Enbody, R.: Further research on feature selection and classification using genetic algorithms (1993)
7. Sherrah, J.R., Bogner, R.E., Bouzerdoum, A.: The evolutionary pre-processor: automatic feature extraction for supervised classification using genetic programming. In: Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M., Iba, H.,Riolo, R.L., (eds.) Genetic Programming 1997: Proceedings of the Second Annual Conference, Stanford University, CA, USA, pp. 304–312. Morgan Kaufmann (1997)
8. Ritthoff, O., Klinkenberg, R., Fischer, S., Mierswa, I.: A hybrid approach to feature selection and generation using an evolutionary algorithm. In: Proceedings of 2002 U.K. Workshop on Computational Intelligence (UKCI-02), pp. 147–154 (2002)

9. Shafiq, M.Z., Tabish, S.M., Farooq, M.: On the appropriateness of evolutionary rule learning algorithms for malware detection. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, pp. 2609–2616. ACM (2009)
10. Rafique, M.Z., Chen, P., Huygens, C., Joosen, W.: Evolutionary algorithms for classification of malware families through different network behaviors. In: Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO). ACM Press (2014). To appear
11. Mehdi, S.B., Tanwani, A.K., Farooq, M.: IMAD: in-execution malware analysis and detection. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1553–1560. ACM (2009)
12. Edge, K.S., Lamont, G.B., Raines, R.A.: A retrovirus inspired algorithm for virus detection and optimization. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO 2006, New York, NY, USA, pp. 103–110. ACM Press (2006)
13. Loong, S.N.K., Mishra, S.K.: De Novo SVM classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures. Bioinform. Comput. Appl. Biosci. **23**, 1321–1330 (2007)

Multi-centroid Cluster Analysis in Malware Research

Ciprian Oprişa[1,2(✉)], George Cabău[1,2], and Gheorghe Sebestyen Pal[2]

[1] Bitdefender, 1, Cuza Vodă Street, City Business Center, 400107 Cluj-Napoca, Romania
{coprisa,gcabau}@bitdefender.com
[2] Technical University of Cluj-Napoca, 28, Gh. Bariţiu Street, Room M01A, 400027 Cluj-Napoca, Romania
gheorghe.sebestyen@cs.utcluj.ro

Abstract. Verdicts assignment is a recurring problem in malware research and it involves deciding if a given program is clean or infected (if it contains malicious logic). Since the general problem of identifying malicious logic is undecidable, a certain amount of manual analysis is required. As the collections of both clean and malicious samples are continuously increasing, we would like to reduce the manual work to a minimum, by using information extracted by automated analysis systems and the similarity between some programs in the collection.

Based on the assumption that similar programs are likely to share the same verdict, we have designed a system that selects a subset from a given collection of program samples for manual analysis. The selected subset should be as small as possible, given the constraint that the other verdicts must be inferable from the manually-assigned ones. The system was tested on a collection of more than 200000 clusters built using the single linkage approach on a collection of over 20 million samples.

Keywords: Malware · Clustering · Graph analysis · Dominating set

1 Introduction

Malware research involves working with large collections of potentially malicious samples. To decide whether a given sample contains malicious logic is a difficult task, even for a human [4]. For instance, a sequence of instructions that encrypt the user's personal files might be contained in a benign program that helps the user to protect his privacy. The same sequence of instructions could be used to encrypt the user's personal files in order to demand a ransom for restoring them. The only difference is that in the first case the user desires the effect of the program (i.e. the encryption of the files) while in the second case the user is rather tricked into running the malware.

The difficulty of malicious logic detection was shown more formally by Cohen. His paper [5] focuses on a particular type of malicious logic, called computer virus

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_7

(formally defined as a sequence of symbols that replicate themselves in a Turing machine). The paper states that "it is undecidable whether an arbitrary program contains a computer virus". The claim is proved by reducing the virus detection to the halting problem [17].

Although methods for detecting whether a program is malicious or not exist (most of them involve running the sample in a controlled environment to detect malicious actions [6]), they cannot fully substitute manual analysis performed by human researchers. Indeed, if known malicious actions are observed, the program can be classified as malware, but if nothing is observed, no claim can be made. We notice here the similarity with the halting problem [17]: by executing an algorithm, we can observe if it halts after a certain amount of time. However, if it still runs, we cannot draw any conclusion.

In this paper, we will deal with the fact that manual analysis can't be fully substituted but we will try to reduce it as much as possible. By studying a large collection of programs from the last 3 years ($\sim 10^8$ samples), both clean and malicious we have observed that there are some similarities between them. Different versions of the same application or different generations of the same malware family will share similar code.

The following section will present some related work, highlighting the advancements brought by the authors work. The third section will describe the process of programs clustering and argue about the verdicts assignment issue. The problems will be addressed in the next section, where we will show how to select meaningful centroids from a cluster and use them to automatically assign verdicts to every other sample. The algorithm was tested on a real-world collection of samples and the experimental results are shown in the 5th section. The final section draws the conclusions and provides some ideas for future work.

2 Related Work

The problem of clustering a large collection of binary programs in order to ease their analysis have been previously studied. The preferred features for clustering are OpCode n-grams, which are sequences of consecutive operation codes, extracted from the disassembled code.

Bilar [3] has studied the OpCodes extracted from both clean and malicious samples and run a frequency analysis on them. His analysis showed that OpCodes can help to discriminate between different program classes and successfully identify malware samples. Raw n-grams, as simple sequences of consecutive bytes showed promising results in [1], as the authors showed that similar programs have many common n-grams. Further refinements were shown in [15], where OpCodes and n-grams were combined in order to build powerful classifiers.

Clustering techniques can also be applied on non-executable malware, like PDF files. The authors of [18] worked on a large corpus of documents. First, they separated the samples collection in similarity classes then clustered using different methods.

Features extracted by dynamic analysis can also work with clustering algorithms. The authors of [2] built behavioral profiles and found connections

between malware samples with different appearances. Their method proved to be scalable and managed to cluster 75000 samples in less than three hours.

The current work takes a step further to the automated malware analysis process by analyzing existing clusters and selecting a representative subset that is worth further analysis.

3 Clustering and Verdicts

Previous work has shown how to compute the distance between two programs [12], based on features extracted from a program's code and how to perform cluster analysis [13], even on large collections.

In what follows, we will denote by $\mathcal{S}$, the set of all feature sets extracted from the sample programs. The recurring issue in malware analysis is the verdicts assignments. Each sample should be given a verdict, that can be *clean* or $infected$, as in Eq. 1.

$$verdict : \mathcal{S} \rightarrow \{clean, infected\} \tag{1}$$

A distance function d, as in Eq. 2 will compute the dissimilarity between two samples.

$$d : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1] \tag{2}$$

For instance, the extracted features can be a set of n-grams, as in [13]. In this case the Jaccard distance can be used ($d_J(S_1, S_2) = 1 - \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$). However, the reasoning will be similar for any metric distance defined on $\mathcal{S}$ that takes values between 0 and 1.

For such a metric d, we will make the following assumption: for two samples $A, B \in \mathcal{S}$ the probability they will share the same verdict is:

$$P(verdict(A) = verdict(B)) = 1 - \frac{d(A, B)}{2} \tag{3}$$

Basically, Eq. 3 states that two samples that are very similar (the distance between them is small) are likely to share the same verdict. However, if the samples are completely dissimilar ($d(A, B) = 1$), no assumption can be made (the probability will be 0.5).

The first step for minimizing the manual work of assigning verdicts to each sample is to cluster them. We will use the single linkage approach [16], in order to ensure that each pair of samples that are similar enough will end up in the same cluster. More formally, for a given distance threshold θ, $\forall A, B \in \mathcal{S}$ such that $d(A, B) \leq \theta$, we will have $cluster(A) = cluster(B)$. For dealing with a large collection of samples, we will use the locality-sensitive hashing approach from [13] that gives a very good approximation of the clusters.

Each cluster can be interpreted as a connected graph, $G = (V, E)$, where $V \subset \mathcal{S}$ is the set of all features sets associated to the samples in that cluster. Two nodes are connected by an edge if their distance is smaller than the threshold

θ: $E = \{(u, v) \in V \times V \mid d(u, v) \leq \theta\}$. Also, each edge will be weighted. The weight function is the distance between the two nodes, as in Eq. 4.

$$\begin{aligned} w &: E \to [0, 1] \\ w((u, v)) &= d(u, v) \end{aligned} \tag{4}$$

The biggest issue with the single linkage approach is the *chaining effect*. Sample A can be similar with B and B can be similar with C, while A and C are not necessarily similar. The issue is illustrated in Fig. 1, where we have used simple lines to represent distances smaller than the threshold θ and dashed lines for larger distances.

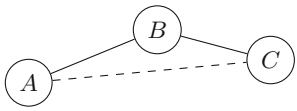


Fig. 1. Chaining effect illustrated

The single linkage approach would place the nodes A, B and C from Fig. 1 in the same cluster. To minimize the amount of work required to assign verdicts for all samples, a researcher can be asked to assign verdicts only for some of the elements of a cluster. Then, by association, the verdict can be extended to the neighboring nodes in the graph. Because of the chaining effect, we will only allow verdict extension to the neighboring nodes. For instance, if we have a verdict given by a human for the node A, we can expand the verdict to the node B, but we won't expand it further from B to C, so we will have to request further investigations for sample C. If we asked the human to analyze the sample B, the verdict can be directly extended to both nodes A and C, reducing the human work to a single analysis.

Another issue may arise for the cluster in Fig. 1 if A and C will be given different verdicts (e.g. A is declared clean and B infected). In this case, further human investigation is necessary in order to establish the correct verdict for B. However, if A and C were given the same verdict, the verdict for B could be automatically inferred with a high degree of confidence.

By generalizing the above issues to any graph, we can define the two main problems that the next section will try to solve:

- Given a cluster of possibly malicious programs, select a subset of samples to be manually analyzed so that the rest of verdicts can be inferred by neighborhood associations.
- For a cluster where some of the samples already have verdicts (given by humans or automated systems), determine if the rest of the verdicts can be automatically inferred or there are conflicts that need to be addressed manually.

4 Multi-centroid Selection Algorithm

Several clustering algorithms use the concept of *centroid*. For instance, the k-means algorithm [11] determine a set of centroids, then groups the remaining points around them, by proximity. Such centroids can be used as representative points for the cluster. However, hierarchical clustering methods, such as single linkage do not use centroids.

In order to manually analyze the most relevant samples, we would like to determine one or more centroids for every cluster. The rule that we have established in the first section, to infer verdicts only for the direct neighbors of the analyzed nodes will require a subset of samples, such that each sample in a cluster is either a centroid or is adjacent to one.

This problem of finding such a set of centroids is called in literature the *minimum dominating set* and has been proved to be NP-hard [8]. It can be linearly reduced to the *set cover problem* [10] which already has a $(1+\log|V|)$-approximation [9]. The problem was also proved not be $(1-\epsilon)\log|V|$-approximable for any $\epsilon > 0$ [7]. An exact algorithm has been published by van Rooij and have a complexity of $O(1.5048^n)$ [14].

The $(1+\log|V|)$-approximation algorithm was adapted from [9] to find a set of centroids for a given graph in Algorithm 1.

Algorithm 1. FIND-MULTIPLE-CENTROIDS(G)

Require: A cluster of samples, represented as a graph $G = (V, E)$
Ensure: The set of centroids, $C \subset V$

1: $C \leftarrow \emptyset$
2: $NotReached \leftarrow V$
3: **for all** $v \in V$ **do**
4: $\quad Neighb[v] \leftarrow \{v\} \cup \{u \in V \mid (v, u) \in E\}$
5: **end for**
6: **while** $NotReached \neq \emptyset$ **do**
7: $\quad c \leftarrow \underset{v \in NotReached}{\arg\max} |Neighb[v]|$
8: $\quad C \leftarrow C \cup \{c\}$
9: $\quad crtReach \leftarrow Neighb[c]$
10: $\quad NotReached \leftarrow NotReached \setminus crtReach$
11: $\quad$ **for all** $v \in crtReach$ **do**
12: $\quad\quad Neighb[v] \leftarrow Neighb[v] \setminus crtReach$
13: $\quad$ **end for**
14: **end while**
15: **return** C

The algorithm starts by initializing the set of centroids with the empty set (line 1) and the set of nodes that have not been reached yet with V (line 2). For each node, we will build a neighborhood set, comprised of itself and all the other nodes directly reachable from it (line 4).

While the set of nodes that weren't reached yet is not empty, the node with the highest neighborhood is selected as a new centroid (lines 7–8). Its entire neighborhood is then eliminated from the list of nodes not reached yet (line 10) and from the neighborhoods of its neighbors (line 12).

The greedy part of the algorithm is the selection of the next centroid (line 7) as the node with the highest neighborhood, considering only the nodes not yet reached. This heuristic tries to eliminate as many nodes as possible from the $NotReached$ set at each step. The smaller number of steps, the smaller the number of centroids for that cluster will be.

For each iteration of the algorithm, at least one vertex is eliminated from the $NotReached$ set, so we have at most $|V|$ iterations. In each iteration, the vertex with the largest neighborhood is selected (line 7), also in $O(|V|)$ steps, so the complexity so far is $O(|V|^2)$. The subtraction of the currently reachable set from the neighborhoods (line 12) has a running time proportional with size of $crtReach$, assuming the sets are implemented as look-up tables. However, each node will only be eliminated once from the neighborhoods, so this operation also takes $O(|V|^2)$. We conclude that the running time of Algorithm 1 is quadratic in the cluster size.

After a set of centroids is selected and some verdicts are assigned to each of them, the verdicts can be extended to all the nodes in the graph. When all the verdicts for the centroids agree (all of the centroid samples are clean or all of them are infected), the problem is trivial, as every node in the graph will share the same verdict. If some centroids are given different verdicts, there will be at least one contradictory edge (the two ends of the edge will have different verdicts). Since the graph is connected, there will either be two centroids that are directly connected and have different verdicts, or there will be a non-centroid node adjacent to both a clean and an infected centroid.

The aforementioned contradictions may appear for two reasons:

- some verdicts were incorrectly assigned
- some nodes that are connected by an edge are in fact not similar

To discern between the two cases and to fix the graph verdicts, further human intervention is required. An analyst can identify incorrectly assigned verdicts and fix them or he can remove some edges of the graph in order to separate it into smaller connected components with uniform verdicts.

5 Experimental Results

We have tested the algorithm presented in this paper on a collection of more than one million clusters built using the single linkage approach from more than 20 million unique samples. The tests ran on a machine with Intel i7 vPro processor at 2 GHz with 8 GB of RAM.

For Algorithm 1 we are interested on how much the human effort was reduced, by computing the size of the centroids for different cluster sizes. Since the number of centroids is influenced by the cluster shape, not only by the cluster size, we

Table 1. Average number of centroids and running times for various cluster sizes

Cluster size	Avg. nr. centroids	Avg. running time (ms)
2–9	1.411	0.036
10–19	1.796	0.061
20–29	2.567	0.126
30–39	3.191	0.192
40–49	3.741	0.232
50–59	4.338	0.254
60–69	4.926	0.320
70–79	5.611	0.406
80–89	6.112	0.462
90–99	6.766	0.568
100–109	6.889	0.617
110–119	7.352	0.729
120–129	8.438	0.832
130–139	8.530	0.931
140–149	8.170	1.082
150–159	8.609	1.150
160–169	11.113	1.307
170–179	10.667	1.402
180–189	11.299	1.514
190–199	11.231	1.594
200–209	11.556	1.791
210–219	13.691	1.916
220–229	14.468	2.087
230–239	13.852	2.221
240–249	13.790	2.313

have split the clusters into groups of similar sizes and computed the average number of centroids for each group. Table 1 shows this number of centroids, along with the average running time for computing the centroids of a cluster.

The average number of centroids found by Algorithm 1 in a graph of size n is shown in Fig. 2. Although the trend line shows a linear dependency, the slope is only $5.37 \cdot 10^{-2}$ so the amount of samples in cluster that require manual analysis is greatly reduced.

Figure 3 confirms the theoretical analysis of the algorithm's complexity and shows its quadratic running time. For most clusters, finding the centroids requires less than two milliseconds.

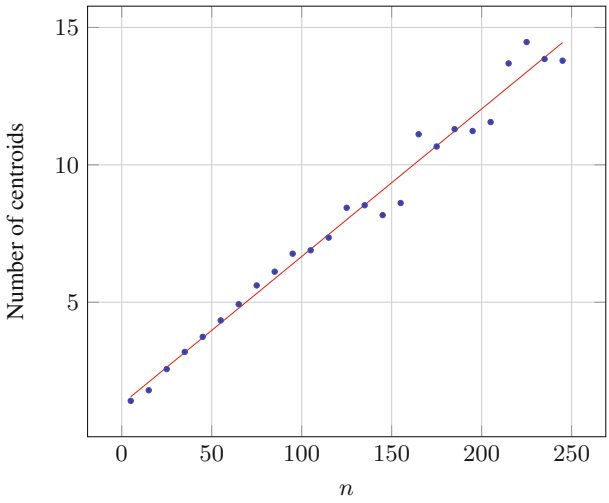


Fig. 2. Number of centroids found by Algorithm 1

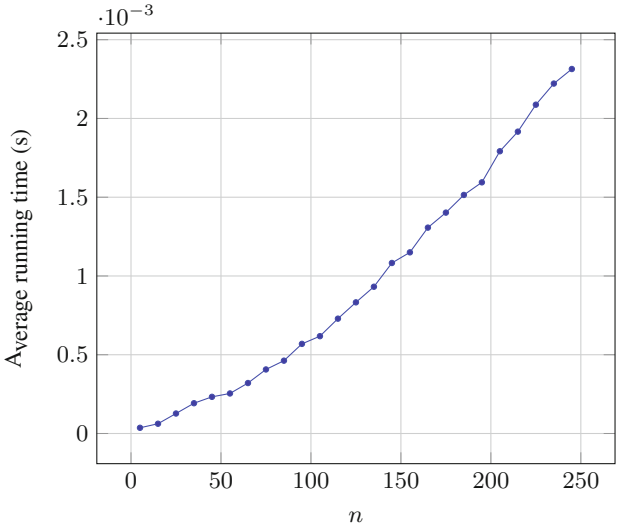


Fig. 3. Running time of Algorithm 1

6 Conclusions and Future Work

This paper showed some solutions for reducing the amount of manual work performed by malware analysts while classifying large collections of potentially malicious samples.

The issue was how to select a subset of representative samples from a cluster of similar ones, such that every other sample is similar with at least a sample

that was selected. The problem was reduced to the *minimum dominating set*, a known NP-hard problem that allows an approximation algorithm.

Experimentally, we have shown that the number of centroids is about 5% of the cluster size, so the manual analysis is reduced with almost 95%. The running time of the approximation algorithm is quadratic, but even a Python implementation renders very fast execution times (1 or 2 ms).

As future work, we will try to develop a method for automatically solving the conflicts between the manually or automatically assigned verdicts.

References

1. Abou-Assaleh, T., Cercone, N., Kešelj, V., Sweidan, R.: N-gram-based detection of new malicious code. In: Proceedings of the 28th Annual International Computer Software and Applications Conference, COMPSAC 2004, vol. 2, pp. 41–42. IEEE (2004)
2. Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable, behavior-based malware clustering. In: NDSS, vol. 9, pp. 8–11. Citeseer (2009)
3. Bilar, D.: Opcodes as predictor for malware. Int. J. Electr. Secur. Digit. Forensics **1**(2), 156–168 (2007)
4. Bishop, M.: Computer Security: Art and Science. Addison-Wesley, Reading (2002)
5. Cohen, F.: Computational aspects of computer viruses. Comput. Secur. **8**(4), 297–298 (1989)
6. Colesa, A.: Fast creation of short-living virtual machines using copy-on-write ramdisks. In: 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, pp. 1–6. IEEE (2014)
7. Feige, U.: A threshold of ln n for approximating set cover. J. ACM (JACM) **45**(4), 634–652 (1998)
8. Hedetniemi, S.T., Laskar, R.C.: Bibliography on domination in graphs and some basic definitions of domination parameters. Discrete Math. **86**(1), 257–277 (1990)
9. Johnson, D.S.: Approximation algorithms for combinatorial problems. In: Proceedings of the Fifth Annual ACM Symposium on Theory of Computing, pp. 38–49. ACM (1973)
10. Kann, V.: On the approximability of NP-complete optimization problems. Ph.d. thesis, Royal Institute of Technology Stockholm (1992)
11. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, California, USA, pp. 281–297 (1967)
12. Oprisa, C., Cabau, G., Colesa, A.: From plagiarism to malware detection. In: 2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp. 227–234. IEEE (2013)
13. Oprisa, C., Checiches, M., Nandrean, A.: Locality-sensitive hashing optimizations for fast malware clustering. In: 2014 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 97–104. IEEE (2014)
14. van Rooij, J.M., Nederlof, J., van Dijk, T.C.: Inclusion/exclusion meets measure and conquer. In: Algorithms-ESA 2009, pp. 554–565. Springer (2009)
15. Shabtai, A., Moskovitch, R., Feher, C., Dolev, S., Elovici, Y.: Detecting unknown malicious code by applying classification techniques on opcode patterns. Secur. Inf. **1**(1), 1–22 (2012)

16. Sibson, R.: Slink: an optimally efficient algorithm for the single-link cluster method. Comput. J. **16**(1), 30–34 (1973)
17. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. J. Math. **58**(345–363), 5 (1936)
18. Vatamanu, C., Gavriluţ, D., Benchea, R.: A practical approach on clustering malicious pdf documents. J. Comput. Virol. **8**(4), 151–163 (2012)

Computational Game Theory

Cooperation in Multicriteria Repeated Games

Réka Nagy(✉), Mihai Suciu, and Dan Dumitrescu

Babes-Bolyai University, Cluj-Napoca, Romania
reka@cs.ubbcluj.ro

Abstract. In classical Game Theory a rational player's goal is to maximize its payoff by choosing a strategy that is best response to the opponent's strategy. This theoretical presumption leads to mutual defection in dilemma games. Despite theoretical prediction in real life situations players tend to cooperate. Our goal is to develop a model that overcomes some of the limitations of classical models. We investigate the emergence of cooperation for the Prisoner's Dilemma game in a spatial framework with multicriteria payoffs. We propose a multicriteria model where a second criterion, that reflects the identity of a player, is introduced. Numerical experiments show that the second criterion promotes cooperation without any external interactions. The proposed model allows the interaction of different type of players which leads to more realistic outcomes.

1 Introduction

In classical Game Theory (GT) choices made by rational and conscious individuals determine the outcome of a game. A player maximizes its payoff by choosing a strategy that is best response to the one chosen by its opponent. Thus players have no reason to change their chosen strategies. Real-life interactions between players who belong to the same group are better modeled by repeated games. Through repeated interactions one can learn from its opponents, the actions of a player may influence the actions of the group and conversely.

Since its appearance the Prisoner's Dilemma game (PD) has raised many open question in Game Theory, most of them related to the emergence of cooperation. The repeated version of the game, Iterated Prisoner's Dilemma (IPD), has been used as a model for many real world phenomena in areas such as social dilemmas [1], evolutionary biology [2], governance of commons [3,4]. Despite the vast research in the area building more accurate models is still a challenge.

It is known, that even though cooperation is in the common interest of a population and in the long term gives the best gains, self interest demands defection. Standard GT models show that the outcome of the IPD game is a population where everyone defects, which leads to the worst possible outcome. However, despite theoretical predictions in real world situations there exists cooperation.

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_8

Based on theoretical models the rational choice is defection, but real players have a willingness to cooperate [5–7].

Cooperation in the IPD game has been widely studied [1,8,9]. Several additional mechanisms like punishments [2,6,10], group selection [11], reciprocal altruism, etc. have been introduced in order to promote cooperation.

In our approach, we consider that standard Game Theory models have some limitations. The standard IPD models are overly simplified and consider that players are fully rational agents whose only goal is to maximize their own payoffs. The fact that real players have a tendency for cooperation is neglected. Real players are not always rational and they often make their choices based on more than one criteria. Moreover, it is unrealistic to assume that all players are uniform.

Our goal is to develop a model that overcomes some of these limitations. We believe that by taking into account the nature of real players more realistic models can be built. We propose a model that besides the actual payoffs also takes into account the players' tendency to cooperation. We study the Prisoner's Dilemma within the context of multicriteria games [12] and we attempt to better model the irrational nature of real life agents. We aim to create a more realistic interaction model between real life agents.

2 Multicriteria Games - Games with Vector Payoffs

In real life situations players usually make decisions considering more than one, often conflicting, criteria. Most of the times, these criteria are not measured by the same unit, they can not be aggregated into one single criterion.

Multicriteria games (or games with vector payoffs) [12] are natural extensions of standard non-cooperative games and offer a more realistic model for real life situations.

A finite strategic multicriteria game is defined as a system

$$\Gamma = ((N, S_i, u_i), i = 1, n),$$

where:

- N represents a set of n players, $N = \{1, ..., n\}$;
- for each player $i \in N$, S_i represents the set of pure strategies available to its, $S_i = \{s_{i_1}, s_{i_2}, ..., s_{i_m}\}, S = S_1 \times S_2 \times ... \times S_n$ is the set of all possible strategy profiles;
- for each player $i \in N, u_i : S \rightarrow R^r$ represents the vector payoff function, where $r \in \mathbb{N}$ is the number of criteria.

We consider that each player is a maximizer.

Any multicriteria game G with r criteria is composed of r standard non-cooperative games: $G_1, ..., G_r$. If all players have only one criterion ($r = 1$) then we have a standard non-cooperative game.

The main solution concept in Game Theory is Nash equilibrium [13,14]. Nash equilibrium has been adapted to multicriteria games in various versions [12,15–18].

The most studied multicriteria equilibrium concept is the *Pareto-Nash equilibrium* introduced in [12]. The Pareto-Nash equilibrium concept is an extension of the Nash equilibrium for single-criterion games and is based on Pareto domination.

A strategy profile $s^* \in S$ is a Pareto-Nash equilibrium if and only if the following condition holds

$$u_i(s^*) \succ_P u_i(s_i, s^*_{-i}),\ \forall s_i \in S_i, \forall i \in N.$$

3 Modeling Identity in a Multicriteria Game Theory Framework

Multicriteria games are suitable for building more adequate game models. They capture in a realistic way the conflicting elements that contribute to the decision process. With the help of multicriteria games, beyond the actual gain, a player's tendency to cooperation can also be considered.

Standard single criterion games can be thus transformed into muticriteria games by extending the payoff function. We consider a game with two criteria: the first criterion is the standard payoff while the second criteria captures the player's inclination towards cooperation.

Multicriteria PD. One of the most relevant questions regarding real agents is whether they are willing to act in a way that is favorable for the community or they act only to maximize their own gain. All players benefit by mutual cooperation, but in the same time the temptation to defect against a cooperator is high. However fair play suggests the best strategy would be to cooperate.

Real life agents do not make their decision based only on actual payoffs. Beyond the actual payoffs their identity determines their choices. For example an agent with a cooperator identity is less likely to defect even though rational thinking implies defection regardless of the strategy of the opponent.

The payoff matrix for the generalized standard PD game is:

	C	D
C	(R, R)	(S, T)
D	(T, S)	(P, P)

For the game to be an actual dilemma the conditions $T > R > P > S$ and $2 \cdot R > T + S$ must hold.

A multicriteria PD extends the standard game by adding a second payoff that captures the players' tendency to play strategy C or D. We refer to this second criterion as an *identity payoff*. In case of a player who favors strategy C the identity-payoff for cooperating is considerably larger than for defecting. For the multicriteria version of the game we propose the following payoff matrix:

	C	D
C	$[(R,r);(R,r)]$	$[(S,s);(T,t)]$
D	$[(T,t);(S,s)]$	$[(P,p);(P,p)]$

The payoffs for the first criterion (R, S, T, and P) are the actual gains and are equal to the payoffs of the standard PD game. The payoffs for the second criterion (r, s, t, and p) correspond to the identity payoffs. In order to capture a cooperative identity, we propose the following relation between the values of the identity payoff: $r > s > p > t$.

While actual payoffs suggest defection, the identity payoffs are higher for cooperation. So even if cooperation does not necessarily imply high material benefits, it offers a moral gain. The actual payoff is the highest when defecting against a cooperator, i.e. playing strategy D against strategy C. However the identity payoff in this situation is very low.

Since the first criteria suggests defection and the last criteria suggests cooperation, the multicriteria PD has no multicriteria Nash equilibrium.

3.1 Multicriteria IPD

When the Prisoner's Dilemma is played only once, cooperation between players is not likely to happen. Considering only the actual payoffs (i.e. considering only the material gain), no matter what the opponent does, a player can win more by playing strategy D. However, if the game is repeated mutual cooperation seems a rational choice.

The Iterated Prisoner's Dilemma [8,19,20] is a spatial model where each player is regarded as an element of a matrix. Initially, the players choose randomly between cooperation or defection. According to the standard IPD, the game is repeated for several rounds; in each round a player plays the Prisoner's Dilemma with all its neighbors, including herself. The payoff of each player is equal to the total payoff earned in each of these games. At the end of each round the players imitate their most successful neighbor. The dominant strategy in these settings is D. If the temptation to defect is high enough, defectors invade the whole space [1,8].

Cooperation is essential for evolution [20]. Several rules that punish defectors and reward cooperators have been introduced in order to achieve cooperation [6,10,11,21].

Punishing defectors, localized interactions between cooperators and relying on reputation promotes cooperation in the population [2,10,22,23]. In case of the public good game, punishment leads to cooperation if non-cooperative agents are punished by reducing their payoff [6].

Group selection plays an important role in promoting a cooperative behavior [11]. If no punishment is present in the group then altruistic cooperation decreases, but when altruistic punishers are present agents with a defector behavior are excluded. Agents that reinforce a cooperative behavior are not removed from the population during the game.

We propose the use of multicriteria games to simulate more accurately real life interactions. We use no auxiliary techniques to promote cooperation. The base of the proposed model is the multicriteria PD with identity payoffs that favor cooperation. We use a lattice to model players interactions; it provides a simple topology which permits the investigation of the effect of the second criterion.

An $n \times n$ square lattice is considered, thus the number of players is n^2. The game is repeated for K rounds. In each round, each player plays with all its neighbors (in a spatial framework based on a lattice there are 8 neighbors - considering all nodes that surround a player). For the sake of simplicity, the most widely studied update rule is adopted [19]. At the end of each round t each player i changes its strategy according to the following rules:

- the cumulative payoff for each criteria (the total payoffs for all the 8 games played in round t) is computed for all players as follows: $(u_i^1, u_i^2) = (\sum_{j=1}^{8} u_j^1, \sum_{j=1}^{8} u_j^2)$;
- the cumulative payoff of each player is compared to the payoffs of neighbors;
- if there is a neighbor whose cumulative vector payoff Pareto dominates the payoffs of player i, than player i adopts the strategy of that neighbor;
- in case there are more than one neighbors with better payoffs one of them is chosen randomly.

In other words each player imitates the most successful neighbor with respect to Pareto domination.

Modeling different identities. In standard GT models, all players are uniform and are guided by the same goal. The interaction of different types of players is not possible. In order to achieve this, we weight the second criterion of the payoff function with parameter $\lambda \in [0, 1]$. By altering the value of the parameter λ we can model players with different tendencies to cooperation.

The parameter λ permits the modeling of players having different inclinations towards cooperation. Introducing a parameter to the second criterion allows us to simulate a heterogeneous population.

4 Numerical Experiments

We study the effect of the second criterion on the emergence of cooperation. By introducing an identity criterion we hope to better model real life interactions and promote cooperation without any external interactions.

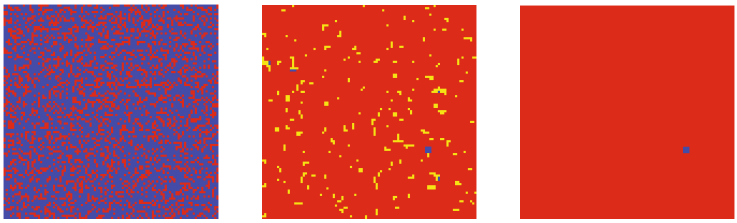
Experimental setup. Trough numerical experiments we observe the emergence of cooperation in different environments:

1. In order to set a baseline we study the classical IPD [19]. It is known that this yields to a defecting population.
2. Next we study a homogeneous environment (see Experiment 2), all players have the same identity and the same payoffs. Our goal is to study whether by introducing an identity payoff leads us to a population that plays strategy C.

3. We attempt to build a more realistic model by emulating a heterogeneous environment (see Experiment 3). Two type of players are considered having different, more or less cooperative, identities.

For our experiments we consider a 100×100 matrix (10000 players). The game is repeated for 150 rounds. Presented results represent the average over 30 independent runs.

Experiment 1 - Standard IPD. Values for the standard Prisoners Dilemma game are: $R = 3, T = 5, S = 0$, and $P = 1$.



(a) Round 1 (b) Round 2 (c) Round 5

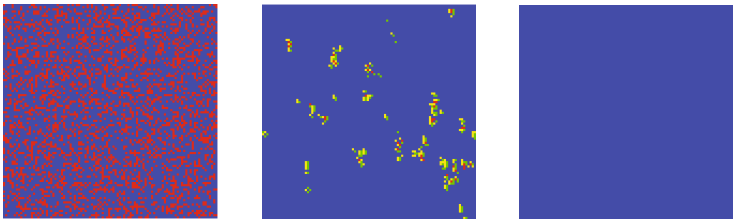
Fig. 1. IPD - The population in different rounds. In just a few rounds strategy D is adopted by the all players. (blue - players that choose strategy C, red - players with strategy D, yellow indicates a player that has just switched from C to D, and green - players that switch from D to C). (Color figure online)

Figure 1 depicts the population for the IPD in different rounds of the game. The initial population (Fig. 1a) is randomly initialized, 30% of players will play strategy D. However, in round 2 the majority of the population defects, while in round 5 a final state is reached, when there is a small island of cooperators and the rest of the population defects. By round 150 the whole population adopts strategy D.

As Fig. 1 shows the standard IPD does not promote cooperation. If one tries to promote cooperation in this environment additional techniques must be considered.

Experiment 2 - Multicriteria PD. We study the evolution of cooperation in case of a multicriteria IPD game. Beyond the standard payoff, a second criterion is added to the payoff function. This criterion captures the player's identity or its tendency for cooperation. In case of the multicriteria version of the game for the first criterion we keep the values as in the standard game and for the second criterion we use the following setting: $r = 3, t = 0, s = 3$, and $p = 1$. For the payoff matrix considered the second criterion favors cooperation.

The outcome of spatial evolutionary games, if no external rules are defined, is the Nash equilibrium. Figure 2 depicts the evolution of cooperation for different rounds of the game. The percentage of defectors in the first round is 30%, and defectors are randomly distributed. In round 25 the ratio of defectors is



(a) Round 1 (b) Round 25 (c) Round 150

Fig. 2. Experiment 2. - Iterated Multicriteria PD. The population in different rounds. After each round there are less and less players that choose strategy D, by round *150* every player cooperates.

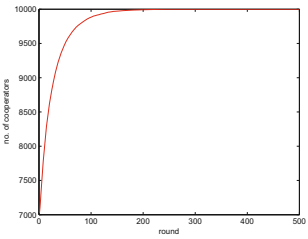


Fig. 3. The number of cooperators in each round.

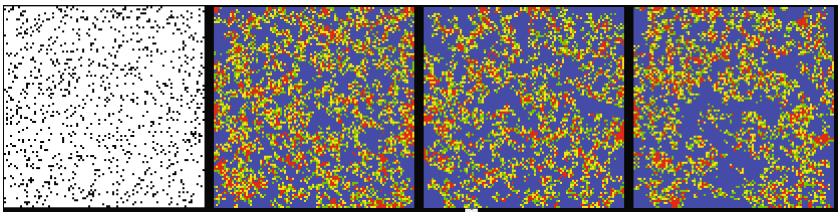
reduced to 15% and defectors are grouped together in small islands. The islands of defectors persist through the rounds. After round 50 these islands get smaller and smaller and eventually, after round 150, all players adopt strategy C.

Figure 3 represents the number of cooperators for each round. At the beginning the number of defectors is relatively big, but it slowly decreases in each round, and by round 150 100% of the population cooperates.

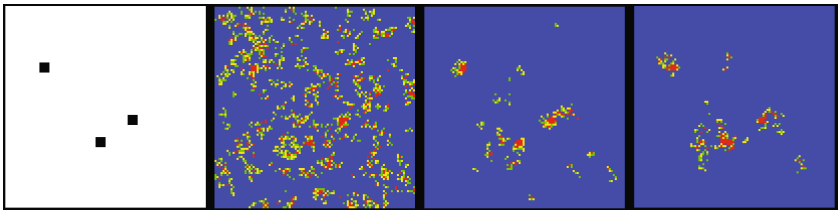
Numerical experiments show that the outcome of the game is not influenced by the distribution of the initial strategies. Regardless of the initial distribution, eventually all players end up with strategy C.

Experiment 3 - Multicriteria PD - Different Types of Players. We experiment with a model where players are allowed to have different identities, they can be more or less cooperative. We model two types of players: players having a *cooperator identity* and players having a *defector identity*. Note that the identity of the player does not necessarily determine the chosen strategy. A player with a cooperator identity can easily choose strategy D and also, a player with a defector identity can adopt the strategy C.

Figure 4a explores the case when the percentage of the players having a cooperator identity ($\lambda = 1$) is 90% and the percentage of players that are initialized with strategy C is 50%, i.e. the population has a tendency for cooperation but initially only half of the players will play strategy C. Figure 5a shows the percentage of players that adopted strategy C for cooperative players ($\lambda = 1$) and



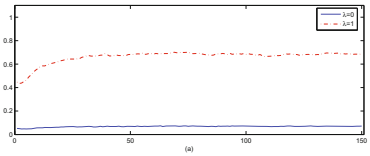
(a)



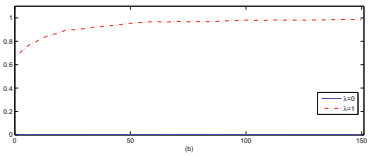
(b)

Fig. 4. Strategy evolution in Iterated Multicriteria Prisoner's Dilemma. First figures from the left presents the identity of the players, a black square corresponds to a player with a defector identity ($\lambda = 0$) and a white square corresponds to a player with a cooperator identity ($\lambda = 1$). The rest of the figures present each players strategy in rounds 25, 100, and 125.

players with defector identity ($\lambda = 0$, red line). We can observe that a cooperative identity promotes the spread of strategy C. The number of "defectors" that choose strategy C is relatively constant.



(a)



(b)

Fig. 5. Evolution of percentage of strategy C in each round for Fig. 4a and b.

Figure 4b illustrates the case when the majority of the population has a cooperator identity, and less that 1% have a defector identity (60 players) and they are grouped in three small islands. Cooperative players in the neighborhood of defectors might adopt the strategy D, but in contrary to the classical IPD, the strategy D does not spread trough the population. We can conclude that players stick to their identity.

4.1 Discussion

In the case of the multicriteria PD where the second criterion favors cooperation, in our experiments, Nash equilibrium is not reached. Thus no predictions can be made for the repeated scenario. We study the influence of the second criterion (called identity payoff) on the emergence of cooperation in IPD.

Three cases are investigated:

1. Classical IPD (one criterion): we can observe that players adopt the strategy D (Fig. 1).
2. Multicriteria IPD: a second criterion is introduced, thus there are two conflicting criteria. Numerical experiments show that cooperation emerges regardless the initial strategy distribution.
3. Multicriteria IPD - players with different identities ($\lambda \in \{0, 1\}$): in a situation where multiple identities are present we can observe that in most cases players choose a strategy that corresponds to their identity.

Based on our experiments we can conclude, that the introduction of identity payoffs helps the emergence of cooperation. When all players have the same (cooperating) identity no additional techniques or rules need to be defined in order to reach a cooperating population. In case there are two types of players a more realistic outcome is obtained: a population where some cooperate and some defect. The identity of the players has an effect on the adopted strategy. Players with a cooperative identity choose defection only when they have players with defector identity in their neighborhood. When players with cooperative identity are present in the population the strategy D is not adopted by the entire population.

5 Summary and Conclusions

The Iterated Prisoner's Dilemma is used to model a wide range of real life situations from biological evolution to social interactions. Classical Game Theory predicts that the outcome of the game is defection. However the emergence of cooperation can be observed in many naturally occurring PD situations.

Multicriteria games offer an ideal framework for the study of IPD and up to our knowledge they have not been used for model IPD models. Multicriteria games allow the development of more realistic models, where the decision making is based on more than one criteria. The fact that real players have an inclination towards cooperation, is neglected in standard models. Thus adding a second

criterion that models the identity of a player to the standard game is a natural choice. This way basic single criterion cooperation dilemmas are extended to multicriteria games.

We study the emergence of cooperation in multicriteria IPD. In contrary to the standard payoff, the identity payoff favors cooperation. Based on our numerical experiments we can conclude, that the introduction of identity payoffs helps the emergence of cooperation; cooperation is reached solely based on the identity criterion.

In real life situations players involved in a game are rarely uniform. Usually real life interactions involve players that think differently and have different goals. We propose a model that encompasses players with different identities. Based on our experiments we can conclude that player will choose the strategy that best reflects their identity - which is a more realistic outcome.

References

1. Nowak, M.A., May, R.M.: The spatial dilemmas of evolution. Int. J. Bifurcation Chaos (IJBC) **3**(1), 35–78 (1993)
2. Nowak, M.A.: Five rules for the evolution of cooperation. Science **314**(5805), 1560–1563 (2006)
3. Ostrom, E.: Governing the Commons-The Evolution of Institutions for Collective Actions. Political Economy of Institutions and Decisions. Cambridge University Press, Cambridge (1990)
4. Ostrom, E., Gardner, R., Walker, J.: Rules, Games, and Common-Pool Resources. University of Michigan Press, Ann Arbor (1994)
5. Fehr, E., Gächter, S.: Cooperation and punishment in public goods experiments. Am. Econ. Rev. **90**(4), 980–994 (2000)
6. Fehr, E., Gächter, S.: Altruistic punishment in humans. Nature **415**(6868), 137–140 (2002)
7. Wedekind, C., Milinski, M.: Human cooperation in the simultaneous and the alternating prisoner's dilemma: Pavlov versus generous tit-for-tat. Proc. Nat. Acad. Sci. **93**(7), 2686–2689 (1996)
8. Nowak, M.A., May, R.M.: Evolutionary games and spatial chaos. Nature **359**, 826–829 (1992)
9. Szabó, G., Fáth, G.: Evolutionary games on graphs. Phys. Rep. **446**(4–6), 97–216 (2007)
10. Brandt, H., Hauert, C., Sigmund, K.: Punishment and reputation in spatial public goods games. Proc. R. Soc. London Ser. B Biol. Sci. **270**(1519), 1099–1104 (2003)
11. Boyd, R., Gintis, H., Bowles, S., Richerson, P.J.: The evolution of altruistic punishment. Proc. Nat. Acad. Sci. **100**(6), 3531–3535 (2003)
12. Shapley, L.S., Rigby, F.D.: Equilibrium points in games with Vector payoffs. Naval Res. Logistics Q. **6**(1), 57–61 (1959)
13. Nash, J.: The bargaining problem. Econometrica **18**(2), 155–162 (1950)
14. Nash, J.: Non-cooperative games. Ann. Math. **54**(2), 286–295 (1951)
15. Borm, P., Tijs, S., van den Aarssen, J.: Pareto Equilibria in Multiobjective Games. Technical report, Tilburg University (1988)
16. Zhao, J.: The equilibria of a multiple objective game. Int. J. Game Theory **20**, 171–182 (1991)

17. Wang, S.Y.: Existence of a Pareto equilibrium. J. Optim. Theory Appl. **79**, 373–384 (1993)
18. Borm, P., van Megen, F., Tijs, S.: A perfectness concept for multicriteria games. Math. Meth. Oper. Res. **49**, 401–412 (1999)
19. Axelrod, R., Hamilton, W.: The evolution of cooperation. Science **211**(4489), 1390–1396 (1981)
20. Nowak, M.A., Tarnita, C.E., Antal, T.: Evolutionary dynamics in structured populations. Philos. Trans. R. Soc. B Biol. Sci. **365**(1537), 19–30 (2010)
21. West, S.A., Griffin, A.S., Gardner, A.: Evolutionary explanations for cooperation. Current Biol. **17**(16), R661–R672 (2007)
22. Jacquet, J., Hauert, C., Traulsen, A., Milinski, M.: Shame and honour drive cooperation. Biol. Lett. **7**(6), 899–901 (2011)
23. Sasaki, T., Uchida, S.: The evolution of cooperation by social exclusion. Proc. R. Soc. B Biol. Sci. **280**(1752), 20122498 (2013)

Evolving Game Strategies in a Dynamic Cournot Oligopoly Setting

Mihai Alexandru Suciu(✉), Rodica-Ioana Lung, Noémi Gaskó, Tudor-Dan Mihoc, and Dan Dumitrescu

Centre for the Study of Complexity, Babeş-Bolyai University, Cluj Napoca, Romania
mihai-suciu@ubbcluj.ro
http://csc.centre.ubbcluj.ro

Abstract. A Cournot oligopoly is used as a benchmark for Nash equilibria tracking and detection in a dynamic setting. Several dynamics that induce different trajectories for the equilibria are considered: random, linear, cosine, and spiral. A new Extremal Optimization based method is tested on the proposed dynamic setting.

1 Introduction

Ever since it was proposed by Antoine Augustin Cournot [3] in 1838 the oligopoly model was extensively studied and used for education, research and applications.

With the emergence of dynamic games several models have been studied in order to develop more useful computational solutions. A large range of modifications of the classical Cournot game and their study methodologies have been applied, but ultimately the results are not satisfactory.

The instability of Nash equilibria for nonlinear discrete-time Cournot duopoly games where players have heterogeneous behaviors was investigated by Agiza and Elsadany in 2004 [1]. As expected the dynamics of their model becomes complex, chaotic and highly unpredictable. In order to improve the stability of equilibrium for this game a research has been conducted by Rong and Chen [6]. Their main goal was to find solutions for equilibria's sensitive dependence on initial conditions. Another study for Nash equilibrium in dynamic games lead Karafyllis et al. [7] to provide sufficient conditions for a robust global asymptotic stability. However these models are have strong mathematical constrictions making them difficult to be applied for concrete problems.

Binschi proposed a nonlinear discrete time Cournot duopoly game, where players have adaptive expectations [2] and faced the same problem: instability - the long run solution strongly depends on the initial conditions. In his paper he proposed a method based on the method of critical curves to study the global bifurcations that conduct to basins of attraction for equilibria as the model parameters vary.

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_9

A more practical approach was the attempt to model a competitive dynamic electricity market similar to the one from US developed by Kian [8]. Highlighting the benefices of using a dynamic game versus a static one is probably the main result, and one of the its drawbacks of is the fact that was not compared with a real market (even a simplified one). Greenfield and Kwoka developed also in [5] an oligopoly model to investigate investment and production of electricity in a market where demand evolves over time, and the two players are completely independent.

Another practical problem was analyzed by Qiao Ru Li et al. in [9]. Using evolutionary computation techniques they solved a dynamic mixed behavior traffic network model. Evolutionary methods proved also in [14] to be reliable techniques in approximating game equilibria in dynamic games.

In this paper a new approach to dynamic games and equilibria in a dynamic setting is proposed. The static Cournot oligopoly model is used as the bases for a dynamic model to be used as a benchmark for computational intelligence methods that aim to approach dynamic games. The dynamic Cournot oligopoly is appropriate for this task as it can be scaled to any number of players and the exact position of the Nash equilibrium can be computed at any moment analytically.

Following the construction of the benchmark a new algorithm for computing and tracking equilibria in large games is proposed, the Dynamic Nash Extremal Optimization (DNEO) and its performance is compared to that of the Dynamic Equilibria Tracking Differential Evolution in [15].

2 Some Basic Notions

A non cooperative game $\Gamma = (N, S, U)$ is defined by a set of players N, a set of actions S available to them and a set of payoff functions U:

- N is the set of players, $N = \{1, 2, \ldots, n\}$, n is the number of players;
- for each player $i \in N$, S_i is the set of actions available to him and

$$S = S_1 \times S_2 \times ... \times S_n$$

  is the set of all possible situations of the game. An element $s \in S$ is called a strategy profile of the game;
- for each player $i \in N$, $u_i : S \to \mathbb{R}$ represents the payoff function of i, and $U = (u_1, u_2, \ldots, u_n)$

The most popular solution concept in non-cooperative Game Theory is the Nash equilibrium [13], which is a strategy profile such that no player can increase its payoff by unilaterally deviating. Formally, a Nash equilibrium is a strategy profile $s^* \in S$ such that the inequality:

$$u_i(s_i, s^*_{-i}) \leq u_i(s^*), \forall i = 1, .., n, \forall s_i \in S_i,$$

holds, where (s_i, s^*_{-i}) denotes the strategy profile obtained from s^* by replacing the strategy of player i with s_i.

Let s and s^* be two strategy profiles; $k(s^*, s)$ denotes the number of players which benefit by deviating from s^* to s [4]:

$$k(s^*, s) = card\{i \in N, u_i(s_i, s^*_{-i}) > u_i(s^*), s_i \neq s^*_i\}.$$

We say that strategy s^* is better than strategy s with respect to Nash equilibrium, and we write $s^* \prec_N s$, if the inequality:

$$k(s^*, s) < k(s, s^*)$$

holds. $k(s^*, s)$ is a relative quality measure of s and s^* with respect to the Nash equilibrium. The relation $\prec_N$ is considered as a *generative relation of Nash equilibrium*, i.e. that the set of non-dominated strategies with respect to $\prec_N$ induces the *Nash equilibrium* [11].

A more realistic approach is to consider that some aspects of the game change in time. We can consider changes in the set of players (some may leave the game, others may enter), in the set of strategy profiles, or changes in the form of the payoff functions. In the following we will consider the latter and write $\Gamma(t) = (N, S, U(t), t \in \mathbb{N})$, where $u_i(s, t) : S \times \mathbb{N} \to \mathbb{R}$, $i = 1, \ldots, n$ form $U(t)$.

In this way we obtain a dynamic game [10] in which the state of the game from a moment t to moment $t + 1$ may or may not depend on the choices of players at moments $0, \ldots, t$.

The Nash equilibrium concept does not extend to dynamic games directly. Several solution concepts maintain the Nash equilibrium unilateral deviation philosophy, but implement it differently depending on the type of game and dynamic approach considered. Examples are the Open Loop Nash Equilibrium (OLNE) and the Markov Perfect Nash Equilibrium (MPNE), The OLNE computes the NE for each instance of the game moving from t to $t+1$ until the end of the game. The MPNE starts with the last epoch and moves toward the first one [10].

In our approach we consider computing the Nash equilibrium at each moment t and if the game setting requires it, to include the results in subsequent epochs. The scope of this endeavor is to compute and track the equilibria of such a game by using evolutionary computation tools. Moreover, we want to find out if we can identify certain trends in the movement of equilibria in time and cope with games with large number of players.

3 Discrete-Time Dynamic Cournot Oligopoly

In the following we will consider a discrete-time dynamic Cournot oligopoly benchmark with different dynamics and number of players.

A static asymmetric Cournot oligopoly [3] models n companies that produce s_1, s_2,...,s_n quantities of a product. The payoff function for the company i can be computed as:

$$u_i(s_1, ..., s_n) = s_i \cdot \left(b_i - \sum_{j=1,n} s_j \right), i = 1, ..., n$$

where $b_i, i = 1, ..., n$ are constants derived from the economic model.

The Nash equilibrium in this case, $NE = (NE_1, NE_2, ...NE_n)$, is:

$$NE_i = b_i - \frac{1}{n+1}\left(\sum_{k=1}^{n} b_k\right).$$

A discrete-time dynamic Cournot oligopoly model can be easily derived by inducing a change in the parameter controlling the payoffs, i.e. $b = (b_1, b_2, \ldots, b_n)$. Different types of changes will induce different dynamics in the position of the Nash equilibria of the game. Thus we can induce some random dynamics by using a certain probability distribution or we can induce a change following a certain trend.

Cauchy perturbation. If we consider a Cauchy perturbation we modify b by:

$$b_i(t+1) = b_i(t) + \Phi(0, \gamma), \tag{1}$$

where $\Phi(0, \gamma)$ represents a random number generated from a Cauchy distribution with mode 0 and scale parameter γ. Here γ controls the amplitude of the change from moment t to $t+1$.

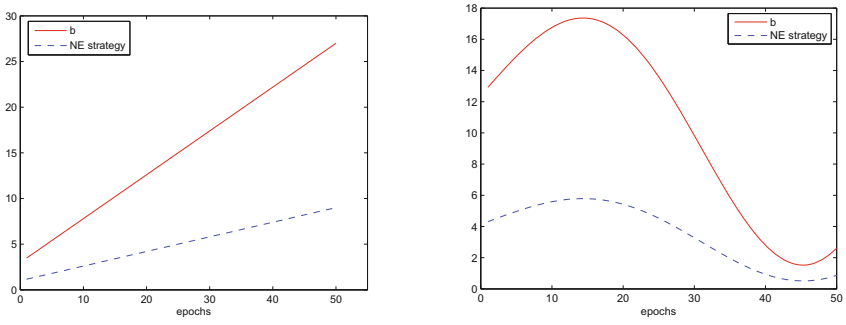
Linear dynamics. Let the game dynamics be described by the following equation:

$$\begin{cases} b(t) = a \cdot t + b, \\ a = \dfrac{8}{50}(n+1), \\ b = (n+1), \end{cases} \tag{2}$$

where $b(t)$ represents the value of the payoff parameter at generation t and n represents the number of players. In this case, the *Nash* equilibrium of the game is:

$$NE = \frac{b(t)}{n+1}.$$

Figure 1(left) represents the trajectory of the NE in time.



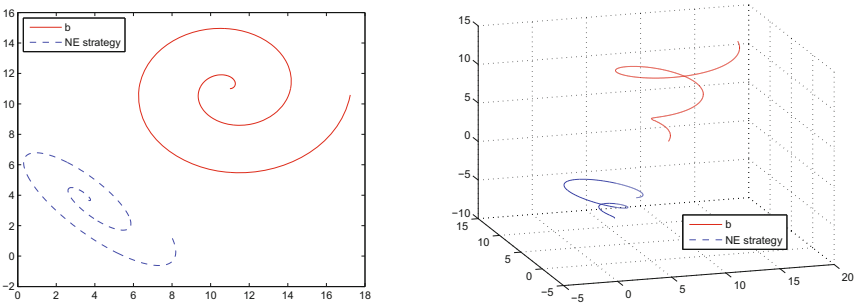
(a) Linear dynamics (b) Cosine dynamics

Fig. 1. The trajectory of NE and b in the linear case (a) and cosine (b).

Cosine dynamics. Let the game dynamics be described by the following equation:

$$b(t) = t\cos(t) + 10 \tag{3}$$

where $b(t)$ represents the value of the payoff parameter at generation t. The trajectory of the NE in this case is represented in Fig. 1, right (Fig. 2).



(a) 2 players (b) 3 players

Fig. 2. The trajectory of the NE and b for the spiral dynamics for 2 players (a) and 3 players (b).

Spiral dynamics. For a Cournot duopoly the dynamics of parameter b is described by the following equations:

$$\begin{cases} b_1(t) = t\cos(t) + 10, \\ b_2(t) = t\sin(t) + 10, \end{cases} \tag{4}$$

where $b_i(t)$ represents the value of the payoff parameter at moment t for player i.

For three players the dynamics of the game is described as:

$$\begin{cases} b_1(t) = t\cos(t) + 10, \\ b_2(t) = t\sin(t) + 10, \\ b_3(t) = t. \end{cases} \tag{5}$$

4 Dynamic Nash Extremal Optimization

In [15] a Differential Evolution algorithm, DET-DE is designed to compute and track Nash equilibria for a discrete-time Cournot oligopoly. The main features of DET-DE are: using a sentinel to identify if a change has occurred in the search space and using an adaptive mutation to boost the search after a change has been identified.

While DET-DE proved to be very efficient for games with small number of players (up to 10), further research showed that it did not scale very well to larger games, which was not surprising since it did not scale well for the static

case either [12]. However, the Nash Extremal Optimization algorithm in [12] does scale well, and therefore we have tried to adapt it to track the NE also in the dynamic environment.

The Dynamic Nash Extremal Optimization (DNEO) proposed is based on NEO [12] with the following new features: DNEO uses a population of EO individuals that independently search the space; DNEO uses a sentinel to identify changes in the search space; and to cope with changes, the best configuration for each EO pair of individuals is re-initialized in order to encourage the exploration of the new fitness landscape (Algorithm 1). DNEO uses only two parameters, the population size and the maximum number of iterations.

Algorithm 1. Dynamic Nash Extremal Optimization

Initialize randomly the population P of strategy profiles;
For each $P_k \in P$ set $P_k^{best} := P_i$;
repeat
  if *Change* detected **then**
    Re-initialize all P_k^{best};
  end if
  With each individual $P_k \in P$:
    - evaluate u_i for each player i;
    - find player j with the worst payoff, i.e. satisfying $u_j \leq u_i$ for all i;
    - replace randomly strategy of player j in P_k;
  if P_k Nash ascends P_k^{best} **then**
    set $P_k^{best} := P_k$.
  end if
until maximum number of iterations is reached.
Return the non-dominated elements of P^{best} at each epoch of the dynamic game.

5 Numerical Experiments

Numerical experiments were performed on all the four types of dynamics presented on Sect. 3. Results were averaged over 10 independent runs using different seeds for the random numbers generators.

Parameter settings. For the DET-DE we considered the following parameters: a population of 100 individuals and 50 epochs. On Each epoch the value b changed (according to the desired dynamic) and the search took place for 200 generations; $\sigma = \{0.5, 1, 5\}$, $p_{min} = 0.01$, $p_{max} = 0.07$. The basic DE parameters were: $C_r = 0.8$, and $F = 0.2$ (at each change F was set to 0.5 and was linearly decreased to 0.2). The probability of mutation, when a change was detected, was directly proportional with the amplitude of the change [15].

DNEO ran with a population of 50 individuals for the same number of generations and epochs as DET-DE.

Cauchy perturbation. γ was set between 1 and 2. Box plots representing the results obtained with DET-DE and DNEO for 2, 10, and 100 players are presented in Fig. 3. As expected, DET-DE did not scale well to 100 players, while DNEO performed significantly better compared to DET-DE only for 100 players, which is also true for the static variant of Cournot.

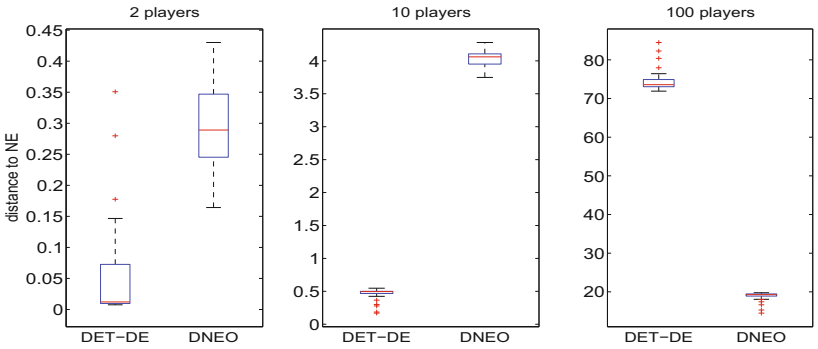
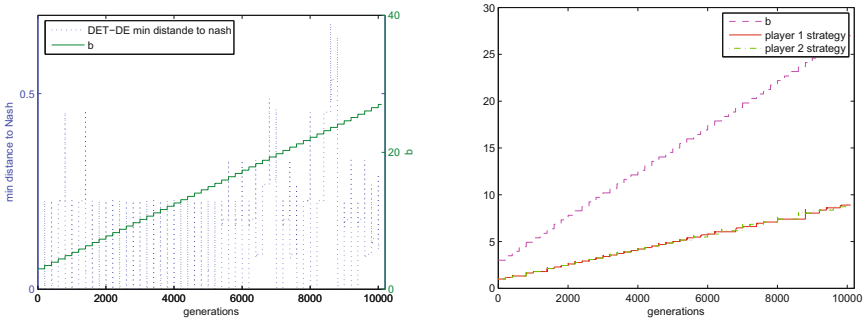


Fig. 3. Average minimum distance (for each epoch) to the NE for 2, 10, and 100 players, for the DET-DE and DNEO. Differences between the two methods are significant according to a Wilcoxon non-parametric test.

We may draw the conclusion that for small number of players DET-DE is more appropriate, while DNEO can be used for large games. Following this conclusion, the rest of the dynamics were only studied with DET-DE.

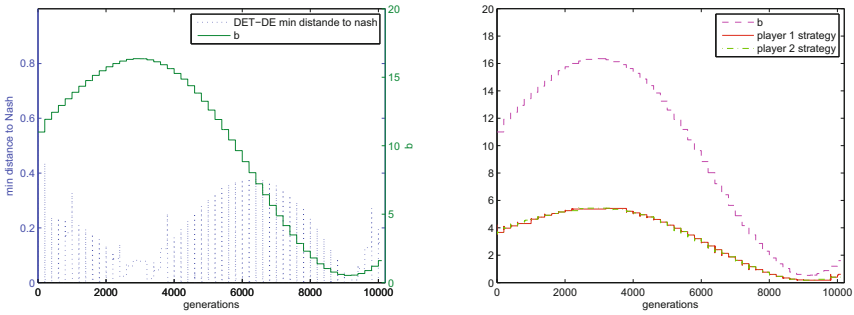
Linear dynamics. Figure 4 presents the results obtained for a 2 player Cournot dynamic game with a linear dynamics with $b \in [3, 27]$. As it can be observed DET-DE is able to track the Nash Equilibrium of the game in each epoch.



(a) Mean distance to NE.

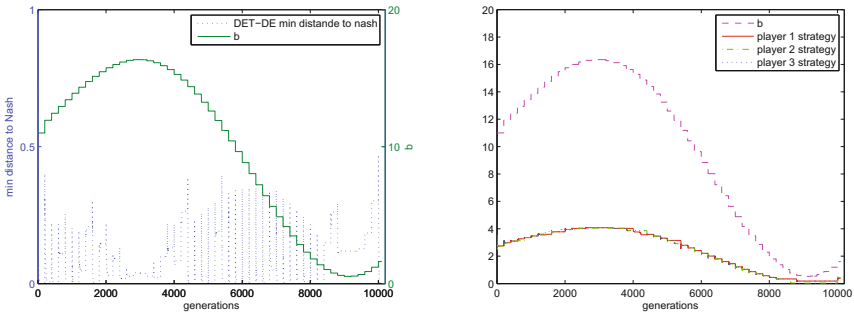
(b) Strategies in each epoch.

Fig. 4. Two player **linear** dynamic Cournot game: mean distance to *Nash* equilibrium and b values in each round (left), strategies for each player and b values in each round (right).



(a) Mean distance to NE. (b) Strategies in each epoch.

Fig. 5. Two player **cosine** dynamic Cournot game: mean distance to *Nash* equilibrium and b values in each round (left), strategies for each player and b values in each round (right).

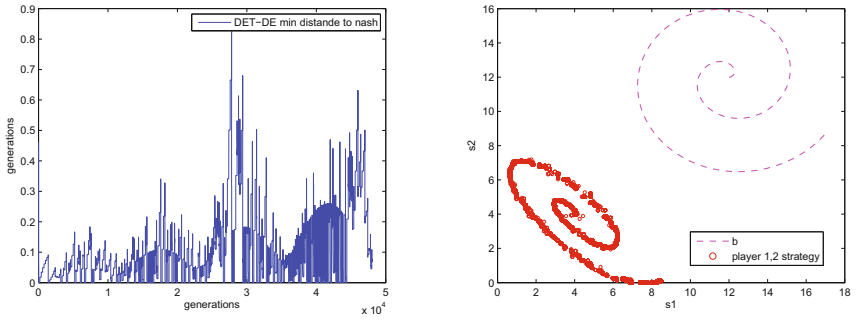


(a) Mean distance to NE. (b) Strategies in each epoch.

Fig. 6. Three player **cosine** dynamic Cournot game: mean distance to *Nash* equilibrium and b values in each round (left), strategies for each player and b values in each round (right).

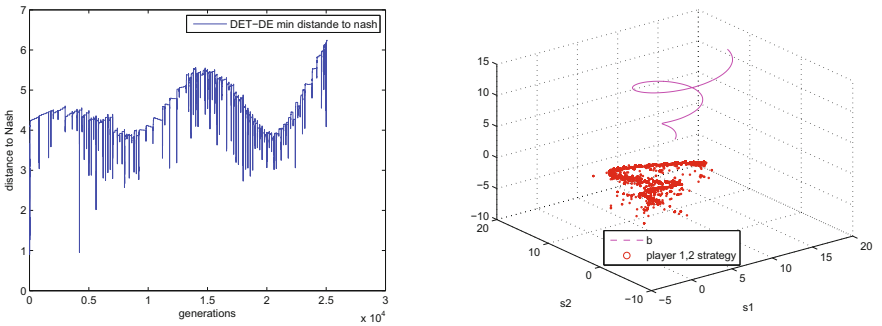
Cosine dynamics. Figure 5 and 6 presents the results obtained for 2 and 3 players. Again DET-DE was capable to track the position of the NEs. In both cases (2 and 3 players) $t \in [0, 12]$ is increased with step 0.25 in each generation.

Spiral dynamics. Figures 7 and 8 illustrate the results obtained for 2 and 3 players. DET-DE follows the Nash Equilibrium of the game each epoch.



(a) Mean distance to NE. (b) Strategies in each epoch.

Fig. 7. Two player **spiral** dynamic Cournot game: mean distance to *Nash* equilibrium and b values in each round (left), strategies for each player and b values in each round (right).



(a) Mean distance to NE. (b) Strategies in each epoch.

Fig. 8. Three player spiral dynamic Cournot game: mean distance to *Nash* equilibrium and b values in each round (left), strategies for each player and b values in each round (right).

6 Conclusions and Further Work

This paper tackles the problem of computing and tracking the Nash equilibrium of a discrete - time dynamic Cournot oligopoly with four types of dynamics with two evolutionary approaches: a differential evolution one, DET-DE and one based on extremal optimization - DNEO.

For the considered games, DET-DE performed significantly better than DNEO for a small number of players (2,10); for 100 players the results obtained by DNEO were in turn much better than those of DET-DE, but still not very precise, indicating that further work may include a hybridization that would improve both results.

Acknowledgments. The authors would like to acknowledge the support received from the OPEN-RES 212/2012 grant (www.uefiscdi.gov.ro).

References

1. Agiza, H., Elsadany, A.: Nonlinear dynamics in the Cournot duopoly game with heterogeneous players. Phys. A: Stat. Mech. Appl. **320**, 512–524 (2003), http://www.sciencedirect.com/science/article/pii/S0378437102016485
2. Bischi, G.I., Kopel, M.: Equilibrium selection in a nonlinear duopoly game with adaptive expectations. J. Econ. Behav. Organ. **46**(1), 73–100 (2001), http://www.scopus.com/inward/record.url?eid=2-s2.0-0001902879&partnerID=tZOtx3y1
3. Cournot, A.: Recherches sur les Principes Mathematique de la Theorie des Richesses. Hachette, Paris (1838)
4. Dumitrescu, D., Lung, R.I., Mihoc, T.D.: Generative relations for evolutionary equilibria detection. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (2009)
5. Greenfield, D., Kwoka, J.: The Cost Structure of Regional Transmission Organizations. Energy J. **32**(4), 183–218 (2011), http://www.scopus.com/inward/record.url?eid=2-s2.0-84864972092&partnerID=tZOtx3y1
6. Hu, R., Chen, Q.: Chaotic dynamics and chaos control of cournot model with heterogenous players. In: Jiang, L. (ed.) Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE 2011), November 19-20, 2011, Melbourne, Australia SE - 70. AISC, vol. 110, pp. 549–557. Springer, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-25185-6_70
7. Karafyllis, I., Jiang, Z.P., Athanasiou, G.: Nash equilibrium and robust stability in dynamic games: a small-gain perspective. Comput. Math. Appl. **60**(11), 2936–2952 (2010), http://www.sciencedirect.com/science/article/pii/S0898122110007510
8. Kian, A.R., Cruz, J.B.: Bidding strategies in dynamic electricity markets. Decis. Support Syst. **40**(3-4), 543–551 (2005), http://www.scopus.com/inward/record.url?eid=2-s2.0-24944591946&partnerID=tZOtx3y1
9. Li, Q.R., Ma, S.F., Chen, L., Long, Y., Wei, L.Y.: Genetic algorithm approach to dynamic mixed behavior traffic network equilibrium problem. Chang'an Daxue Xuebao (Ziran Kexue Ban)/J. Chang'an Univ. (Nat. Sci. Ed.) **27**(6), 87–90 (2007), http://www.scopus.com/inward/record.url?eid=2-s2.0-38549157564&partnerID=tZOtx3y1
10. Long, N.V.: A Survey of Dynamic Games in Economics, vol. 1. World Scientific Publishing Co. Pte. Ltd. (2010), http://EconPapers.repec.org/RePEc:wsi:wsbook:7577
11. Lung, R.I., Dumitrescu, D.: Computing Nash equilibria by means of evolutionary computation. Int. J. Comput. Commun. Control **3**, 364–368 (2008)
12. Lung, R.I., Mihoc, T.D., Dumitrescu, D.: Nash extremal optimization and large cournot games. In: NICSO, pp. 195–203 (2011)
13. Nash, J.: Non-Cooperative Games. Ann. Math. **54**(2), 286–295 (1951), http://dx.doi.org/10.2307/1969529

14. Suciu, M., Gaskó, N., Lung, R.I., Dumitrescu, D.: Nash equilibria detection for discrete-time generalized cournot dynamic oligopolies. SCI, vol. 512, pp. 343–354 (2014), http://www.scopus.com/inward/record.url?eid=2-s2.0-84883646840&partnerID=tZOtx3y1
15. Suciu, M.A., Lung, R.I., Gaskó, N., Dumitrescu, D.: Differential evolution for discrete-time large dynamic games. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013, pp. 2108–2113 (2013), http://dx.doi.org/10.1109/CEC.2013.6557818

Theory on Evolutionary Computation

Efficient Real-Parameter Single Objective Optimizer Using Hierarchical CMA-ES Solvers

Madalina M. Drugan(✉)

Artificial Intelligence Lab, Vrije Universiteit Brussels,
Pleinlaan 2, 1050 Brussels, Belgium
madalina.drugan@gmail.com

Abstract. Monte Carlo Tree Search (*MCTS*) is a novel machine learning paradigm that is used to find good solutions for complex optimization problems with very large search spaces (like playing GO). We combine *MCTS* with Covariance Matrix Adaptation Evolution Strategies (*CMA-ES*) to efficiently optimize real-parameter single objective problems by balancing the exploitation of promising areas with the exploration of new regions of the search space. The novel algorithm is called *hierarchical CMA-ES* and it is influenced by both machine learning and evolutionary computation research areas. Like in evolutionary computation, we use a population of individuals to explore the commonalities of CMA-ES solvers. These CMA-ES solvers are structured using a MCTS tree like structure. Our experiments compare the performance of hierarchical CMA-ES solvers with two other algorithms: the standard CMA-ES optimizer, and an adaptation of MCTS to solve real-parameter problems. The hierarchical CMA-ES optimizer has the best empirical performance on several benchmark problems.

1 Introduction

Natural paradigms are a major inspiration source in many areas of statistics and computer science. Machine Learning (ML) could be considered a collection of nature inspired paradigms that map inputs (as sensors and other empirical data structures) using learning and optimization paradigms to automatically build a model that makes predictions or decisions. A recent trend in ML is the transfer of knowledge from one area to another. Multi-armed bandits (MAB) [5] is a well-established decision problem that was studied since the 1930s with application in different domains. Although at first they seem very different, MAB and EC are two learning techniques that address basically the same problem: the maximization of the agent's reward in MAB and fitness function in EC in an unknown environment. The MAB paradigm is attractive because it provides a mathematical framework for modelling decision making with a simple algorithm, while the main strength of EC is its general applicability and computational efficiency. We highlight two important differences between the EC and MAB techniques:

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_10

(1) EC is a stochastic optimization algorithm for large (usually) deterministic environments, whereas MAB is (usually) a non-stochastic algorithm for stochastic environments. (2) EC optimizes multi-dimensional search spaces whereas multi-dimensional environments are a hot topic in MABs. Section 2 gives a short overview of the ML techniques used.

Real-parameter single objective problems are complex optimization problems that require engineered design techniques to solve them. Evolutionary Strategies (ES) [3] is a variant of EC for continuous environments that uses only mutation to generate new individual solutions and deterministically selects the best individuals. ES considers that a well-performing optimizer tunes its parameters automatically since manually tuning parameters is a time and resource consuming process. Covariance Matrix Adaptation Evolution Strategies (CMA-ES) [8] wants to be a parameter free ES that generates new solutions using a multivariate normal distribution to learn the distribution of the solutions. CMA-ES is a popular derivative free numerical optimization technique due to its large applicability on non-linear, non-convex, continuous optimization functions.

Our main contribution is to steer real-parameter optimization with a variant of Monte Carlo Tree Search (MCTS) that uses hierarchical CMA-ES solvers. Each node in the hierarchy of MCTS is a CMA-ES solver over a search space defined by a D-rectangle. The search space of a parent CMA-ES solver is iteratively decomposed in a tree of finer disjunct D-rectangles that have associated CMA-ES solvers as well. Each CMA-ES solver has associated: (1) a population of solutions from which the generation of new individual solutions is adapted, and (2) a value corresponding with the mean fitness of the individual solutions in the corresponding D-rectangle. A solver is selected according to its value such that the most promising regions are exploited the most by sampling fit individual solutions in the corresponding D-rectangle. On the other hand, each solver can be selected with a non-zero probability such that new regions of the search space are sampled. Similarly as in MCTS, the exploration versus exploitation balance is maintained by selecting the less fit CMA-ES solvers with low frequency and fit solvers with high frequencies.

Section 3 introduces the tree structure to be used with the variant of MCTS for real-parameter optimization. Each node in the tree is characterised by a D-rectangle representing the Cartesian product of D continuous and contiguous intervals and a value representing the mean value of the individuals sampled in the D-rectangle. The relation between nodes is explained.

In Sect. 4, we introduce a real-coded MCTS with the goal of finding the optimal solution of a given continuous function. It starts from a set of solutions generated uniformly at random, and Monte Carlo tree like search gradually focuses towards promising areas of the search space. We classify these algorithms using two main policies to generate individual solutions: (1) uniform random, and (2) adaptive normal distributions. Each iteration, a population of solutions is generated from a node selected with a stochastic multi-armed bandit (MAB) paradigm [5]. We use two different policies to traverse the MCTS nodes: (1) from the root to leaves in a top-down manner and (2) selecting the most promising node anywhere in the MCTS tree.

Section 5 combines real-coded MCTS with CMA-ES solvers [2] into an efficient optimizer for multi-dimensional environments. We call this algorithm *hierarchical CMA-ES solver*. The nodes in MCTS are CMA-ES solvers on bounded D-rectangles that generate individuals solutions using an adaptive multivariate normal distribution. Hierarchical CMA-ES solver is different than real-coded MCTS because the later uses the uniform random distribution to generate new individuals.

Section 6 experimentally compares the proposed algorithms and their variants on several multi-dimensional benchmark functions [10] for real-parameter optimization. We show that the most efficient algorithm is hierarchical CMA-ES solvers and we explain its performance using several metrics for both optimization and learning. If the standard CMA-ES requires large populations to efficiently solve a problem, we experimentally show that hierarchical CMA-ES solvers work well with small populations and bounded search spaces. Section 7 concludes the paper.

2 Preliminaries

Multi-armed bandits. Stochastic MAB [1] is a popular mathematical formalism to study sequential decision-making under uncertainty where the objective is to maximize a long-term objective. An agent must choose between N-arms such that the expected reward over time is maximized. The distribution of the stochastic pay-off of the different arms is assumed to be unknown to the agent. A MAB algorithm starts by uniformly exploring the N-arms, and then gradually focuses on the arm with the best observed performance. Since, the means are unknown one has to allocate a number of trials over the different arms so that, based on the obtained rewards, the optimal arm is identified as soon as possible and this with (very) high confidence.

Exploration vs exploration trade-off in EC and MABs. The exploration (the search for new useful solutions) versus exploitation (the use and propagation of such solutions) trade-off is an attribute of successful adaptation in both MAB and EC. In EC, the exploration implies the evaluation of new solutions that could have low fitness and the exploitation means the usage of already known good solutions, e.g. selection. Selecting and using these trade-off strategies are not trivial and actually, they can increase the time needed to find an acceptable solution.

In MAB, exploration means that one tries a suboptimal arm to improve the estimate of the mean reward value while exploitation means that one tries the best observed so far arm which is not necessarily the true best arm. An arm selection policy determines which arm is selected at what time step based on the rewards obtained so far. The selection policies for the MAB-problem is essential for its performance. An important heuristic that has emerged is that good policies, e.g. variants of the upper confidence bound (or UCB) policy, are *optimistic in the face of uncertainty* [11].

There are many variants of MABs with different goals (i.e. minimizing the loss of pulling suboptimal arms or identifying optimal arms), with different mechanisms for efficient optimization.

Monte Carlo Tree Search (MCTS). One MAB variant is the hierarchical bandit approach, or MCTS, [4,9] where the reward of one arm in the hierarchy is the reward of another node that is one level deeper in the hierarchy [11]. MCTS is a recently proposed search method that builds a search tree in an incremental and asymmetric manner accordingly to a tree policy that selects the node with the highest priority to expand. The tree policy needs to balance exploration versus exploitation, which for MCTS methods resembles the same trade-off as in EC. Exploration means to search in areas not sampled yet, whereas exploitation means to search in promising areas. Each round in MCTS consists of four steps: selection, expansion, simulation, and back-propagation.

Selection starts from the root, it selects successive expandable child nodes down to a leaf node. It selects child nodes that expand the tree towards the most promising moves, which is the essence of MCTS. A node is expandable if it represents a non-terminal state and it is unvisited. *Expansion:* unless a stopping criteria is met, MCTS creates one or more child nodes and chooses from them a node, designated as the current node, using a tree policy. If no child was created, the simulation starts from a leaf node. *Simulation* plays at random from the current node using a default policy. *Back-propagation* uses the results from the previous steps to update the information in the nodes on the path from the current node to the root. MCTS is a statistical any-time algorithm for which more computing power means better results.

A popular technique to design MCTS is to use stochastic bandits, i.e. upper confidence bound (UCB1) [1]. To select the next node to expand, the rewards are sampled from an unknown distribution. UCB1 is a very simple and efficient MAB and has appealing theoretical properties. Each promising candidate is evaluated according to the UCB1 policy. This class of algorithms is denoted as the upper confidence bound for tree (UCT) [9]. UCT builds incrementally a search tree using random samples in the search space by expanding the nodes selected by the arm selection policy [4]. This approach is largely responsible for the success of Monte Carlo Tree Search (MCTS) where other methods fail, e.g. for the game of GO.

[7] proposes the schemata bandits algorithm to solve binary combinatorial optimisation problems, like the trap functions and NK landscape, where potential solutions are represented as bit strings. The schemata from the schema theorem for genetic algorithms are structured as hierarchical multi-armed bandits in order to focus the optimisation in promising areas of the search space.

[6] uses a variant of MCTS to optimize continuous and stochastic sequential decision making problems.

3 Real-Parameter Tree Structure

In order to adapt Monte Carlo Tree Search (MCTS) to global optimization of continuous functions, we use a different definition for nodes and leaves for multi-dimensional real-parameter environments. We consider only maximization problems, a requirement of the multi-armed bandit paradigm. The tree is structured as follows:

Tree nodes. Each node in the tree has the four attributes: (1) a D-rectangle, (2) a value, (3) several children and (4) at most one parent. Here, we consider D larger or equal to 1, thus $D \geq 1$ and that the search space is normalised in all D dimensions. Consider a node i in the tree. Let m_i^j and M_i^j be the upper and the lower bound in each dimension j for the D-rectangle of node i, respectively, such that $0 \leq m_i^j < M_i^j \leq 1$. A D-rectangle is defined as the Cartesian product over all dimensions $\times_{1 \leq j \leq D}[m_i^j, M_i^j)$. A D-rectangle is defined as the Cartesian product over all dimensions $\times_{1 \leq j \leq D}[m_i^j, M_i^j)$.

The most general node in the tree is called *root*. The *root* of the tree has associated a D-rectangle representing the bounds of the D dimensional continuous search space to be optimised. The root node has no parents and it has a D-rectangle that contains all the other D-rectangles in the tree. For simplicity, we assume that all the edges of any D-rectangle associated with a node are equal, and thus we have D-cubes. The root node has a D-cube defined by the Cartesian product $[0, 1] \times \ldots \times [0, 1]$ in all D dimensions. The root's volume is the product over all intervals of length 1.

A *leaf* node is a D-cube with no children; thus this node is not split in other smaller D-cubes.

Children. Let K be the fixed number of equal sized non-overlapping intervals considered in each dimension $K \geq 2$. Each *node* i, which is not a leaf, has K^D children defined as the Cartesian product of K subintervals in each dimension. Thus, each interval $[m_i^j, M_i^j)$ is divided in each dimension j in K disjoint subintervals of length $\xi_i^j \leftarrow \lceil \frac{M_i^j - m_i^j}{K} \rceil$. Each child of the i-th D-cube $\times_{1 \leq j \leq D}[m_i^j, M_i^j]$ is another D-cube $\times_{1 \leq j \leq D}[m_i^j + \frac{k^j}{K}\xi_i^j, m_i^j + \frac{k_i^j+1}{K}\xi_i^j)$, where the index k_i^j ranges from 0 till $K - 1$.

Parents. Each node that is not the root has a single parent for which the D-cube contains the D-cube of its children.

Value of nodes. Each node in the tree, thus also the leaves and the root, has associated a value that is the mean fitness value of the solutions inside the bounds of that node. Let $\{s_1, s_2, \ldots, s_{n_i}\}$ be the set of n_i solutions evaluated in the i-th D-cube. The estimated value of this node is $\widehat{f_i} = \sum_{j=1}^{n_i} \frac{f(s_j)}{n_i}$. When $n_i \leftarrow \infty$, the estimated value of this node is $f_i = \int_{\mathbf{m_i}}^{\mathbf{M_i}} f(\mathbf{x}) d(\mathbf{x})$ the area under function f is the estimated value of the corresponding D-cube. By convention, when there is no individual solution generated in a D-cube, the value of that node is set to the minimal value $\widehat{f_i} = 0.001$ and $n_i = 1$ such that the node will

be sampled sometimes. Each time a child is selected, we update the counters $n \leftarrow n+1$ and $n_i \leftarrow n_i + 1$.

Tree properties. Let ℓ be the number of nodes on the path between the root and a leaf. Each leaf i has a minimal size of the interval $\min_{1 \leq j \leq D} M_i^j - m_i^j > \epsilon$, where $\epsilon > 0$ a small number. Then, $\ell \leftarrow \log_K \lceil \frac{1}{M_i^0 - m_i^0} \rceil$.

When $K = 2$, the tree has the longest path between the root and a leaf. A complete tree with length ℓ has K^{D^ℓ} nodes. This means that if $K = 2$, $D = 10$ and $\ell = 10$, we could have $2^{10^{10}}$ nodes in the complete tree which is a huge number. The root alone has $2^D = 2^{10} = 1024$ children, and a path generated by a solution from the root to the leaf has $\ell = 2^{10}$ nodes. When 10^3 solutions are generated uniformly at random, we may generate 10^6 nodes of the tree.

Therefore, a computational feasible MCTS algorithm generates a small set of nodes from this tree by focusing the search in the promising areas of the search space. In the next section, we will present several methods to traverse the tree and to generate solutions in this tree.

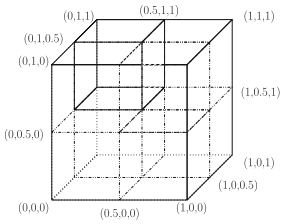


Fig. 1. Cubes representing the root and its 8 children.

An example. If $K = 2$ and $D = 3$, then there will be $2^3 = 8$ children for each node of the tree. Thus, the root has 8 children with the corresponding cubes $[0, 0.5]^3$, $[0, 0.5]^2 \times [0.5, 1]$, $[0, 0.5] \times [0.5, 1]^2$, and so on. Note that these cubes have equal volume that is $0.5^3 = 1/8$ from the root's volume. Similarly, a child of such cube has exactly $1/8 \cdot 1/8 = 1/64$ from the root's volume. Figure 1 show that spatial distribution of the 8 children cubes into the cube representing the parent.

Let $s_1 = (0.2, 0.7, 0.9)$, $s_2 = (0.6, 0.2, 0.7)$ and $s_3 = (0.3, 0.8, 0.9)$ be 3 sampled individual solutions. Let $f : [0, 1]^D \rightarrow \mathbb{R}$ be a continuous real-valued function and $f(s)$ is the value of solution s. As expected, the root node contains all solutions. Further, two solutions correspond to a child of the root, s_1 $s_3 \in [0, 0.5] \times [0.5, 1]^2$, and the third solution corresponds to another of the root's children $s_2 \in [0.5, 1] \times [0, 0.5] \times [0.5, 1]$. The estimated value of the root is the mean of the three solutions whereas the estimated value of the node containing the solutions s_1 and s_3 is their mean, and the estimated value of the node containing s_2 is equal to $f(s_2)$. If no other solutions are generated, the rest of the root's children have the default minimal value of 0.001, which is a pessimistic initialization. Figure 2 illustrates the example of MCTS structure with $K = 2$ and $D = 3$.

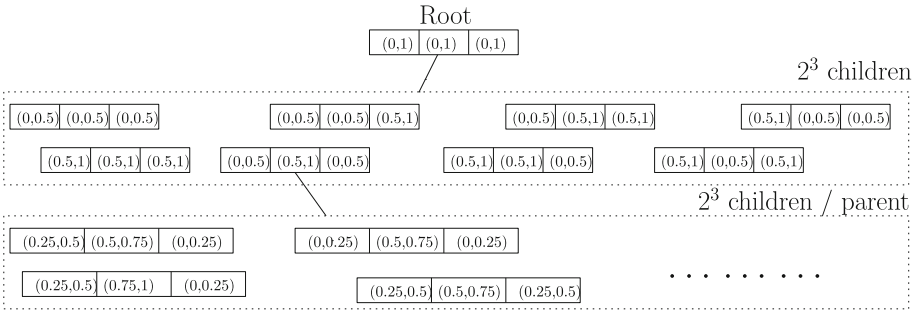


Fig. 2. An example of real-parameter MCTS optimizer.

1: **function** Baseline_real-parameter_MCTS *rMCTSN*
2: Generate N random individual solutions, and set $n \leftarrow N$ {Initialization}
3: Build a first Monte Carlo Tree instance
4: **while** the stopping criteria is *NOT* met **do**
5: Select the node i that maximizes the value $\widehat{f}_i + C \cdot \sqrt{\frac{\ln n}{n_i}}$ {Selection}
6: Generate N solutions $S_i \leftarrow \{s_1, \ldots, s_N\}$ in the selected node i {New solutions}
7: **for** all solutions s_j in the population S_i **do**
8: Update the mean value and the counters of all the nodes that contain s_j {Update}
9: **end for**
10: $n \leftarrow n + N$;
11: **end while**
12:
13: **return** the best individual solution found
14: **end function**

4 Real-Parameter MCTS Algorithms

In this section, we investigate several selection and propagation policies for Monte Carlo Tree Search (MCTS) for real-parameter optimization. We classify the algorithms using two criteria. We consider two policies to select a node, and these are: (1) a top-down approach that select nodes by traversing the tree from the root to the leaves, and (2) a uniform approach where all the nodes in the tree are considered for selection. The latter policy is specific for the standard MCTS, whereas the former policy is a top-down approach similar with deterministic optimistic optimization (DOO) [11]. We also consider two policies for generating solutions in the selected nodes using either: (1) a uniform random distribution, or (2) a covariance adaptive matrix that adapt a multi-variate distribution (this approach is introduced in Sect. 5).

4.1 Baseline Real-Parameter MCTS

A baseline real-parameter MCTS algorithm considers all the tree nodes for selection, and it generates solutions using a uniform distribution on the selected D-cube. The pseudo-code is presented in Function BASELINE_REAL- PARAMETER_MCTS (*rMCTS*).

At the initialization, a set of N initial solutions is generated uniform randomly in the initial D-cube $[0,1]^D$, where $D \geq 1$. Using each of these initial solutions, we update from the root to the leaves the estimated value for all the nodes whose D-cube contains that solution. This represents the first instance of the Monte Carlo tree. A tree iteration includes the following successive steps: (1) the selection of a tree node, (2) the generation of a population set of N solutions within the selected node, and (3) the update of the tree for each individual solution.

A UCB1 policy is associated with the tree, such that each iteration, a tree node i is selected using the index value of the UCB1 algorithm

$$\widehat{f}_i + C\sqrt{\frac{\ln n}{n_i}} \tag{1}$$

where $\widehat{f}_i$ is the average fitness value of the solutions in the i-th D rectangle. The counter n_i represents the number of times the node i is selected, and n is the total number of solutions that were generated by *rMCTS*. The exploration parameter $C > 0$ is usually set to $C = 2$, but its value usually depends on the properties of the optimization problem.

The first component of Eq. 1 corresponds to exploitation. A node with a high average fitness is selected more often than a node with low average fitness. The second component of Eq. 1 corresponds to exploration and allows low fitted nodes to be sometime selected in order to explore new regions of the search space. N individual solutions, $S_i \leftarrow \{s_1, \ldots, s_N\}$ are generated uniform randomly within the bounds of the D-cube associated with i. For each individual solution s, we update the mean fitness value of all nodes that contain the fitness value of s. Thus, the value of the node i is updated, but also the average fitness value of all nodes, from the root to the leaves, on a path that contains that node. *rMCTS* stops when a maximum number of solutions was evaluated.

4.2 Top-Down Node Real-Parameter MCTS

The top-down selection policy always starts with the root node and ends up in a leaf node that contains an individual solution. The pseudo-code of this algorithm is given in Function TOP-DOWN_REAL-PARAMETER_MCTS (*tMCTS*). Each tree node is associated to a UCB1 strategy to select a child of the node. *tMCTS* uses the same index for the UCB1 strategy like in Eq. 1, but with different meaning.

function Top-down_real-parameter_MCTS *tMCTSN*
Generate N random individual solutions, and set $n \leftarrow N$ {Initialization}
Build a first Monte Carlo tree instance
while the stopping criteria is *NOT* met **do**
  Select the root node {Selection}
  while the node i was visited before and i is not a leaf node **do**
    Select the node j that maximizes the index value $\widehat{f}_{ij} + C \cdot \sqrt{\frac{\ln n_i}{n_{ij}}}$
  end while
  Generate N individual solutions $S_i \leftarrow \{s_1, \ldots, s_N\}$ in the selected node i {New solutions}
  for all solutions s_j in the population S_i **do**
    Update the mean value and the counters of all the nodes that contain s_j {Update}
  end for
  $n \leftarrow n + N$
end while

return the best individual solution found
end function

Let i be a tree node. We select a child node j of the node i using a UCB1 with the index value

$$\widehat{f}_{ij} + C\sqrt{\frac{\ln n_i}{n_{ij}}} \tag{2}$$

where the estimated value $\widehat{f}_{ij}$ equals the average fitness value of solutions corresponding to node j, and thus also to its parent node i. The counter n_i indicates how many time the node i was selected whereas the counter n_{ij} measures how many times the child j was selected using the UCB1 instance. The parameter C is a positive real, as before.

Solutions are generated for nodes there are selected for the first time, or for leaf nodes. For each solution generated, the tree is updated as before.

The counters of two UCB1 strategies associated to nodes that are not on the same path between the root and a leaf have independent counters and values. Two UCB1 strategies that are on the same path are correlated, the closer on the same path they are, the larger the correlation. Thus, a parent and its children have correlated UCB1 index values since they share some of the individual solutions.

The baseline and top-down policies for the real-parameter MCTS are experimentally compared in Sect. 6. In order to compare the algorithms, we use the same stopping rule, naming a maximum number of generated solutions.

Related work. Interesting parallels can be made between the *tMCTS* and the deterministic optimistic optimization (*DOO*) [11]. The main difference is the use of a population of solutions in *tMCTS*, which has an impact on both updating and expansion rules. For example, in DOO, each node is evaluated using a single

function Hierarchical_CMA-ES_solvers *CMA-MCTS*λ,N
Assign uniform weights $\omega_i = 1/\lambda$ {Initialization}
Generate N random individual solutions and initialize root's (λ, N) CMA-ES
Build a first Monte Carlo tree instance
while the stopping criteria NOT met **do**
  Select the node i that maximizes the value $\widehat{f}_i + C\sqrt{\frac{\ln n}{n_i}}$ {Selection}
  Generate N individual solutions $S_i \leftarrow \{s_1, \ldots, s_N\}$ using the corresponding (λ, N) CMA-ES solver {New Solutions}
  for all best λ solutions s in the population S_i **do**
    for all the nodes j that contain s **do**
      {Update}
      Update the mean value and the counters of j
      Update the corresponding (λ, N) CMA-ES solver
    end for
  end for
  $n \leftarrow n + N$; $t \leftarrow t + 1$
end while

return the best individual solution found
end function

function evaluation in the middle of the representing interval, whereas in *tMCTS* the value of each function is the average fitness of individual solutions spread in the same interval. This is an improvement in our algorithm because the value of each node is a better approximation of the expected optimal value in the given interval.

4.3 Exploration/Exploitation Trade-Off

We consider that the exploration/exploitation trade-off in real-parameter MCTS algorithms is a measure of the selection pressure that is correlated with the size of the generated population N. A large population size N means a smaller number of selection steps in the tree but a lower selection pressure. The selection pressure comes from the successive tree iterations of solutions meaning that two solutions generated in different tree iterations are correlated. The best nodes in the previous iterations generate solutions in the current and future iterations, whereas the solutions in the same tree iteration are independently generated.

Another way to control the exploration/exploitation trade-off is the exploration parameter C in the UCB1 index from Eq. 1 and 2. A small C value means that the node with the best fit individuals is selected very often, whereas a large C gives a large weight to the exploration term meaning that all nodes are selected often. Thus, unlike in a standard MCTS, both the population size and the exploration parameter control the exploration/exploitation trade-off.

5 Hierarchical CMA-ES Solvers

In this section, we propose a variant of the real-coded Monte Carlo Tree Search (MCTS) that interlace with (λ, N) CMA-ES solvers to efficiently exploit the structural information contained in the individual solutions. We denote this algorithm as *hierarchical CMA-ES solvers* (*CMA-MCTS*) and its pseudo-code is presented in Function HIERARCHICAL_CMA-ES_SOLVERS. A tree with only one node, the root node, coincides with the standard (λ, N) CMA-ES algorithm. Each node of this tree has associated one (λ, N) CMA-ES solver to generate populations of solutions that exploit the promising regions of the search space.

Baseline CMA-MCTS algorithm. *CMA-MCTS* alternates the selection and updating steps of MCTS with the generation of new fit solutions of (λ, N) CMA-ES. At initialization, N individuals are uniform randomly generated and the (λ, N) CMA-ES of the root node is initialized from this population. Like in the standard CMA-ES, the best fraction of the proposed individual solutions, λ, are used to update the covariance matrix and the step size of each (λ, N) CMA-ES solver. The mean value of the selected node is also updated using the best λ solutions generated by the corresponding (λ, N) CMA-ES solver.

Each iteration, a node i is selected using a UCB1 strategy over all nodes in MCTS, like in the baseline real-parameter MCTS. The larger the estimated mean fitness $\widehat{f}_i$ is, the higher the probability that the node i is selected. A tree node with a bad estimation could be selected less often, therefore the exploration term $C\sqrt{\frac{\ln n}{n_i}}$ ensures that each tree node gets a fair number of trials, where C, n and n_i as before. There are N solutions sampled with the corresponding CMA-ES solver. The most fit λ solutions are used to update the estimated functions of the tree nodes on the path between the root node, the selected node and the possible leaves. In case a solution is generated outside the bounds of the given D-cube, we ignore that solution and, instead, we sample a new solution within the given bounds. This step is the expansion step because new tree nodes are created when the leaf node is not already in MCTS. The algorithm stops when a maximum number of individual solutions were sampled.

CMA-ES solver. In the following, we describe the CMA-ES algorithm we use in each node of *CMA-MCTS*. Consider the selected node i. At initialization, the first generation of N individual solutions $\{x_1^{(1)}, \ldots, x_N^{(1)}\}$ are uniform randomly generated. A first estimation of the covariance matrix for $\widehat{\mathbf{C}}_i^{(1)}$ is computed for the first N solutions with the following equation

$$\widehat{\mathbf{C}}_i^{(t)} = \sum_{j=1}^{\lambda} \omega_j \frac{\left(x_{1:N}^{(t)} - \boldsymbol{\mu}_i^{(t)}\right)}{\sigma^{(t)}} \cdot \frac{\left(x_{1:N}^{(t)} - \boldsymbol{\mu}_i^{(t)}\right)^T}{\sigma^{(t)}} \tag{3}$$

where $i = 1$ for the root node. The mean $\boldsymbol{\mu}_i^{(t)}$ is updated for the t-th iteration

$$\boldsymbol{\mu}_i^{(t)} = \sum_{i=1}^{\lambda} \omega_i \mathbf{x}_{i:N}^{(t)}$$

where $\sum_{i=1}^{N} \omega_i = 1$, and $\omega_1 \geq \omega_2 \geq \ldots \geq \omega_N > 0$. Only a fraction of solutions λ, where $1 \leq \lambda \leq N$, are selected to update the covariance matrix and the fitness mean. The λ individual solutions that are selected are the most fit individuals from the population. This is considered similar to the truncation selection. $\mathbf{x}_{j:N}^{(t)}$ is the j-th best individual out of $\{\mathbf{x}_1^{(t)}, \ldots, \mathbf{x}_N^{(t)}\}$ individuals and $j : N$ denotes the index of the i-th ranked individual a population of N individuals for a maximization problem, $f(\mathbf{x}_{1:N}^{(t)}) \geq f(\mathbf{x}_{2:N}^{(t)}) \geq \ldots \geq f(\mathbf{x}_{N:N}^{(t)})$. In a practical setting, $\lambda \approx \lceil \frac{N}{2} \rceil$, and $\omega_i \propto N - j + 1$.

Iteratively, in each selected node i, a population on N individuals is generated from a normal distribution

$$\mathbf{x}_i^{(t)} \sim \boldsymbol{\mu}_i^{(t)} + \sigma_i^{(t)} \mathcal{N}\left(\mathbf{0}, \widehat{\mathbf{C}}_i^{(t)}\right)$$

where $\sigma_i^{(t)}$ is the variance, or the step size, it is also adapted

$$\sigma_i^{(t+1)} = \sigma_i^{(t)} \times e^{\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma^{(t+1)}\|}{E\|\mathcal{N}(0,\mathbf{I})\|} - 1\right)}$$

where

$$\mathbf{p}_\sigma^{(t+1)} = (1 + c_\sigma)\mathbf{p}_\sigma^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)m_{\boldsymbol{\omega}}}\mathbf{C}_i^{(t)^{-1/2}} \frac{\boldsymbol{\mu}_i^{(t+1)} - \boldsymbol{\mu}_i^{(t)}}{\sigma^{(t)}}$$

and $m_{\boldsymbol{\omega}}^{-1} = \sum_{i=1}^{N} \omega_i^2$, $c_\sigma = \frac{m_{\boldsymbol{\omega}}+2}{D+m_{\boldsymbol{\omega}}+5}$ and $d_\sigma = 1 + c_\sigma + 2\max\{0, \sqrt{\frac{m_{\boldsymbol{\omega}}-1}{D+1}} - 1\}$.

For each node i, its covariance matrix is updated from the previous covariance matrix

$$\mathbf{C}_i^{(t+1)} = (1 - c_1 - c_m + (1 - h_\sigma)c_1 c_c(2 - c_c))\mathbf{C}_i^{(t)} + c_1 \mathbf{p}_\sigma^{(t+1)}(\mathbf{p}_\sigma^{(t+1)})^T + c_m \cdot \widehat{\mathbf{C}}_i^{(t)}$$

where $\widehat{\mathbf{C}}_i^{(t)}$ is the empirical covariance matrix defined in Eq. 3 and $c_1 = \frac{2}{(D+1.3)^2+m_{\boldsymbol{\omega}}}$, $c_c = \frac{4+m_{\boldsymbol{\omega}}/D}{D+4+2m_{\boldsymbol{\omega}}/D}$, and $c_m = \min\{1 - c_1, 2\frac{m_{\boldsymbol{\omega}}-2+1/m_{\boldsymbol{\omega}}}{(D+2)^2+m_{\boldsymbol{\omega}}}\}$. We take

$$\mathbf{p}_c^{(t+1)} = (1 + c_c)\mathbf{p}_c^{(t)} + h_\sigma\sqrt{c_c(2 - c_c)m_{\boldsymbol{\omega}}}\frac{\boldsymbol{\mu}_i^{(t+1)} - \boldsymbol{\mu}_i^{(t)}}{\sigma^{(t)}}$$

where $h_\sigma = 1$ if $\|\mathbf{p}_\sigma^{(t+1)}\| < \sqrt{1 - (1 - c_\sigma)^{2(t+1)}}(1.4 + 2/(D+1))E\|\mathcal{N}(0, \mathbf{I})\|$, and $h_\sigma = 0$ otherwise.

Exploration/exploitation trade-off. CMA-ES introduces another mechanism that commands exploration/exploitation trade-off. A large λ means a smaller selection pressure inside a single CMA-ES, which leads to a smaller adjustment in the covariance matrix. A small λ implies a higher selection pressure, but also a higher time of reaction if the distribution of the selected solutions changes considerably. The weights and the learning rate of a covariance matrix are two other parameters that fine tune the performance of a single CMA-ES. Note that in the current version of the algorithm, CMA-ES of each tree node evolves independently of other CMA-ES from other nodes. The interplay between different exploration/exploitation parameters is considered future work.

6 Experimental Section

In this section, we compare the performance of five algorithms on a set of benchmark real-parameter single objective problems [10]. Because MCTS algorithms solve maximization problems, the test problems were transformed from minimization to maximization problems.

Tested algorithms. We compare three algorithms.

- *tMCTS* is the top-down approach of real-parameter MCTS introduced in Sect. 4.2 where all the tree nodes have associated a UCB1 to select from their children
- *tCMA-MCTS* is the top-down approach of hierarchical CMA-ES solvers where all the tree nodes have associated a UCB1 to select from their children and a CMA-ES solver
- *CMA-ES* is the standard algorithm or a MCTS with a single node associated with a single CMA-ES solver

We have performed experiments also with the baseline real-parameter MCTS algorithm (*rMCTS*) from Sect. 4.1 with a single UCB1 strategy for the entire tree and with the baseline hierarchical CMA-ES solvers (*CMA-MCTS*) from Sect. 5 with a single UCB1 strategy for the entire tree but a CMA-ES for each tree node. But the results were not very different from the top-down approach of the homologous algorithms.

For each test function, the number of dimensions is $D = 10$, and the number of child nodes is 2^D, thus $K = 2$. Each function is a minimization function that is transformed into a maximization function by subtracting the maximum value for that function the function value for each solution. The interval in each dimension is $[-30.0, 30.0]$ and the interval of a leaf in MCTS is set to $\epsilon = 10^{-6}$.

The population size is $N = 2^6 = 64$. We have performed experiments with larger population sizes ($N = 100$ and $N = 256$) but the results deteriorate for all algorithms (except for CMA-ES) due to its low selection pressure.

All the algorithms run until a maximum of 10^5 individual solutions were generated, and the experiments were independently repeated for 30 times.

Results. To compare the performance of MCTS for real-parameter optimization, we consider measures for both optimization and multi-armed bandits: (i) the optimal individual solution found, (ii) the average of total solutions, (iii) the number of tree nodes, (iv) average estimated values of tree nodes, and (v) the entropy on the frequencies on selecting the tree nodes. To make the algorithms resemble even more CMA-ES, we only select the highest 50% percent of the population, $\lambda = 32$ that is generated each population. In fact, the best solutions were generated when only 5% percent of the generated solutions were used, meaning $\lambda = 2$.

In Table 1, we show that hierarchical CMA-ES finds a higher optimal solution than the other two algorithms, even though the average of the solutions for hierarchical CMA-ES is smaller than for real-coded MCTS. Note that the average per tree node is much smaller since many of the nodes would be visited only

Table 1. Performance of the tested algorithms.

Single node CMA-ES solver *CMA-ES*					
Function	Best sol	Mean sol	nr nodes	Mean nodes	Entropy
Rastrigin	7205 ± 1326	1123 ± 206	1.0 ± 0.0	1123 ± 6153	0.0 ± 0.0
SchafferF6	2.417 ± 0.02	0.0659 ± 1.427	1.0 ± 0.0	0.0696 ± 0.374	0.0 ± 0.0
Ackleys	24.518 ± 0.22	20.205 ± 0.002	1.0 ± 0.0	20.205 ± 0.002	0.0 ± 0.0
SchafferF7	0.094 ± 0.009	-0.333 ± 0.0	1.0 ± 0.0	-0.0318 ± 0.095	0.0 ± 0.0
Rosenbrock	7.618E8 ± 1.023E7	8.159E7 ± 108560	1.0 ± 0.0	8.158E7 ± 4.394E8	0.0 ± 0.0
Hierarchical CMA-ES solvers *CMA-MCTS*					
Rastrigin	8707 ± 372	1176 ± 352	12298 ± 2452	32.78 ± 176.56	5.36 ± 0.045
SchafferF6	5.62 ± 0.001	1.17 ± 0.017	22346 ± 1259	0.0329 ± 0.177	8.289 ± 0
Ackleys	37.34 ± 7.16	23.77 ± 4.49	10052 ± 3747	0.685 ± 3.69	4.34 ± 0.035
SchafferF7	0.111 ± 0.0	-0.08 ± 0.0	28547 ± 666	-0.0014 ± 0.0079	8.608 ± 0.0
Rosenbrock	7.784E8 ± 0.0	8.128E7 ± 4443732	15490 ± 3287	4298478 ± 2.31E7	5.42 ± 0.061
Real-parameter MCTS *rMCTS*					
Rastrigin	8374 ± 174	8193 ± 241	7094 ± 1135	84.56 ± 523.4	3.606 ± 0.0
SchafferF6	3.731 ± 0.157	0.59 ± 0.629	44808 ± 47	0.046 ± 0.103	9.764 ± 0.0
Ackleys	30.134 ± 1.016	25.78 ± 1.6	13660 ± 10024	0.338 ± 1.823	4.46 ± 0.004
SchafferF7	0.091 ± 0.009	-3.655 ± 0.0023	44686 ± 188	-0.0364 ± 0.195	9.735 ± 0.0
Rosenbrock	7.779E8 ± 252463	6.323E8 ± 985717	6730 ± 1270	8191552 ± 4.41E7	3.284 ± 0.0

few times in 10^5 generated solutions. The number of tree nodes generated with hierarchical CMA-ES is smaller, and the average of the generated tree nodes is smaller than for the real-coded MCTS.

7 Conclusions

We have proposed a real-parameter optimization paradigm that adapts Monte Carlo tree search (MCTS), generally acknowledged as an efficient method to find good solutions for very large and complex problems like playing games as GO, to maximize a real coded function. The multi-dimensional search space is iteratively decomposed in disjunct D-cubes, each D-cube representing a node in MCTS. The estimated value of each D-cube corresponds with the average value of the individual solutions generated in the D-cube. Two policies to select a node in the tree are investigated. Like in the standard MCTS, all nodes in a tree are considered at once for selection with a stochastic multi-armed bandit algorithm that selects often nodes with high estimated rewards and has a non-zero probability of selecting any node in the tree. The top-down approach assigns to each node a multi-armed bandits to select one of its children.

The policy used to generate new solutions is crucial for exploration/exploitation trade-off of the real-parameter MCTS. The simplest strategy to generate solutions use a uniform random distribution over a selected D-cube but the selection pressure, and thus the quality of the solution, is lower than when the solutions are generated with an adaptive multi-variate normal distribution, like in CMA-ES. Hierarchical CMA-ES solvers is a MCTS that uses a CMA-ES solver in each tree node.

The experimental results shows that the hierarchical CMA-ES solver outperforms both real-parameter MCTS with a uniform random distribution and a standard CMA-ES algorithm. We conclude that hierarchical CMA-ES solvers is an efficient algorithm doe to its efficient mechanism of generating solutions in D-cubes of different dimensions. We showed that, unlike the standard CMA-ES, the hierarchical CMA-ES solvers can work with a small population of solutions and bounded search spaces.

Acknowledgements. Madalina M. Drugan was supported by FWO project G.087814N "Multi-criteria RL".

References

1. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. J. Mach. Learn. Res. **3**, 397–422 (2002)
2. Auger, A., Hansen, N.: Tutorial CMA-ES: evolution strategies and covariance matrix adaptation. In: Genetic and Evolutionary Computation Conference, GECCO 2013, pp. 499–520 (2013)
3. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies: a comprehensive introduction. J. Nat. Comput. **1**(1), 3–52 (2002)
4. Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P.I., Rohlfshanger, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. IEEE Trans. Comput. Intell. AI Games **4**(1), 1–46 (2012)
5. Bubeck, S., Cesa-Bianchi, N.: Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. In: Foundations and Trends in Machine Learning, vol. 5 (2012)
6. Couetoux, A.: Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems. PhD thesis, Université Paris Sud - Paris XI (2013)
7. Drugan, M.M., Isasi, P., Manderick, B.: Schemata bandits for binary encoded combinatorial optimisation problems. In: Simulated Evolution and Learning - 10th International Conference (SEAL), pp. 299–310 (2014)
8. Igel, C., Hansen, N., Roth, S.: Covariance matrix adaptation for multi-objective optimization. Evol. Comput. **15**(1), 1–28 (2007)
9. Kocsis, L., Szepesvari, C.: Bandit based monte-carlo planning. In: Machine Learning: European Conference of Machine Learning (ECML) (2006)
10. Liang, J.J., Qu, B.Y., Suganthan, P.N., Chen, Q.: Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization. Technical Report Technical Report 201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China (2015)
11. Munos, R.: From bandits to monte-carlo tree search: the optimistic principle applied to optimization and planning. Found. Trends Mach. Learn. **7**(1), 1–129 (2014)

Multi-point Efficient Global Optimization Using Niching Evolution Strategy

Hao Wang(✉), Thomas Bäck, and Michael T.M. Emmerich

Leiden Institute of Advanced Computer Science, Leiden University,
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
{h.wang,m.t.m.emmerich,t.h.w.baeck}@liacs.leidenuniv.nl

Abstract. The Efficient Global Optimization (EGO) is capable of using limited function evaluation budget to find the global optimum. However, EGO is by design not built for parallelization, which is an important technique to speed up the costly computer codes. Some approaches have been developed to fix this issue. e.g. Constant Liar Strategy. In this article we propose an alternative way to obtain multiple points in the Efficient Global Optimization cycle, where a niching evolution strategy is combined into the classic EGO framework. The new approach is discussed and compared to other methods which aim at the same goal. The proposed approach is also experimented on the selected test functions.

Keywords: Global optimization · Expected improvement · Kriging · Niching evolution strategies

1 Introduction

Metamodelling or surrogate modelling is widely applied for tasks where the computer codes are expensive. The global optimization task arises naturally for the expensive computer codes, where the goal is to consume a limited number of computer code evaluations to locate the global optimal design.

In order to solve the global optimization task, the *Efficient Global Optimization* algorithm is proposed in [7], which is developed based on Kriging metamodel [8,13] and Expected Improvement criterion [10]. Kriging method has received the most attention among all the metamodels. If no noise presents in the computer codes, Kriging predictions are exact, meaning that the predictions perfectly match the observed data. Kriging method can be considered from Gaussian Process respective and offers a Kriging variance as prediction uncertainty measure. Expected Improvement exploits both the Kriging prediction and the variance to give a qualitative measure for the points in the search space.

Although EGO is quite successful to solve multi-model test problems [7], it also has some limitations. First, it only deals with single objective functions. However, in the real application, multiple objective functions are common cases. An EGO-based multiobjective optimization algorithm using $\mathcal{S}$-metric selection is

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_11

proposed in [12] to extend the capability of classical EGO algorithm. In addition, several multiobjective EGO algorithms are compared and analysed in [23].

In the real application, the parallelization of costly computer codes is a common way to accelerate the model evaluations. EGO algorithm by design can not fit into the parallelization framework because only one new point can be generated in each EGO iteration. To tackle this issue, the Multi-points Expected Improvement criterion (q-EI) is defined [17], aiming at providing multiple solutions to be parallelly evaluated. q-EI criterion is hard to compute exactly and the analytical expression is only derived for 2-EI [3]. A recent work manages to compute it exactly by Tallis formulas [1]. However, it is still difficult to maximize the q-EI criterion. Two strategies called Kriging Believer and Constant Liar are devised to approximately maximize the q-EI criterion [3].

This article aims at proposing an alternative way to obtain q points in each EGO iteration. However, instead of following the difficult q-EI maximization problem, we utilize a niching evolution strategy [19] to obtain q local maxima of EI criterion. The niching approach guarantees the q points obtained are diverse such that in the search space, those q points would possibly locate in the different basins of attraction.

This article is organised as follows: Sect. 2 briefly introduces all the background methods that we rely on and the related work. In Sect. 3, we discuss in detail our idea of applying niching evolution strategy into EGO framework and compare its similarities and differences to the Constant Lair Strategy. In Sect. 4, we provide the tested results from 6 selected test functions. Finally, Sect. 5 concludes the paper.

2 Background and Related Work

In this section, the basic idea of the Efficient Global Optimization and the techniques behind it: Kriging metamodel and Expected Improvement are briefly introduced. Prior to applying niching evolution strategy, its basic algorithmic framework is discussed here. We also introduce the Constant Lair Strategy which manages to maximize q-EI approximately in order to compare to our niching approach.

2.1 Efficient Global Optimization

The Efficient Global Optimization algorithm [7] (EGO) is founded on the Kriging interpolation method [8] and the Expected Improvement criterion [10]. It is designed for the case where function evaluation is expensive.

Kriging. It is a stochastic interpolation approach, which stems from earth science [8] and originally targets at mining problems. It has been widely used as a *metamodel* in the design and analysis of computer experiments [15,16], where the time-consuming computer model simulations are replaced by prediction from

the Kriging model trained beforehand. Note that Kriging method is also termed as Gaussian Process Regression [13] in the statistical machine learning literature.

Normally, in the design and analysis of computer experiments, Kriging method is trained on the some design sites $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ where n is the number of design sites. The observed response value corresponding to each of the design site is computed from an objective function $y(\cdot)$ and denoted as $\mathbf{y} = \{y(\mathbf{x}_1), \ldots, y(\mathbf{x}_n)\}$. Kriging method interpolates unknown sites by modelling the response values as a Gaussian Process (GP), which is completely defined by a prescribed mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$ [13, Chap. 2.2]:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[y(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(y(\mathbf{x}) - m(\mathbf{x}))(y(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned}$$

When the mean function is assumed to be constant and unknown, the method is called *Ordinary Kriging* (OK) on which the EGO algorithm is built. Then the response value can be express as [3]:

$$y(\mathbf{x}) = m + \varepsilon(\mathbf{x}),$$

where m is the constant but unknown mean function and $\varepsilon(\mathbf{x})$ represents a stationary GP with zero mean and covariance function $k(\cdot, \cdot)$.

Now we wish to predict $y^* = y(\mathbf{x}^*)$ at an unknown site $\mathbf{x}^*$. Without giving the derivation, the *posterior* conditional distribution on the observed response values is a Gaussian distribution and reads:

$$y^* | \mathbf{y} \sim \mathcal{N}\left(m_{OK}(\mathbf{x}), s^2_{OK}(\mathbf{x})\right).$$

The posterior conditional mean function $m_{OK}(\cdot)$ is used as the predictor and $s^2_{OK}(\cdot)$ measures the uncertainty of the prediction. The prediction variance is of high importance for the development of expected improvement. The detailed derivation and the expression for posterior mean and covariance function can be found in [3,13].

Expected Improvement. The simple optimum-seeking process on the Kriging model is not satisfactory because it pretends the Kriging model is exact and does not use the uncertainty (prediction variance) provided by the model. Sampling on sites possessing high uncertainty might also be promising because the real response value would be possibly better than the value predicted.

The Expected improvement [7] is the way to combine and balance between the approach to search for site with the good Kriging mean and the approach to search for the site with high Kriging variance. Given a minimization task, it is defined as:

$$\begin{aligned} EI(\mathbf{x}) &= \mathbb{E}[\max\{0, \min(\mathbf{y}) - y(\mathbf{x})\} \mid \mathbf{y}] \\ &= \int_{-\infty}^{\min(\mathbf{y})} (\min(\mathbf{y}) - u) \frac{1}{s_{OK}(\mathbf{x})} \phi\left(\frac{u - m_{OK}(\mathbf{x})}{s_{OK}(\mathbf{x})}\right) \mathrm{d}u, \end{aligned}$$

where $\phi(\cdot)$ denotes the density function of the standard normal distribution. It is a integration of the possible improvement at the site $\mathbf{x}$ and serves as the compromise between Kriging mean and variance. The expected improvement is increasing when Kriging variance increases and when Kriging mean decreases.

The Efficient Global Optimization. EGO [7] is proposed to efficiently solve global optimization based on EI criterion. It iteratively finds promising sample sites by maximizing the EI criterion. The sample sites generated are evaluated according to the objective function and inserted back into the Kriging training set. EGO then re-estimates the hyper-parameters of the Kriging model using the updated training set. The algorithmic framework is given in Algorithm 1.

Algorithm 1. Efficient Global Optimization

```
1 Generate design sites X and evaluate them: y = y(X)
2 Fit the Kriging model hyper-parameters (covariance function parameters) to the
  design sites.
3 while the stop criteria are not fulfilled do
4    Find global optimum of EI criterion: x' = argmax_x EI(x)
5    Evaluate x': y' = y(x') and add x', y' to X, y.
6    Re-estimate the Kriging model hyper parameters
7 end while
```

On the Branin function [7,17], EGO performs successfully [7] and iteratively finds all three global minima. Its main disadvantage is that the parallel evaluation is not possible because the sample generated in the next iteration depends on the current sample point. This limits its ability in the real application where computer simulations are extremely time-consuming but parallelable.

2.2 Multipoint EGO: Constant Liar Strategy

In order to allow for parallel function evaluations, the EGO must look for multiple search points at one iteration to update the Kriging model. The multi-point expected improvement criterion (q-EI) is proposed [17] as an generalization of EI to solve this task. Less formally, q-EI criterion involves the expectation of the smallest order statistic among a collection of correlated Gaussian random variables, which is neither easy to compute and thus nor easy to maximize. A computational feasible method called "Constant Liar" (CL) is suggested by David Ginsbourger et. al. [3] as an approximated q-EI maximization. CL strategy perform q stepwise sequential EI maximization in which the Kriging model is updated at each step without hyper parameter re-estimation. A constant value L set by the user is used as a "lie" to the true response value for every point found in the EI maximization. Three different L values are suggested in the first place: $\min\{\mathbf{y}\}, \text{mean}\{\mathbf{y}\}$ and $\max\{\mathbf{y}\}$. It has been observed that CL behaves in the way that the visited points generate a repulsion from each other [3].

2.3 Niching Evolution Strategies

In this section, the niching evolution strategies will be briefly introduced. In general, Evolutionary Algorithms (EAs) have the tendency to converge quickly into a single solution in the search space. However, in many problem solving scenario (e.g. global optimization), locating and maintaining multiple solution/optima is required. Niching is developed to achieve this goal by forming sub-populations in order to maintain the population diversity of EAs. The most successful techniques are *fitness sharing* [4] and *Crowding* [2].

Although initially niching was proposed mainly for Genetic Algorithms (GAs), it has also been introduced to classic $(1 \stackrel{+}{,} \lambda)$ self-adaptive Evolution Strategies by Ofer et. al [19]. Later, it is further developed for the derandomized ES [18] including the well-known Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [5] and finally evolved into a self-adaptive approach which allows for the niche radius and the niche shape adaptation [20]. In this paper, we will only focus on the $(1 \stackrel{+}{,} \lambda)$-niching evolution strategies with fixed niche radius and spherical niche shape.

Algorithm 2. $(1 \stackrel{+}{,} \lambda)$-Niching ES

```
1  Initialize D-set {D_i}_{i=1}^{q+p}
2  Set generation counter c ← 0
3  while the stop criteria are not fulfilled do
4     for i = 1 → (q + p) do
5        Generate λ mutations according to D_i
6     end for
7     Evaluate the population using fitness function
8     Obtain the dynamic peaks set DPS by performing Dynamic Peak Identification.

9     for p in DPS do
10       Set p as a new search point
11       Inherit the D-set from the parent of p and update the D-set accordingly.
12    end for
13    if N = size of DPS < q then
14       Generate N − q new search points and reset corresponding D-sets.
15    end if
16    c ← c + 1
17    if c  mod  κ = 0 then
18       Randomly re-generate (q + 1)^th ... (q + p)^th D-sets.
19    end if
20 end while
```

The $(1 \stackrel{+}{,} \lambda)$-niching evolution strategies works as follows: provided q optima expected to investigate, the niching procedure initializes $q + p$ so-called "D-sets" [21] which are evolution strategy kernels containing all the adapted strategy parameters (step-size, covariance matrix) as well as decision parameters (current search point/solution, mutation vectors). Each D-set defines the current search

point and all the internal information regarding an evolution strategy in a given time during the evolution. The q D-sets are meant for identifying q possible local optima/peaks while p D-sets are for "non-peak" domain, which are randomly regenerated every κ generations. The purpose of p "non-peak" D-sets is to explore the search space so that new niches would emerge and the probability of finding undiscovered optima is increased. The niching procedure then proceeds to generate λ offspring for each D-set. The population of $\lambda \cdot (q + p)$ offspring are evaluated according the fitness function. Using their corresponding fitness values, the selection of p search points is conducted based on *dynamic peak identification* (DPI) algorithm [9] based on a prescribed niche radius ρ. The functionality of DPI is to select a subset from the population in which each search point have a good fitness value and is not within the radius of the rest points. The selected p search points are considered as parental points for the next generation and their D-sets are inherited from their parents. Finally, the D-sets are updated based on the selected points. The process above is repeated until the termination criteria are satisfied. The algorithmic framework is listed in Algorithm 2.

3 Combining EGO with Niching

Considering EI criterion as an objective function, our approach is motivated by the observation that the EI landscape is usually highly multi-modal [7]. The original EGO algorithm aims at finding the global optimum of EI landscape, which is another global optimization task and thus hard to solve (actually solved by the exhaustive branch-and-bound method). EGO samples at the current global maximum of EI and updates the Kriging model such that the Kriging variances of sample and the region around it are reduced due to the Kriging property. This leads to a sudden decrease of landscape around the global optimum just found. Because of this, the local optimum before the change would possibly become new global optimum and thus will be visited in the next iteration.

Instead of using the sequential EI landscape changing to "expose" promising point, we propose to combine EGO with a niching method as an alternative, where the niching method is applied on the EI landscape. In one iteration, it is possible to use niching method to locate the global optimum of EI as well as some local optima that would be explored in the next few iterations in the classic EGO. The resulting algorithm will be called *q-EGO with niching*.

Furthermore, the niching method guarantees that the multiple points found are distinct, which means the chance that two points stay in the same basin of attraction is marginal. Intuitively, due to the region with high expected improvement possibly leads to a promising region in the objective search space, the combined algorithm is expected to locate and maintain multiple high performance regions in the objective search space at the same time.

From the niching perspective, q-EGO with niching can be considered as a meta-model assisted niching algorithm where the niche formation is performed

on the EI landscape instead of the real objective function. Therefore, the optimization procedure benefits from saving objective function evaluations especially when the evaluations are expensive.

Compared to the Constant Liar Strategy [3], q-EGO with niching is also expected to have a repulsion behaviour between points in one iteration, by the design of niching formation. However, the CL repulsion is caused by changing the EI landscape, which results in "hiding" the current point found from the optimization procedure. The CL repulsion behaviour is enhanced with increasing liar value L. The setting $L = \max\{\mathbf{y}\}$ and $L = \text{mean}\{\mathbf{y}\}$ leads to a space filling behaviour [3]. The behaviours of the niching approach and the CL strategy are further compared and visualized on the Himmelblau's function:

$$y_H(x_1, x_2) = (x_1^2 + x_2 + 11)^2 + (x_1 + x_2^2 - 7)^2$$

The Himmelblau's function has four global optima locating at $(3, 2)$, $(-2.81, 3.13)$, $(-3.78, -3.23)$ and $(3.58, -1.85)$ with global minimal value 0. The contours of it is shown in Fig. 1, left. In order to show that niching approach can maintain multiple distinct points, we choose four points to be generated in each iteration, which are expected to identify four global optima on the Himmelblau's function. The algorithm behaviour in four iterations are illustrated in Fig. 2.

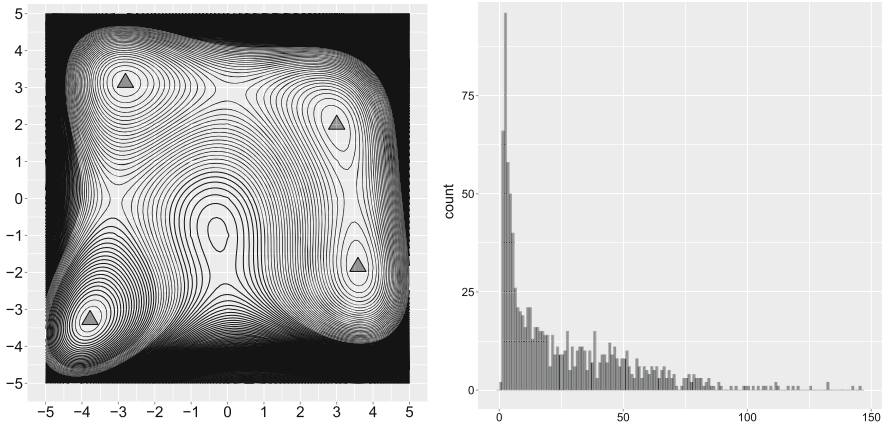


Fig. 1. Left: the contour lines of Himmelblau's function with four global optima marked by black triangles. Right: Histogram of q-EI criterion value measured on a 4-point EGO with niching through 10 iterations, which is test on the Himmelblau's function. The values are collected from 100 runs.

In Fig. 2, we compare the CL strategy with $L = \min\{\mathbf{y}\}$ (CL min) to q-EGO with niching in four consecutive iterations (from top to bottom). Both of the two approaches locate the four basins of attraction. They also explore the search space while keep tracking all the points found, showing a trade-off between exploitation and exploration. The difference is that the niching approach is not

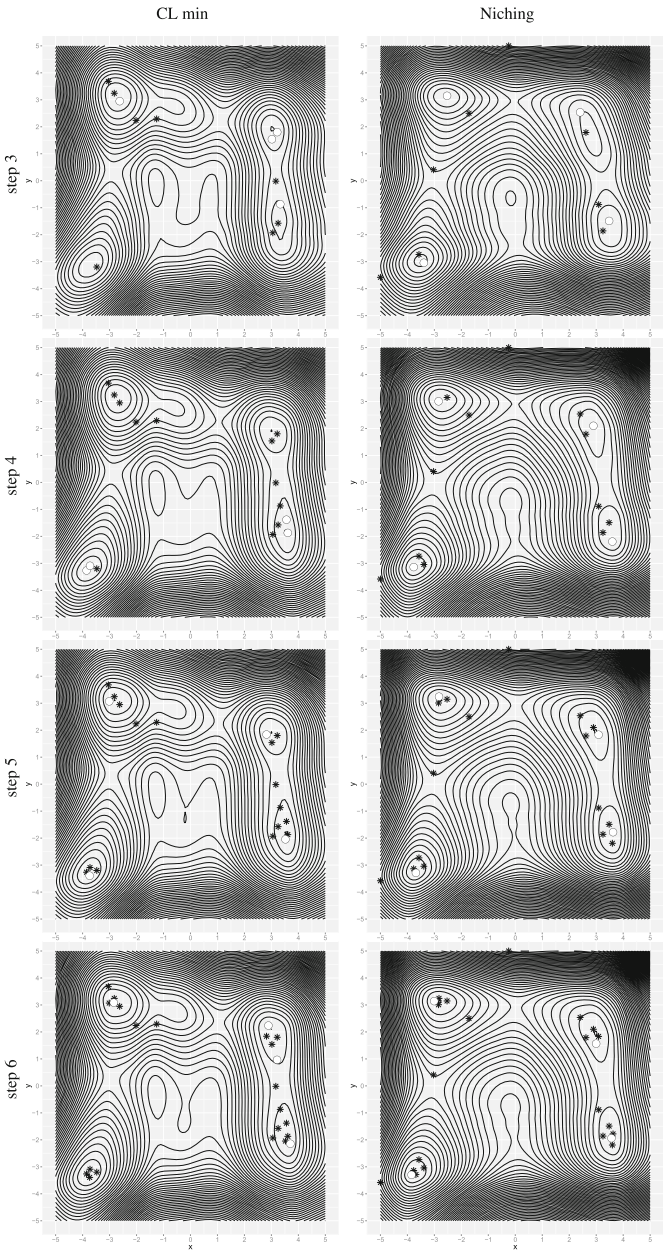


Fig. 2. On Himmelblau's function: 4-point EGO in four iterations. The white circles indicates the current sites found in each iteration while black stars show the sites sampled in history. Left: Constant Lair Strategy using $\min\{\mathbf{y}\}$. Right: The q-EGO with niching

likely to sample two point in one high performance region while CL min strategy would result in two (or even more) points explore the same region (see step 3 and 4 in the figure).

In addition, we also estimate the q-EI criterion values of points found from a 4-point EGO with niching on the Himmelblau's function. The estimation is conducted by Monte Carlo simulations of the q-EI formula [3]. The collection of q-EI values are shown in a Histogram in Fig. 1, right. In the following experiment section, the q-EI values of points provided by CL strategy and EGO with niching are compared.

For implementation issues, we choose the niching-DR2 [19] among many other niching evolution strategies to apply into EGO. It is the niching version of a so-called "the second derandomized evolution strategy" (DR2) [11]. We choose niching-DR2 because it is both simple to implement and converge fast. The parameters of niching-DR2, which are listed in Algorithm 2 are set as following: the function evaluation budget is set the be $10^3(q+p)$. κ that controls the frequency of sub-population resampling is set to 10. The niche radius ρ required by DPI procedure is computed as follow [19]:

$$\rho = \frac{r}{\sqrt[n]{q}}, \quad r = \frac{1}{2}\sqrt{\sum_{k=1}^{n}(x_{k,max} - x_{k,min})^2},$$

where n is the dimensionality and $x_{k,max}, x_{k,min}$ are the upper and lower bound of each coordinate in the search space. The termination criterion for the niching ES is current chosen as depletion of function evaluation budget.

4 Experiment

In this section, we test the q-EGO with niching and three Constant Lair variants: CL min, CL max and CL mix on a collection of test functions. The CL mix strategy [1] is a mixture of CL min and CL max in which two batches of points are generated from CL min and CL max and the batch of better q-EI value is provided to the Kriging model. For all the tests we use the DiceKriging and DiceOptim packages [14]. The experiment is in three parts: First we list the test functions selected. Then, the global convergence is compared among all the tested algorithms and finally the q-EI value of the points found are compared. All the algorithms are tested in 10 iterations and all the results are averaged over 100 runs. The reason to choose a small number of iterations as test run length is simple: on one hand, the classic EGO algorithm is capable of locating all the optima on several test functions [7] using 10 to 20 iterations. Thus, longer runs is not necessary. On the other hand, due to our observations, most of the space explorations and the Kriging model updates happen in the first 10 to 15 iterations on the 2-D functions, in which the EGO algorithm makes large progress.

4.1 Test Functions

We select 6 artificial multi-modal continuous functions and list them in the following:

- Transformed Hartman6 function is defined on $[0,1]^6$ and is a unimodal function. the original function is transformed by $-\log(-\text{Hartman6}(\mathbf{x}))$. This is the test-case where multipoint EGO with niching is expected to perform badly. We set $q = 3$ on this function.
- $\mathcal{M}$ is a hyper-grid multi-global function. Its global optima are uniformly distributed and have optimal value of -1. The function expression is listed below. We test the algorithms in 2-D where 10 minima are located in $[0,1]^2$.

$$\mathcal{M}(\mathbf{x}) = -\frac{1}{n}\sum_{i=1}^{n}\sin^{\alpha}(5\pi x_i).$$

  Note that n is the dimensionality and we choose $\alpha = 6$.
- Branin function is a multi-global function and a classical test-case in global optimization [7,17]. It is defined in 2-D with three global optima. The global minimal value is roughly 0.4.
- Rastrigin function [22] is a multi-modal function, which have only one global optimum and surrounded by number of local minima. The test is performed on 2-D. It has 6 optima in the space $[0, 1.5]^2$.
- Himmelblau's function is a multi-global function which is introduced previously in this paper. The search space is $[-5, 5]^2$.
- The well-known Ackley function is a multi-modal function and has only one global optimum. The global optimum has much lower value than the local optima.

We choose the number of points (q) generated in each iteration equals to the number of minima with two exception: on Hartman6 which is a unimodal function, we choose $q = 3$ and on Ackely function where the local optima increases exponentially with the increasing distance to the global optimum, we choose $q = 9$. For Hartman6 function, we would like to show that the q-EGO with niching would perform quite badly with $q > 1$ setting. We thus choose a moderate value $q = 3$. For Ackley with range $[-5, 5]^2$, there are 8 sub-optimal locations whose function values are the same, inferior to the global optimal but superior to the rest local optima. We would like to locate such sub-optima as well as the global one and thus choose $q = 9$.

4.2 Global Convergence Results

We use the relative mean squared error proposed in [1] to measure the convergence rate to the global optimum. It is defined as:

$$\text{rMSE}_i = \frac{1}{p}\sum_{k=1}^{p}\left(\frac{y_i^k - y^*}{y^*}\right)^2.$$

rMSE_i denotes the relative mean squared error at iteration i. p is the number of runs performed while y_i^k is the minimum value observed at iteration i in run number k. Note that we translate some of the test function optimal value to prevent 0 in order to calculate the rMSE. The relative mean square error on each test function are shown in Fig. 3. Note that rMSE is scaled by $\log 10$.

On the unimodal function Hartman6, the results of 3-points EGO show that the q-EGO with niching performs much worse that any variants of CL strategy. This is the expected behaviour because the niches formed on the EI landscape of Hartman6 does not maps to any local optima and the niching method is performing space-filling using all three niches.

On the multi-modal Rastrigin function, q-EGO with niching actually outperforms all the CL variants. On the Branin function, q-EGO with niching performs equally with CL max in the first three iterations and make a large acceleration from the four iteration. On the $\mathcal{M}$ function, q-EGO niching works much worse in the first 6 iteration and accelerates again in the later iterations. On Rastrigin function, the same behaviour has shown. We consider the reason is that initially the Kriging prediction response surface may differs from the real landscapes drastically such that the niches formed on the EI landscape do not map to any high performance region. After the model is kept updating for some iterations, the local optima of objective function would possibly create local optima in the EI landscape, which attracts the niches.

On the Himmelblau's function. q-EGO with niching is the worst one initially and finally catch the CL mix from iteration 8. On the Ackley function, q-EGO with niching performs roughly the same with the CL min Strategy and outperform both CL min and CL mix after iteration 5.

The rMSE values measured in the last iteration of the test (step 10) are also summarized in Fig. 4 as box plots. In addition to the mean values reported in Fig. 3, it shows the median, the first and third quartiles and outliers.

In general, convergence plots suggest that initially the q-EGO with niching performs worse or equally to the CL variants and accelerates the convergence after some iterations of Kriging model updates. Furthermore, such behaviour may even suggest a possible mixture approach where the CL strategy is applying in the beginning and then the algorithm is switched to niching method to gain from the acceleration.

4.3 q-EI of Search Points

The average q-EI value of the points generated in each iteration are computed by Monte Carlo simulations [3] and shown in Fig. 5. The plotted values are scaled by $\log 10$. All the average q-EI values decreases with respect to increasing iterations, as expected. There is no clue who is the winner in general. Focusing on the first 4 iterations, q-EI values of the search points found by the niching approach are roughly smaller than the q-values by CL variants on Hartman6, $\mathcal{M}$ and Ackley function. On Rastrigin and Himmelblau function, the proposed algorithm gives higher q-EI values in the middle (iteration $4, 5, 6$) of the test. On the Branin

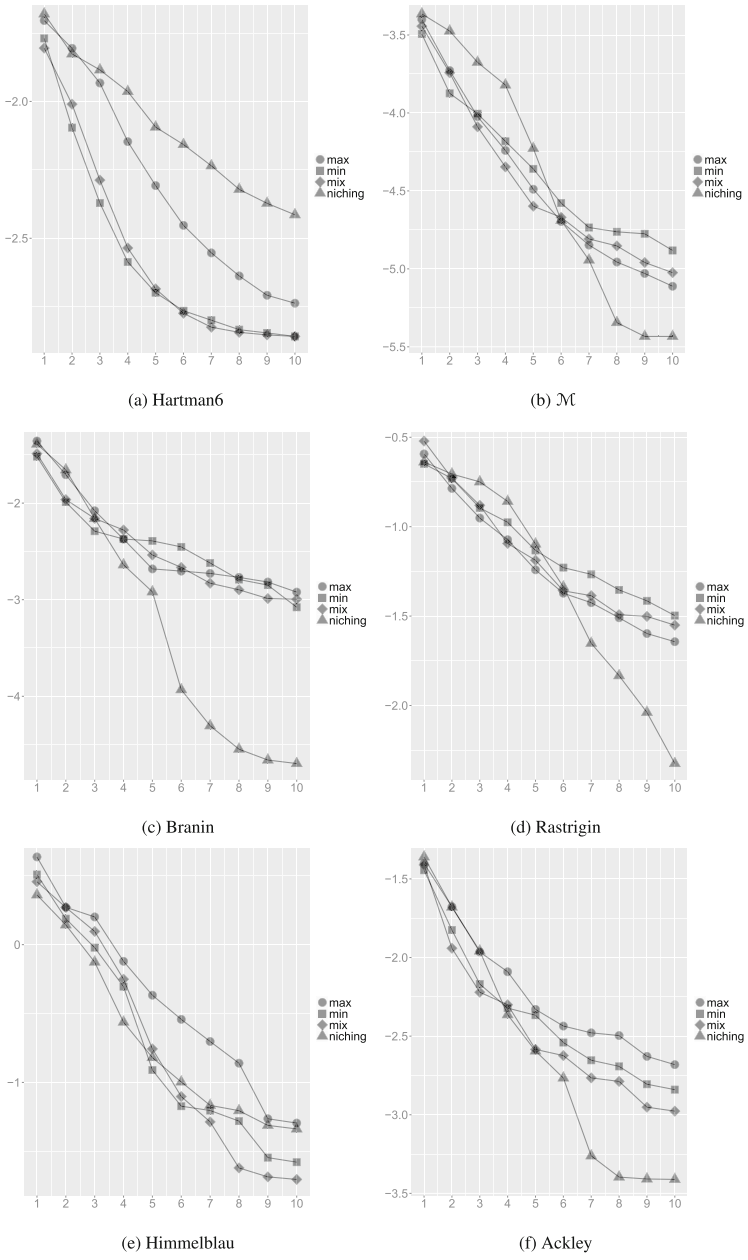
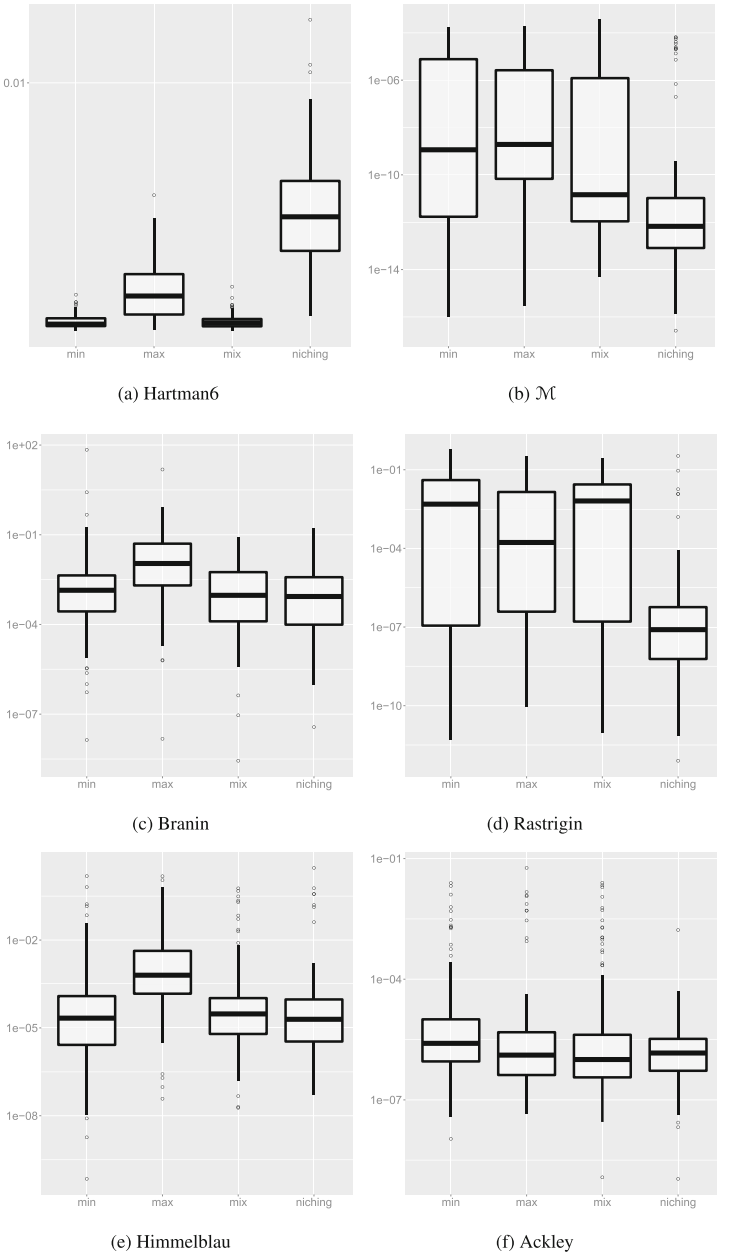


Fig. 3. Convergence to the global optimum of tested algorithm. square: CL max, circle: CL min, diamond: CL mix, triangle: q-EGO with niching. The convergence is measure by relative mean square error to the global optimal value. The x-axis is the iteration steps. The y-axis is $\log 10(\text{rMSE})$.



(a) Hartman6 (b) $\mathcal{M}$

(c) Branin (d) Rastrigin

(e) Himmelblau (f) Ackley

Fig. 4. Box-plot of rMSE values measured in the last iteration. The upper and lower "hinge" correspond to the first and third quartiles (the 25th and 75th percentiles). The thick black line inside box shows the median. The outliers are plotted in circles. The upper whisker extends from the hinge to the highest value that is within 1.5 * IQR of the hinge and the lower whisker extends from the hinge to the lowest value within 1.5 * IQR of the hinge, where IQR is the inter-quartile range.

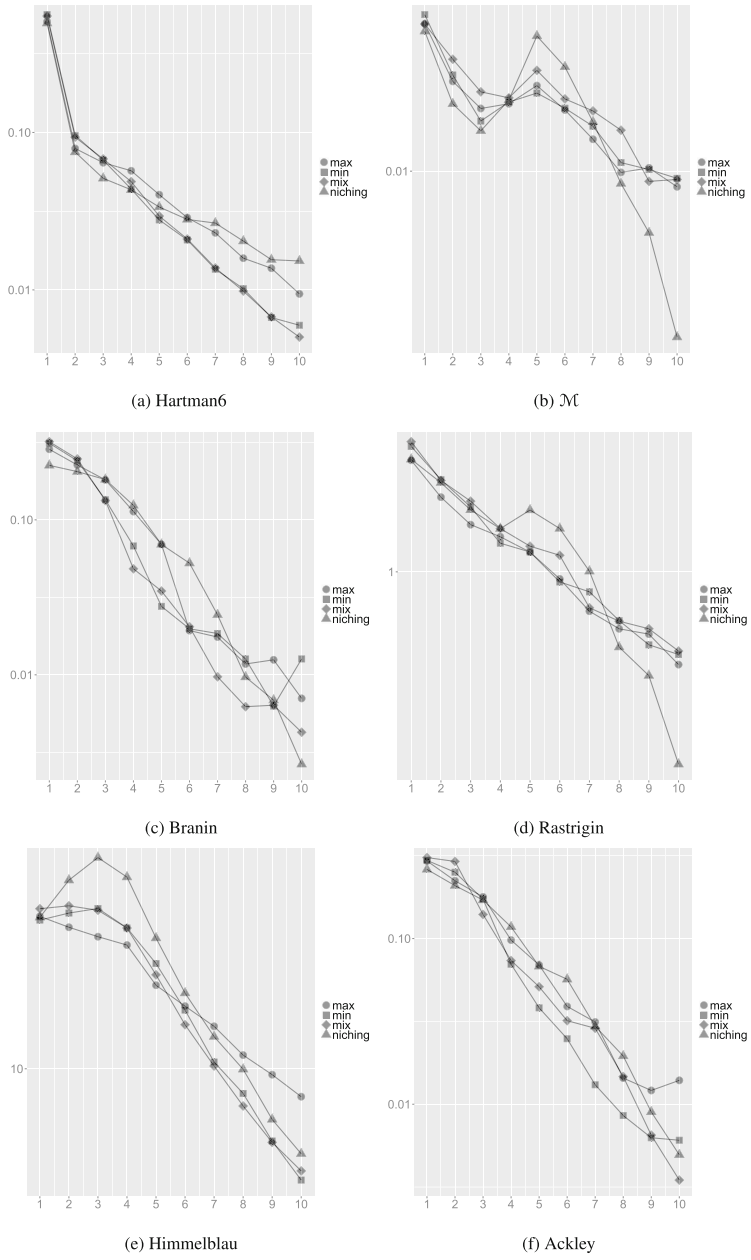


Fig. 5. Average q-EI value of points found by tested algorithm. square: CL max, circle: CL min, diamond: CL mix, triangle: q-EGO with niching. The q-EI criterion is approximated by Monte Carlo simulations. The x-axis is the iteration steps. The y-axis is average q-EI measured.

function, the differences are not significant. In general, the q-EGO with niching shows a much faster q-EI value reduction when the iteration goes upper than 8.

5 Conclusion and Further Works

In this article, we first briefly review the Efficient Global Optimization algorithm and the q-EI criterion aiming at the parallelization of the function evaluations. The strategy to approximate q-EI maximization, Constant Liar is also briefly discussed and compared the our proposed approach. We discuss the niching evolution strategy framework and the combination of EGO and niching ES in detail. The proposed method, q- EGO with niching and CL together are tested on 6 selected functions. Both the convergence to the global optimum and the q-EI measured on the search point are summarized. In general, the q-EGO with niching works a little bit worst than the CL strategies at the first place of the optimization process while it shows a significant acceleration of convergence after some iterations.

The parameter q that determines the number of the niches, is roughly set according to the number of the local optima in the search space. The effects of varying the q setting on the algorithm performance is yet unclear and should be investigated in the further, which is of high importance because the number of the local optima is usually not accessible in the real application.

As suggested by the results on convergence, it might be possible to mix niching ES with CL strategy. By using the CL strategy initially and switch to the niching method, the algorithm would benefit from both.

The time-complexity of q-EGO with niching is determined by the function evaluation budget since it is completely consumed. We did not compare the time-complexity to the CL strategy and thus is left for the further research.

As mentioned before, the q-EGO niching can also be viewed as a meta-model assisted niching approach. Therefore, it is reasonable to compare its performance to other classical niching evolution algorithm.

In the last, we should also consider how to adopt the multi-point EGO with niching algorithm to the multi-objective optimization problems. A recent proposal of multi-points for parallel evaluation in the multi-objective optimization problem can be found in [6].

Acknowledgements. The authors gratefully acknowledge financial support by the Netherlands Organisation for Scientific Research (NWO) within the project PROMI-MOOC.

References

1. Chevalier, C., Ginsbourger, D.: Fast Computation of the multi-points expected improvement with applications in batch selection, pp. 59–69. Springer, Heidelberg (2013)
2. De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, Ann Arbor, MI, USA, (1975). AAI7609381

3. Ginsbourger, D., Le Riche, R., Carraro, L.: Kriging is well-suited to parallelize optimization. In: Computational Intelligence in Expensive Optimization Problems, pp. 131–162. Springer (2010)
4. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, pp. 41–49. Lawrence Erlbaum, Hillsdale (1987)
5. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
6. Horn, D., Wagner, T., Biermann, D., Weihs, C., Bischl, B.: Model-based multi-objective optimization: taxonomy, multi-point proposal, toolbox and benchmark. In: Evolutionary Multi-Criterion Optimization, pp. 64–78. Springer (2015)
7. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998)
8. Krige, D.G.: A statistical approach to some basic mine valuation problems on the witwatersrand. J. Chem. Metall. Min. Soc. S. Africa **52**(6), 119–139 (1951)
9. Miller, B.L., Shaw, M.J.: Genetic algorithms with dynamic niche sharing for multimodal function optimization. In: 1996 Proceedings of IEEE International Conference on Evolutionary Computation, pp. 786–791. IEEE (1996)
10. Mockus, J.B., Mockus, L.J.: Bayesian approach to global optimization and application to multiobjective and constrained problems. J. Optim. Theor. Appl. **70**(1), 157–172 (1991)
11. Ostermeier, A., Gawelczyk, A., Hansen, N.: Step-size adaption based on non-local use of selection information. In: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature, PPSN III, pp. 189–198, Springer, London (1994)
12. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection. In: Parallel Problem Solving from Nature–PPSN X, pp. 784–794. Springer (2008)
13. Rasmussen, C.E., Williams, C.K.: Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning. MIT Press, Cambridge (2006)
14. Roustant, O., Ginsbourger, D., Deville, Y.: DiceKriging, DiceOptim: two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. J. Stat. Softw. **51**(1), 1–55 (2012)
15. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. **4**(4), 409–423 (1989)
16. Santner, T.J., Williams, B.J., Notz, W.: The Design and Analysis of Computer Experiments. Springer Series in Statistics. Springer (2003)
17. Schonlau, M.: Computer Experiments and Global Optimization. University of Waterloo (1998)
18. Shir, O.M., Bäck, T.: Dynamic niching in evolution strategies with covariance matrix adaptation. In: The 2005 IEEE Congress on Evolutionary Computation, 2005, vol. 3, pp. 2584–2591. IEEE (2005)
19. Shir, O.M., Bäck, T.: Niching in evolution strategies. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, pp. 915–916. ACM (2005)
20. Shir, O.M., Emmerich, M., Bäck, T.: Self-adaptive Niching CMA-ES with mahalanobis metric. In: IEEE Congress on Evolutionary Computation, 2007, CEC 2007, pp. 820–827. IEEE (2007)

21. Shir, O.M.: Niching in derandomized evolution strategies and its applications in quantum control. Natural Computing Group, LIACS, Faculty of Science, Leiden University (2008)
22. Zhigljavsky, A., Žilinskas, A.: Stochastic global optimization, vol. 9. Springer (2007)
23. Zaefferer, M., Bartz-Beielstein, T., Naujoks, B., Wagner, T., Emmerich, M.: A case study on multi-criteria optimization of an event detection software under limited budgets. In: Evolutionary Multi-Criterion Optimization, pp. 756–770. Springer (2013)

Community Detection in NK Landscapes - An Empirical Study of Complexity Transitions in Interactive Networks

Asep Maulana[1(✉)], André H. Deutz[1], Erik Schultes[1,2], and Michael T.M. Emmerich[1]

[1] LIACS, Leiden University, Niels Bohrweg 1, 2375 CA Leiden, The Netherlands
e.a.schultes@liacs.leidenuniv.nl
[2] Human Genetics Department, Leiden Center for Data Science, Leiden University Medical Centre, 111 Snellius Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
http://moda.liacs.nl

Abstract. NK-landscapes have been used as models of gene interaction in biology, but also to understand the influence of interaction between variables on the controllability and optimality organizations as a whole. As such, instead of gene networks they could also be models of economical systems or social networks. In NK-landscapes a fitness function is computed as a sum of N trait values. Each trait depends on the variable directly associated with the trait and K other variables – so-called epistatic variables. With increasing number of interactions the ruggedness of the function increases (local optima). The transition in complexity resembles phase transitions, and for $K \geq 2$ the problem to find global optima becomes NP hard. This is not the case if epistatic variables are local, meaning that all interactions are local.

In this research we study NK-landscapes from the perspective of communities and community detection. We will not look at communities of epistatic links but instead focus on links due to correlation between phenotypic traits. To this end we view a single trait as an individual agent which strives to maximize its contributed value to the net value of a community. If the value of a single trait is high whenever that of another trait is low we regard these traits as being conflicting. If high values of one trait coincide with high values of other traits we regard the traits as supporting each other. Finally, if the value of two traits is uncorrelated, we view their relationship as being neutral. We study what happens to the system of traits when the NK-landscape undergoes a critical transition to a more complex model via the increment of k. In particular, we study the effect of locality of interaction on the shape and number of the emerging communities of traits and show that the number of communities reaches its lowest point for medium values of k and not, as might be expected, for a fully connected epistatic matrix (case $k = N - 1$).

Keywords: NK-landscapes · Community detection · Complex networks · Locality · Landscape analysis

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_12

1 Introduction

NK-Models (or NK-Landscapes) were introduced in Kauffman and Levin [8] as models of how the fitness of an organism is related to gene interaction. They also gained popularity in the study of complex organizations (see Anderson [3]) and innovation networks (see Frenken [6]).

In the classical NK-model the parameter N describes the number of components in a network (genes, agents, or nodes in an distributed system) and each component is associated with a control variable x_i and a trait function or output value f_i. k describes the degree of interaction between these components. For each component of the system ($i \in \{1, \ldots, N\}$), the value of f_i depends on the value of the variable x_i that is associated with it and k other variables $x_{e(i,1)}, \ldots, x_{e(i,k)}$. These k variables are called the *epistatic variables*. The fitness of the NK-landscape is the sum of these values:

$$F(x_1, \ldots, x_N) = \sum_{i=1}^{N} f_i(x_i, x_{e(i,1)}, ..., x_{e(i,k)}) \tag{1}$$

In Eq. 1 the values of $E := e(i,j), i \in \{1, \ldots, N\}, j \in \{1, \ldots, k\}$ is the epistatic matrix that determines which variable interacts with which other variable.

Based on the locality of the epistatic matrix two variants of the NK-landscapes are distinguished: The epistatic variables of gene i can be *adjacent* with respect to the index i (local interactions) or their choice can be *random* (global interactions). Note, that in case of adjacent epistatic genes the indices are mapped cyclically, i.e. gene N is a direct neighbor with gene 1 (wrap around). This way every gene has two direct neighbors. If more than two epistatic genes need to be defined we collect the genes in an increasingly big radius (2-step neighbors, 3-step neighbors and so on). The notion of neighborhood stems here from the idea of physical location on a DNA, that, for the sake of simplicity is viewed as a ring.

Figure 1 shows a visualization of the epistasis structure that arises based on these choices.

We can now introduce a concrete realization of a function F, for instance by using the classical binary NK-landscape where each variable can only obtain the values 0 and 1. For this we will define 'upper case' F_i component functions that will accept a bit string the components of which correspond to the values of x (Goedel encoding of the vector):

$$F(x_1, \ldots, x_N) = \sum_{i=1}^{N} F_i(2^0 x_i + 2^1 x_{e(i,1)} + \cdots + 2^k x_{e(i,k)}), \quad \mathbf{x} \in \{0,1\}^N \tag{2}$$

Each one of the functions F_i is looked up in a table of size 2^{k+1}. Each of these tables comprises 2^{k+1} random numbers that are sampled from a uniform distribution in [0,1] (see, e.g. [2]). This was done in order to make the model simple and to not introduce additional complexity in its construction [8]. Subsequent

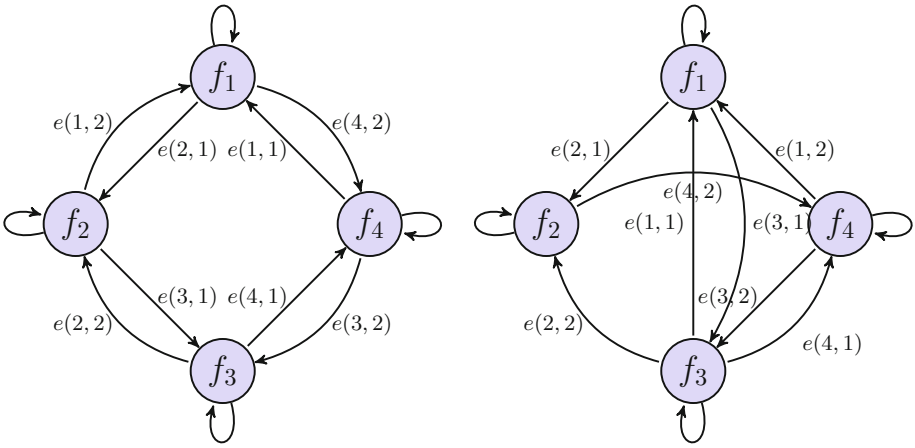


Fig. 1. (Left hand side) Example of an NK-Landscape epistasis network for $N = 4$, $k = 2$ and adjacent epistatic genes. (Right hand side) Example of an NK-Landscape epistasis network for $N = 4$, $k = 2$ and randomly assigned epistatic genes. The arrows labeled with $e(i, j) \in \{1, \ldots, N\}$ indicate the epistatic genes that influence the gene with index $i \in \{1, \ldots, N\}$ for $j \in \{1, ..., k\}$.

analysis revealed interesting, emergent behavior of NK-Landscapes already on this very basic level. One interesting feature of NK-landscapes is that their properties critically depend on the choice of k. Some interesting properties that depend on the choice of k are summarized in the following list (see also [2]):

- $k = 0$ (no epistasis):
  - The problem is separable.
  - There exists a unique global and local optimum.
- $k = 1$
  - A global optimum can be found in polynomial time.
- $k \geq 1$
  - Adjacent epistatic genes: Time complexity is in $O(N^k)$.
  - Randomly assigned epistatic genes: Finding global optimum becomes NP complete; time complexity is in $O(2^N)$ under the assumption that $P \neq NP$.
- $k = N - 1$
  - Random function value assignment; causality is lost and finding the global optimum takes $\Omega(2^N)$ time.

An interesting research question is: What happens to the structure of the problem at the critical transitions from simple ($k = 0$), via polynomially optimizable $k = 1$ or problems with adjacent epistatic genes for greater k, to complex networks $k = 2$ (global interactions). And how does this differ from complete random function value assignments at the level of $k = N - 1$.

In order to study this we propose to study in more depth the interaction between the components of F, namely the functions F_i. A new perspective to look at this question is to view the F_i as objective functions in a many objective optimization problem. The novel perspective taken is to view these trait functions as objective functions that seek to contribute to F with the highest possible contribution, or, that seek to obtain the best adaptive value. This yields a multiobjective optimization problem, which can be written as:

$$F_1(\mathbf{x}) \to \max, \ldots, F_N(\mathbf{x}) \to \max, \quad \mathbf{x} \in \{0,1\}^N \tag{3}$$

The correlation between trait functions can be determined if the input vector is viewed as a random sample. Different trait functions F_i can support each other (positive correlation), be neutral with respect to each other (zero correlation), or conflict with each other (negative correlation).

It is noted that the maximization of each single component function takes time $\Omega(2^k)$ due to the random assignment process. By introducing interaction the complexity of the optimization task grows. So far it is unknown what exactly happens at the transition from binary to ternary interactions, that is from from polynomially time solvable to NP complete problems, and we hope that correlation and community analysis will shed some new light on this.

In summary the contributions of our work will be as follows. In order to better understand the transitions in complexity in NK-landscapes from the perspective of communities of component functions we will

1. visualize the community structure among the different F_i trait functions using state-of-the-art algorithms from community detection
2. provide statistics on number of communities and modularity for different values of k
3. discuss correlation and squared correlation as a measure of connectedness in both adjacent and random NK-landscapes

In the following Sect. 2 we will discuss basic concepts in community detection. Then, in Sect. 3 the approach will be discussed in detail, i.e. how to perform community detection among the different F_i trait functions, and how to use the squared correlation measure for global statistics. The results will be discussed in Sect. 4, followed by a brief summary and outlook (Sect. 5).

2 Community Detection

In community detection it is the goal to identify subgraphs that are densely connected (communities) and separate them from subgraphs with members that have less strong links to the community. A measure for how well a community belongs together is the modularity of a community. The modularity is high if there are many links with height among members of the community and only few and low weighted links to members outside the community. Besides maximization of modularity one might also consider multidimensional scaling as a means to

find communities. This method interprets link weights as distances and then uses distance-based clustering approaches to find communities.

Originating from social science, community detection is nowadays also used in many other network related problems, such as protein-protein-interaction networks and telecommunication networks.

Two common methods for community detection are the Louvain method [5] and the VOS method [10]. Both methods have been implemented in the Pajek software tool [4].

The Louvain[1], for community detection algorithm is an algorithm for performing community detection (clustering) in networks by heuristically maximizing a modularity function. The Louvain algorithm can be used to detect communities in very large networks within short computing times. On the other hand, VOS clustering algorithm is often used as an algorithm for community detection in networks. The difference is that VOS clustering is based on the betweenness centrality. VOS is closely related to multidimensional scaling in so far as the distance between objects is reflected in the planar visualization.

For details on the community detection algorithms the reader is advised to follow the references [5] and, respectively, [10].

3 Approach

The graph that we consider has the N nodes. Each node is associated with a component function F_i. Links between the nodes are weighted by the correlation between the two function values.

Let $(\Omega, \mathcal{S}, P)$ denote a probability space, where $\mathcal{S}$ is the event space and Ω denote the set of elementary outputs – here chosen as the input space $X = \{0,1\}^N$. We will only consider singletons as events and write ω instead of $\{\omega\}$. In the following we consider the entire input space $X = \{0,1\}^N$, and a uniform distribution over this set. For each trait function F_i the random variables $\mathcal{F}_i : \Omega \to \mathbb{R}$ are defined as $\mathcal{F}_i : \omega \mapsto F_i(\omega)$. Next, consider a sample $\Omega' \subseteq \Omega$ and the realizations of random variables $\mathcal{F}_1(\omega), \ldots, \mathcal{F}_m(\omega)$ for $\omega \in \Omega$. Now, for the group of paired evaluations of $\mathcal{F}_i$ and $\mathcal{F}_j$ the empirical correlation coefficient can be computed by means of

$$\rho^e_{ij} = \frac{\frac{1}{1-|\Omega_s|}\sum_{\omega\in\Omega_s}(\mathcal{F}_i(\omega)-\overline{\mathcal{F}_i})(\mathcal{F}_j(\omega)-\overline{\mathcal{F}_j})}{\sqrt{\frac{1}{1-|\Omega_s|}\sum_{\omega\in\Omega_s}(\mathcal{F}_i(\omega)-\overline{\mathcal{F}_i})}\sqrt{\frac{1}{1-|\Omega_s|}\sum_{\omega\in\Omega_s}(\mathcal{F}_j(\omega)-\overline{\mathcal{F}_j})}}$$

Now, ρ^e_{ij} serves as an estimate of the correlation ρ_{ij} between $\mathcal{F}_i$ and $\mathcal{F}_j$. The value of the correlation ρ_{ij} ranges from perfect anti correlation (-1), via independence (0), to perfect correlation $(+1)$.

In the context of optimization of the F_i functions, we can interpret this correlation as follows:

[1] Named after the university of Louvain-de-Neuve in Belgium where this method was originated.

- Positively correlated trait functions can be interpreted as trait functions that support each other. This means that maximizing one function will imply that also the other function will obtain high values.
- Uncorrelated trait functions are considered to be independent of each other. They can be maximized in isolation from each other. The NK-landscape value can be maximized by independently maximizing these traits.
- Negatively correlated objective functions are in a strong conflict with each other, that is increment of one value typically will lead to a deterioration of the other value. Intuitively, one might expect that if there are many conflicting trait functions the maximization of the NK-landscapes gets more difficult.

The next two examples of a correlation matrix of NK-Landscape for $k = 0$ and $k = 1$ with $N = 10$ are provided with Tables 1 and 2 respectively. Note, that in case $k = 0$ not all correlations between different trait functions are 0, although from the construction of the NK-Landscape all trait functions should be independent in the case $k = 0$. The finiteness of the sampling space makes it however unlikely that also the statistical correlation is exactly the same. For $k = 1$, clearly some correlations have values higher than zero and higher than

Table 1. Correlation coefficient matrix for N = 10 and k = 0

1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	1.0000	−0.0135	0.0128	−0.0143	0.0115	−0.0115	−0.0322	0.0000
0.0000	0.0000	−0.0135	1.0000	0.0128	−0.0143	0.0115	−0.0115	−0.0322	0.0000
0.0000	0.0000	0.0128	0.0128	1.0000	−0.0428	0.0345	−0.0345	−0.0966	0.0000
0.0000	0.0000	−0.0143	−0.0143	−0.0428	1.0000	0.0576	−0.0576	−0.1612	0.0000
0.0000	0.0000	0.0115	0.0115	0.0345	0.0576	1.0000	−0.1279	−0.3576	0.0000
0.0000	0.0000	−0.0115	−0.0115	−0.0345	−0.0576	−0.1279	1.0000	0.3576	0.0000
0.0000	0.0000	−0.0322	−0.0322	−0.0966	−0.1612	−0.3576	0.3576	1.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

Table 2. Correlation coefficient matrix for N = 10 and k = 1

1.0000	−0.2366	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	−0.2463
−0.2366	1.0000	0.3247	0.0149	0.0132	−0.0105	0.0097	0.0108	0.0304	0.0000
0.0000	0.3247	1.0000	0.4545	−0.0047	0.0037	−0.0035	−0.0039	−0.0109	0.0000
0.0000	0.0149	0.4545	1.0000	−0.0334	0.0197	−0.0183	−0.0203	−0.0573	0.0000
0.0000	0.0132	−0.0047	−0.0334	1.0000	−0.0648	0.0394	0.0439	0.1236	0.0000
0.0000	−0.0105	0.0037	0.0197	−0.0648	1.0000	−0.3341	−0.0653	−0.184	0.0000
0.0000	0.0097	−0.0035	−0.0183	0.0394	−0.3341	1.0000	0.8035	0.3192	0.0000
0.0000	0.0108	−0.0039	−0.0203	0.0439	−0.0653	0.8035	1.0000	0.3549	0.0000
0.0000	0.0304	−0.0109	−0.0573	0.1236	−0.184	0.3192	0.3549	1.0000	0.0000
−0.2463	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

all correlations for the previous case. As we will see, this strong correlation will vanish again quickly for values of $k \gg 1$.

In summary correlation analysis is applied to derive the weights of links between the pairs of component functions. The $N \times N$ correlation matrix is interpreted as a weighted graph with weighs in $[-1, 1]$. This way, it can be analyzed using graph theoretic algorithms and in particular by community detection. Thereafter, statistics on macroscopic properties of the community graphs can be applied to find regularities that might reveal new insights in the critical transition(s) of the landscape's complexity as k grows.

4 Results

Results depicted in Figs. 4, 5, 6, 7 and 8 visualize the concrete results of the community detection obtained and visualized with `Pajek` using Louvain and, respectively, VOS clustering. First, let us summarize results for the Louvain method. Figure 4 and respectively, Fig. 5 show the transition of community structures for randomly assigned genes and, respectively, for adjacent epistatic genes. The value of k is chosen from 0 to $N - 1$.

From the visual impression it is clear that the highest degree of separation is obtained in the case $k = 0$ for both epistatic link structures (random, adjacent).

The nodes that belong to the same community are indicated by nodes which share the same color. The number of communities reaches its minimum for $k = 2$. A confirmation of this can be obtained when plotting the number of communities over different values of k, which is done in Fig. 2 for randomly assigned epistatic genes and in Fig. 3. The value of $k = 2$ is a clear optimum in all cases.

This is surprising, because it might be expected that for $k = N - 1$ all nodes merge to one big community. This is not the case and – in first approximation

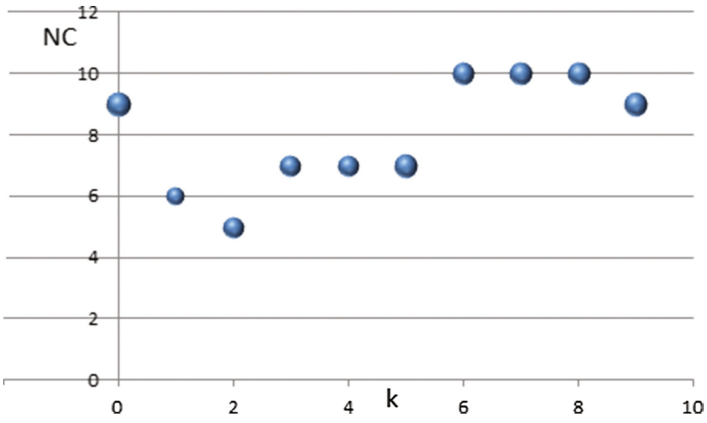


Fig. 2. Community detection by Louvain clustering algorithm based on randomly assigned epistatic genes.

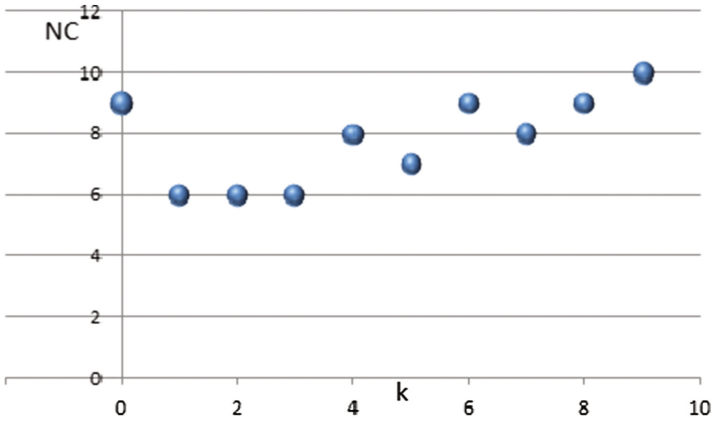


Fig. 3. Community detection by Louvain clustering algorithm based on adjacent epistatic genes.

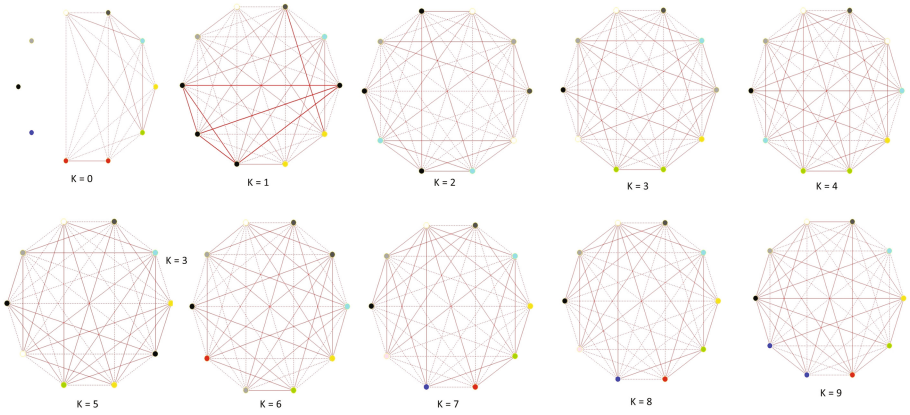


Fig. 4. Community detection by Louvain clustering algorithm based on randomly assigned epistatic genes.

– might be explained by the fact that nodes can also negatively influence each other (conflicting nodes). Note, that the number of communities grows at a slower rate for adjacent epistatic genes. This coincides with the slower increase of computational complexity [11].

Analogous findings have been made with the VOS clustering method for community detection. Figures 7 and 8 show the community structures, whereas Figs. 9 and 10 show the results. The correspondence between the two different approaches for community detection underpin that the findings are not an artifact of the method. More dissonance between Louvain and VOS methods is found for the number of communities for high levels of k. However, in all methods the

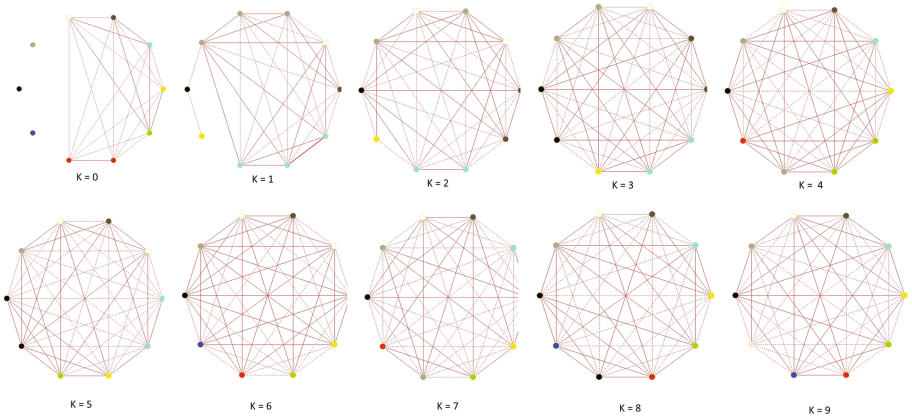


Fig. 5. Community detection by Louvain clustering algorithm based on Neighborhood selection with adjacent epistatic genes.

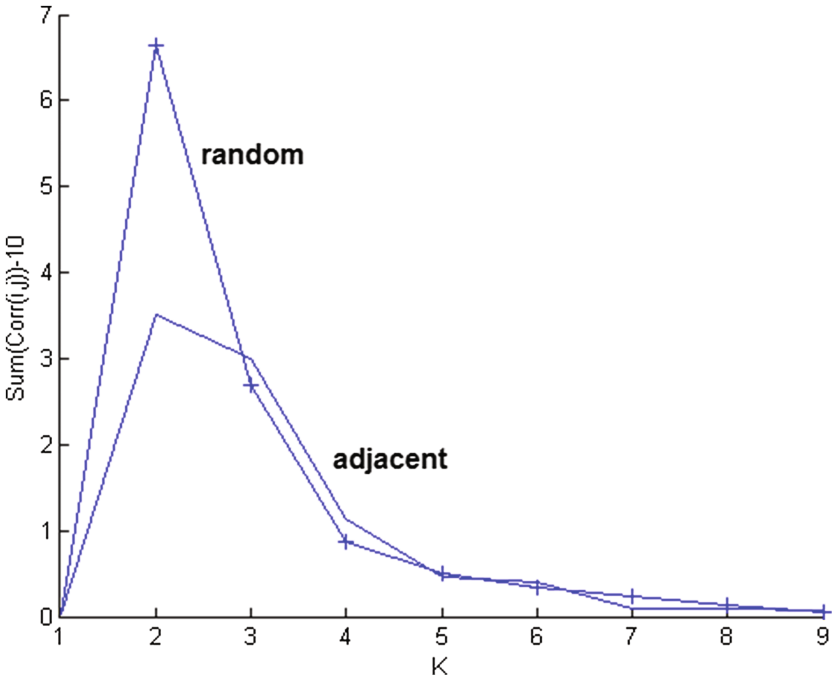


Fig. 6. Comparison of correlation between a node in the network community (as result of community detection), based on randomly assigned epistatic genes and adjacent epistatic genes.

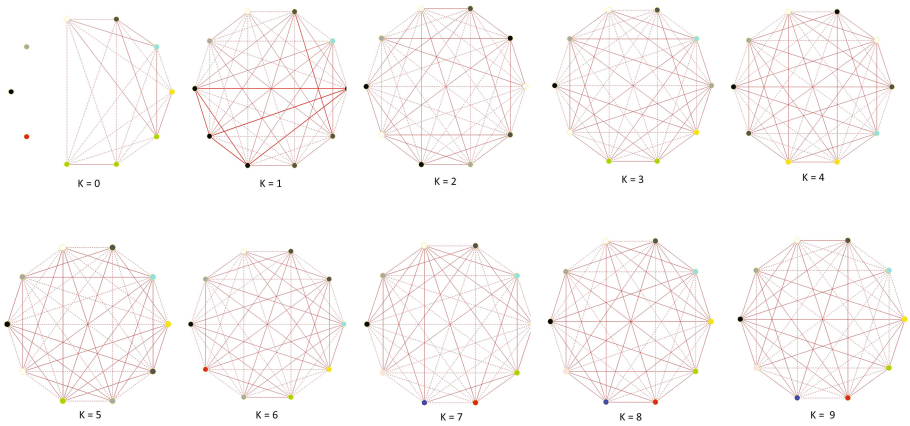


Fig. 7. Community detection by VOS clustering algorithm based on Neighborhood selection with randomly assigned epistatic genes.

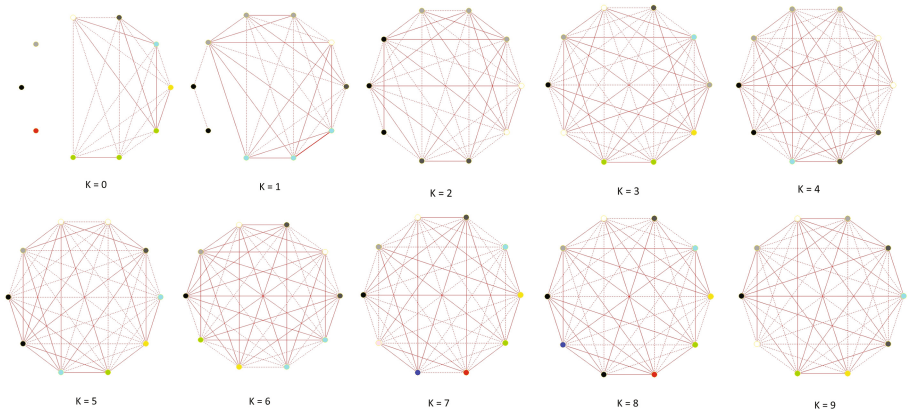


Fig. 8. Community detection by VOS clustering algorithm based on Neighborhood selection with adjacent epistatic genes.

general trend can be observed that the number of communities first decreases and then grows again.

A conjecture we obtained from the pictures is that the correlations or anti-correlations are first very strong and then weaken again. This can be measured by the squared correlation. It is an indication on how much the results of two nodes depend on each other (either positively or negatively). Values close to zero indicate independence of the results at two different nodes. The average squared correlation between nodes in the network is shown in Fig. 6 – both for adjacent and randomly assigned epistatic genes.

Clearly both landscapes have a peak at low values of k. What is striking, is that the peak for the NK-landscape with low values of k has a sharp decay in

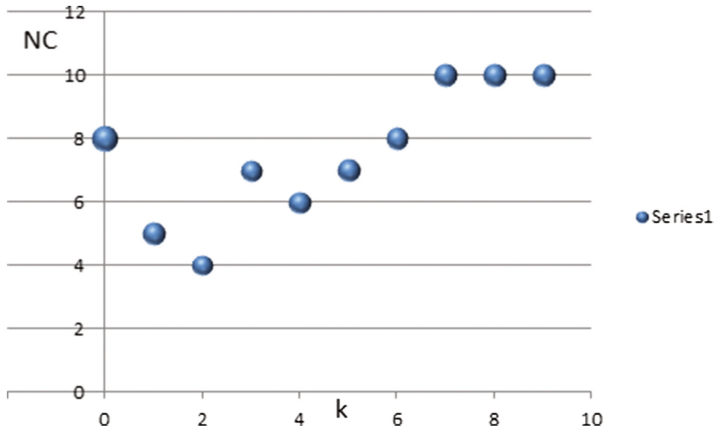


Fig. 9. Community detection by VOS clustering algorithm based on randomly assigned epistatic genes.

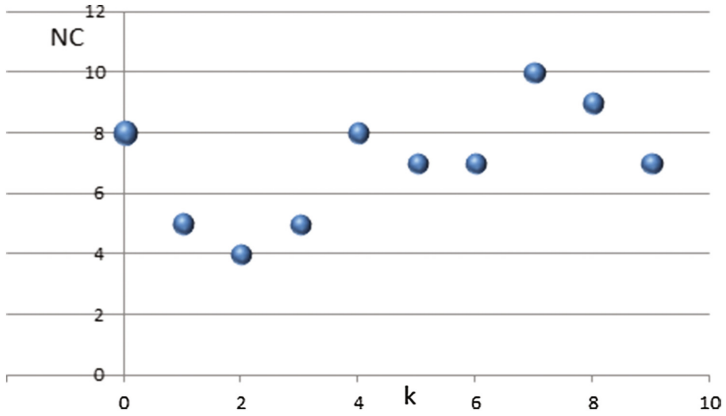


Fig. 10. Community detection by VOS clustering algorithm based on adjacent epistatic genes.

average squared correlation, whereas the decay for the adjacent case is gradual. Again this coincides with the finding that for randomly assigned epistatic genes a sharp transition in computational complexity appears whereas the transition is gradual for the case of adjacent epistatic genes.

Viewing the plot one might even speculate that the observed phenomena is a *sawtooth transition*. This is found in other complex systems at the edge of chaos and is conjectured to be a universal law for macroscopic observations at the transition from systems with complex, but still predictable behavior, to chaotic and unpredictable systems (see for instance Adriaans [1]). Further analysis on larger models and the theoretical analysis of analogies between the models will be required to either confirm or reject this interesting hypothesis.

Table 3. Comparison clustering algorithm applied in community detection, Louvain clustering Algorithm compare to VOS clustering algorithm with randomly assigned epistatic genes

Louvain clustering			VOS clustering		
k	NC	Q	k	NC	Q
0	9	0.842457	0	8	0.8645009
1	6	0.459726	1	5	0.6995902
2	5	0.613504	2	4	0.5636842
3	7	0.630355	3	7	0.6158102
4	7	0.606203	4	6	0.6416844
5	7	0.741914	5	7	0.6716613
6	10	0.701109	6	8	0.6608056
7	10	0.718037	7	10	0.7180642
8	10	0.717084	8	10	0.7285837
9	9	0.757455	9	10	0.7050733

Table 4. Comparison clustering algorithm applied in community detection, Louvein clustering Algorithm compare to VOS clustering algorithm with adjacent epistatic genes

Louvein clustering			VOS clustering		
k	NC	Q	k	NC	Q
0	9	0.842457	0	8	0.8645009
1	6	0.677228	1	5	0.7196617
2	6	0.714034	2	4	0.6285909
3	6	0.713723	3	5	0.6452446
4	8	0.663882	4	8	0.6670689
5	7	0.656833	5	7	0.6213816
6	9	0.679317	6	7	0.6804879
7	8	0.691417	7	10	0.6805063
8	9	0.680426	8	9	0.7017337
9	10	0.715158	9	7	0.6856205

In the plots of Figs. 2 and 3 (Louvain) and Figs. 9 and 10 (VOS) we also show the observed modularity of community components. Here we shift a bit more emphasis on the results of the Louvain method as it explicitly seeks to find communities based on modularity. However, the graphical results are less clear for this and to find a trend we also put the tables with the numerical results in Table 3 (Random) and Table 4 (Adjacent). From these numerical results it can be obtained that the modularity of the communities first decreases slightly and

then goes up again. Clearly the highest average modularity is achieved for $k = 0$ in which case nodes are relatively isolated. Again, the peak is pronounced a bit stronger for randomly assigned epistatic genes.

5 Summary and Outlook

This paper looked at the graph derived from the correlation structure among the component functions of an NK-landscape Model. The results show that the community structure that is detected for this 'correlation graph' does not correspond with the community structure of the epistatic link network which has many components for small values of k and only one big component for $k = N-1$ (every gene is linked to every other gene). Instead the correlation network has the lowest number of components for $k = 2$. For values lower and higher the number of communities clearly grows. As the critical transition from polynomial time solvable maximization problems to NP complete maximization problems appears at the transition from $k = 1$ to $k = 2$ (for random networks) we suspect that these findings might be not coincidental. We show also that the average squared correlation reaches a sharp peak near this value of k. This peak is less pronounced for adjacent epistatic genes which do not undergo a critical transition but a gradual transition in terms of complexity.

So far we have only studied the case $N = 10$ and studies on larger networks are required in the future to improve the generality of the findings. A problem that needs to be solved for such studies is how to tame the 'explosion' in the size of the random number tables needed to generate the NK-landscapes. A useful proposal has been made by Altenberg [2], who suggested to re-generate the random numbers on-the-fly when needed and provided a function that can be used for this.

Further work is also required to understand these findings from a theoretical point of view. For now, the findings show that it will be interesting to study not only the network structure gleaned from the epistatic links but also the resulting network structure obtained from correlation patterns among phenotypic traits. Perhaps one of the most interesting conjectures that should be investigated is the transition of the correlation structure that revealed a sawtooth shape. The examination of this and analogies with other models that undergo a sawtooth transition could be an interesting topic of a follow up work.

Last but not least, in the different context of multiobjective and many-objective optimization [7] the problem of maximizing the components of a NK landscape could yield an interesting test case for many objective optimization with a tunable degree of correlation between the objective functions. To this end, first results on how to exploit community structure for more effective maximization have recently been shown on a different optimization problem [9].

Acknowledgements. Michael Emmerich gratefully acknowledges inspiration from the Lorentz Center Workshop on 'What is Complexity and How do We Measure it?' organized by Erik Schultes, Lude Franke, and Peter Adriaans in the Lorentz Center

Leiden, November 2014. Asep Maulana gratefully acknowledges financial support by the Indonesian Endowment Fund for Education (LPDP).

References

1. Adriaans, P.: Facticity as the amount of self-descriptive information in a data set (2012). arXiv preprint: arXiv:1203.2245
2. Altenberg, L.: Nk landscapes. In: Michalewicz, Z., Bäck, T., Fogel, D. (eds.) Handbook of Evolutionary Computation. Oxford University Press (1997)
3. Anderson, P.: Perspective: complexity theory and organization science. Organ. Sci. **10**(3), 216–232 (1999)
4. Batagelj, V., Mrvar, A.: Pajek-program for large network analysis. Connections **21**(2), 47–57 (1998)
5. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theor. Exp. **2008**(10), P10008 (2008)
6. Frenken, K.: A complexity approach to innovation networks. The case of the aircraft industry (1909–1997). Res. Policy **29**(2), 257–272 (2000)
7. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: a short review. In: IEEE Congress on Evolutionary Computation, pp. 2419–2426. Citeseer (2008)
8. Kauffman, S., Levin, S.: Towards a general theory of adaptive walks on rugged landscapes. J. Theor. Biol. **128**(1), 11–45 (1987)
9. Maulana, A., Jiang, Z., Liu, J., Bäck, T., Emmerich, M.: Reducing complexity in many objective optimization using community detection. In: IEEE Conference on Evolutionary Computation, Sendai, Japan, May 2015
10. van Eck, N.J., Waltman, L.: VOS: a new method for visualizing similarities between objects. In: Decker, R., Lenz, H.-J. (eds.) Advances in Data Analysis. Studies in Classification, Data Analysis, and Knowledge Organization, pp. 299–306. Springer, Heidelberg (2007)
11. Weinberger, E.D.: Local properties of Kauffmans n-k model: a tunably rugged energy landscape. Phys. Rev. A **44**(10), 6399 (1991)

Applications of Evolutionary Algorithms

River Flow Forecasting Using an Improved Artificial Neural Network

Josiah Adeyemo(✉), Oluwaseun Oyebode, and Derek Stretch(✉)

Civil Engineering Department, University of Kwazulu Natal, Durban, South Africa
adeyemoja@gmail.com , 215082067@stu.ukzn.ac.za

Abstract. Artificial neural network (ANN) is a popular data-driven modelling technique that has found application in river flow forecasting over the last two decades. This can be attributed to its ability to assimilate complex and nonlinear input-output relationships inherent in hydrological processes within a river catchment. However despite its prominence, ANNs are still prone to certain problems such as overfitting and over-parameterization, especially when used under limited availability of datasets. These problems often influence the predictive ability of ANN-derived models, with inaccurate and unreliable results as resultant effects. This paper presents a study aimed at finding a solution to the aforementioned problems. Two evolutionary computational techniques namely differential evolution (DE) and genetic programming (GP) were applied to forecast monthly flow in the upper Mkomazi River, South Africa using a 19-year baseline record. Two case studies were considered. Case study 1 involved the use of correlation analysis in selecting input variables during model development while using DE algorithm for optimization purposes. However in the second case study, GP was incorporated as a screening tool for determining the dimensionality of the ANN models, while the learning process was subjected to sensitivity analysis using the DE-algorithm. Results from the two case studies were evaluated comparatively using three standard model evaluation criteria. It was found that results from case study 1 were considerably plagued by the problems of overfitting and over-parameterization, as significant differences were observed in the error estimates and R2 values between the training and validation phases. However, results from case study 2 showed great improvement, as the overfitting and memorization problems were significantly minimized, thus leading to improved forecast accuracy of the ANN models. It was concluded that the conjunctive use of GP and DE can be used to improve the performance of ANNs, especially when availability of datasets is limited.

Keywords: Artificial Neural Networks · Differential evolution · Genetic Programming · River flow

1 Introduction

The management of water resource systems has always been a major concern to water managers and decision makers, most especially in water-stressed countries

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_13

such as South Africa. Hydrologists and water resources engineers have developed various approaches towards managing the relatively little amount of water in these regions in order to meet increasing profile of water demand. However, explosive increase in population continues to place higher demands on the limited water resources. A major constituent of the hydrologic cycle streamflow (river flow), remains a vital point of reference in water resources management due to its role as a major source of freshwater accessibility for the sustainability of man, animal and the natural environment. Thus streamflow prediction is highly essential to the making of worthwhile decisions by relevant stakeholders. Various approaches that have been employed by researchers with the aim of predicting river flows. However, insufficient datasets often pose serious challenge to the achievement of accurate and reliable predictions (Babovic and Keijzer 2002). Therefore, there is a need to find out suitable approaches that could serve as alternatives in the bid to achieving accurate and reliable predictions in data sparse regions. Evolutionary computation (EC) and global optimization techniques have gained much popularity in hydrological modelling studies due to their ability to produce robust models while also enhancing faster convergence towards global optimum (Olofintoye et al. 2014; Oyebode and Adeyemo 2014a; 2014b). The ease with which they are integrated into other modelling techniques is also considered to be a major reason for their prominence. As part of a current research project (Modelling of streamflow response to hydroclimatic variables in the upper Mkomazi River, South Africa), the performance of two evolutionary inspired data-driven modelling techniques - genetic programming (GP) and differential evolution (DE)-trained artificial neural networks (DE-ANN) are being compared. This is done with the aim of determining the better approach to be adopted for river flow prediction in the upper Mkomazi River under limited availability of datasets. The initial results of the application of GP to modelling flow dynamics in upper Mkomazi River have been reported in Oyebode et al. (2014), which does not include a performance comparison with the ANN technique. By contrast, the purpose of this paper is to present the development and application of the DE-ANN to the upper Mkomazi River, and then to compare its predictions with GP monthly flow estimates. There is naturally some aspect of this paper which has the same connection with Oyebode et al. (2014), but the emphases are different. Two case studies involving the use of ANNs are presented. Case study 1 involve the use of the same set of input variables employed in GP model development, while the second case study entails the introduction of early stopping method, incorporation of GP as a screening tool, and the subjection of the ANN learning process to sensitivity analysis. For the purpose of identification, the ANNs in Case study 1 and 2 will hereafter be referred to as ANN-1 and ANN-2 respectively, and their results in comparison with that of GP are presented thereafter.

2 Methodology

2.1 Artificial Neural Networks (ANNs)

ANNs are computational intelligence (CI) techniques inspired by the neurological processing ability of the human brain. ANNs consist of a pool of simple processing units called neurons which communicate by sending signals to each other over a large number of weighted connections (Krse and van der Smagt 1996). The operating principles of ANNs is based on parallel distributed information processing that is capable of storing experiential knowledge gained through the process of learning, and making it available for future use (Elshorbagy et al. 2010). The processing units function by receiving inputs from external sources or other neurons in the network and computing output signals which is transmitted to other units. These processing units are found in layers commonly categorized as input, hidden and output layers. Non-linearities inherent in the inputs of the system being modelled are transformed into a linear space by the use of an activation function in the hidden layer of the network. The commonly used activation functions are sigmoidal functions such as the logistic and hyperbolic tangent functions (Maier and Dandy 2000). The major network topologies that characterize the architecture of ANNs are the feed-forward neural networks (FFNN) and the recurrent neural networks; with multilayer perceptron (MLP), radial basis function (RBF) networks, Kohonens self-organizing feature maps (SOFM) and Elman-type RNN as the most popular ANNs (Coulibaly and Evora 2007; Jha 2007). Numerous specialized learning algorithms have been employed for the purpose of training and subjecting ANNs to adaptive learning. Examples include methods such as back propagation (BP), Levenberg Marquardt (LM), conjugate descent (CG), genetic algorithm (GA) and differential evolution (DE). Hence, the ability of ANNs to assimilate complex and nonlinear input-output interactions makes it suitable for predictive studies in the field of water resources. The multilayer feed-forward neural network (FFNN) architecture was employed in this study. The FFNN was trained using differential evolution (DE) algorithm. DE has been effectively applied for solving real-world science and engineering problems due to its ability to diffuse close to the global optimum solution (Qian and Zhao 2007; Adeyemo and Otieno 2010; Pal et al. 2010). With specific reference to ANN training, DE has been found to produce improved and faster convergence using only small number of parameters for its algorithm setup, unlike other training algorithms such as BP, LM, CG and GA which are either susceptible to local optima, or require high computational times and huge number of iterations to obtain satisfactory results (Piotrowski and Napiorkowski 2011; Oyebode and Adeyemo 2014b). The reader is referred to Storn and Price (1997) and Abdul-Kader (2009) for details on the working principles of DE.

2.2 Study Area and Datasets

The Mkomazi catchment is located in the KwaZulu-Natal province of South Africa and the third largest river in the province. The catchment situated

around 29°17'24"E and 29°35'24"S, derives its source from the upper Drakensberg mountains and discharges into the Indian Ocean, draining an area of about 4 400 km^2. The river is of approximately 160 km in length and elevated at about 3 300 m above sea level, stretching from the Northwest to the Southeast region. The climate is characterized by high seasonality with dry winters and summer rainfall season as Mean Annual Precipitation (MAP) varies between 700 and 1200 mm year-1 with highly intra- and inter-seasonal flows (Flugel and Marker 2003). The MAP is higher in the upper, higher elevations of the river catchment (950 mm to about 1 200 mm) (Taylor et al. 2003), and as a result, a significant amount of catchment runoff is generated in the upper part of the catchment. Datasets relating to the study area were provided by the Department of Water Affairs (DWA) and the South African Weather Service (SAWS). Historical monthly records of flow were obtained from DWA for a 19-year period (1994–2012) from gauging station U1H005 (Mkomazi River @ Lot 93 1821) with geographical coordinates between 29°44'37.3" South longitudes and 29°54'17.8" East latitudes were applied in this study. The SAWS provided the corresponding climatic data (rainfall and temperature) from three independent weather stations namely Pietermaritzburg, Shaleburn and Giant Castle stations located within the study area.

2.3 Selection of Input Variables

The datasets made available by DWA and SAWS include river flows, rainfall and temperature which cover a 19-year period (1994–2012). In this study, the dependency between input variables and the associated lag effect were determined using correlation analysis. Table 1 presents the results of the correlation analysis used in selecting input variables into the GP input space. The results showed high correlation between the flows for the past three years and that of any given year. The results also indicated that flow for the given year highly corresponds to the rainfall and temperature values of the preceding year across the three independent weather stations. However, outcome of the analysis on higher number of lags other than the reported ones produced lower correlation values, and were therefore discarded.

The input vector spaces of the GP models were populated with a total of nine (9) input variables. These inputs comprise of flows for a given month in the last three (3) years (Q_t, Q_{t-1}, Q_{t-2}), rainfall values from the three independent weather stations for the same month in the preceding year ($R1_t, R2_t, R3_t$), and their corresponding temperature values ($T1, T2, T3$). The numbers 1, 2 and 3 represent Pietermaritzburg (PMB), Shaleburn, and Giant Castle weather stations respectively. The approach employed for monthly flow prediction in this study was to adopt a 1-year lead time using individual monthly models. Thus a total of twelve individual monthly models were developed, while the flow being modelled for a given month in the next year is designated as the target output Q_{t+1}. The mathematical representation of the 1-year lead time model adopted can be expressed as:

$$Q_{t+1} = f(Q_t, Q_{t-1}, Q_{t-2}, R1_t, R2_t, R3_t, T1_t, T2_t, T3_t) \quad (1)$$

Following the construction and formulation of the input vector space, the flows and the climatic datasets prepared for each month constituted sixteen (16) data points. As part of the model development process, the datasets were split into two subsets using a random sampling method with two-third of the datasets used for model training and the remaining one-third for validation.

Table 1. Results of correlation analysis showing the relationship between historical values of input parameters and target output

Input parameters	Target output (Q_{t+l})
Qt	0.9983
Q_{t-1}	0.9957
Q_{t-2}	0.9966
$R1_t$	0.7256
$R2_t$	0.9203
$R3_t$	0.8856
$T1_t$	0.8605
$T2_t$	0.8046
$T3_t$	0.8072

3 Model Development

3.1 Case Study 1

As earlier mentioned, the multilayer FFNN topology was employed in this study. The architectural design of the FFNN models developed comprised of three layers: one input, one hidden and one output layer. The modelling strategy employed was to subject the ANN-1 to the same set of data used in the development of the GP-derived models as reported in Oyebode et al. (2014). This was done to avoid any form of bias. Thus, the input layer of the ANN-1 models comprised of nine input nodes, representing the nine selected input variables used for the GP models, while the output node comprised of only one neuron (the target output). The optimal architecture of each monthly model was determined by incrementally varying the number of hidden layer nodes from 2 to 10 using a single stepping approach. The DE algorithm which was utilized in training the network was run for 10 000 generations for each of the monthly models on an Intel Core i7 PC with 3.40 GHz and 4 GB; same as the one used for the GP algorithm run. The DE algorithm was however written using Visual Basic for Applications (VBA) programming language. The population size, NP, and crossover constant, CR, were used to control the algorithm run, while the mutation scale factor, F, controlled the amplification of differential variation during the run.

Following the suggestion of Price and Storn (2013), the DE control parameters, NP, CR and F were set at "D multiplied by 10", 0.9 and 0.4 respectively, (where D is the number of weights and biases in the selected architecture). In terms of data-preprocessing, a logistic sigmoidal-type activation function of between 0 and 1 was used in the hidden layer of the FFNN, to rescale the inputs in the range 0.1–0.9. The rescaling of the inputs to the extreme ranges of the activation function was avoided in accordance with the advice of Maier and Dandy (2000). It has been earlier established that the rescaling of input variables towards the extreme bounds of the activation function could reduce the size of the weight updates, thus resulting into flats spots during training (Maier and Dandy 2000). A linear activation function was however employed in the output layer, in order for the network to transform nonlinearities in the inputs into a linear space.

3.2 Case Study 2

One of the major challenges that confronts modellers in the application of ANNs in hydrological modelling studies is over-parameterization of inputs. Thus in this second case study, an attempt was made to curb the effects of over-parameterization which may likely occur in case study 1; considering that limited amount of datasets were employed. The approach applied was to introduce GP as a screening tool for selecting the input variables into the input vector space of the ANN-2 models. Results reported in Oyebode et al. (2014) have already established the ability of GP to screen and prioritize input variables according to their contribution to the fitness of the best program solutions, thereby determining the impacts of each input variable on predictions made. The impact of each input variable is a function of its frequency of occurrence during the GP algorithm run, and this is computed upon successful completion of each GP algorithm run. Table 2 presents the impact of each of the input variables used for river flow prediction in the study. The results are scaled between 0.0 and 1.0. A value of 1.0 in the frequency column indicates that the particular variable appeared in 100 Following the input impact results computed by GP (see Table 2), four out of the nine variables were selected for the development of ANN-2 models. The selection of the four input variables was based on their impacts on the predictive accuracy of the GP models. Therefore, the dimensionality of the ANN-2 models was reduced when compared with the ANN-1 models. In addition, different combinations of input variables were used across the twelve monthly models unlike in ANN-1 where the same combination was employed in all the models. Table 3 presents the selection and combinations of the four input variables used for the development of each monthly model.

Furthermore, the learning process of the ANN-2 models was fine-tuned by subjecting the DE algorithm to sensitivity analysis. This involved varying the parameter settings during the run to determine the optimal parameters that will generate the least errors. CR was varied between 0.5–0.9, while F was altered between 0.1–0.5. Both CR and F were adjusted incrementally using a stepping value of 0.1. An early-stopping method was also introduced to avoid the problem of overfitting. Early stopping seeks to identify the point where minimum error

Table 2. Impacts of input variables in terms of their frequency of occurrence in the best team programs of each GP monthly model

Inputs	Frequency of input variables												
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg.
Qt	0.83	0.97	0.67	0.67	0.43	0.53	0.50	0.77	0.60	0.30	0.63	0.43	**0.61**
Q_{t-1}	0.77	0.47	0.43	0.70	0.53	0.53	0.63	0.60	1.00	0.70	0.83	0.40	**0.63**
Q_{t-2}	0.50	0.40	0.63	0.83	0.53	0.50	0.90	1.00	0.80	0.93	0.77	0.60	**0.70**
$R1_t$	0.67	0.53	0.20	0.63	0.20	0.37	0.83	0.83	0.27	0.67	0.60	0.73	**0.54**
$R2_t$	0.63	0.53	0.43	0.40	0.53	0.73	0.97	0.20	0.50	0.70	0.53	0.53	**0.56**
$R3_t$	0.63	0.57	0.47	0.63	0.30	0.33	0.40	0.60	0.23	0.83	0.60	0.33	**0.49**
$T1_t$	0.53	0.70	0.97	0.97	0.23	0.30	0.50	0.30	0.60	0.23	0.50	0.63	**0.54**
$T2_t$	0.80	0.77	0.47	0.43	0.43	0.73	0.63	0.70	0.27	0.90	0.80	0.47	**0.62**
$T3_t$	0.60	0.47	0.57	0.37	0.53	0.73	0.23	0.57	0.27	0.27	0.67	0.57	**0.49**

Table 3. Combination of input variables used in developing the ANN-2 models

Inputs	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
$\mathbf{Q_t}$	X	X	X	X				X	X			
$\mathbf{Q_{t-1}}$	X			X	X	X	X		X		X	
$\mathbf{Q_{t-2}}$			X	X	X		X	X	X	X	X	X
$\mathbf{R1_t}$	X						X	X				X
$\mathbf{R2_t}$		X			X	X	X			X		
$\mathbf{R3_t}$										X		
$\mathbf{T1_t}$		X	X	X					X			X
$\mathbf{T2_t}$	X	X				X		X		X	X	
$\mathbf{T3_t}$			X		X	X					X	X

on the validation datasets starts to increase, and immediately stops training to prevent overfitting. With the exception of the incorporation of GP and early-stopping methods, as well as the fine-tuning of the DE algorithm, all other parameter settings and procedural steps employed in developing the ANN-1 models were maintained.

In all, nine input variables were used in the ANN-1 models, while four were used in the ANN-2 models. Thus the optimal network architecture in both case studies is a function of the optimal number of hidden nodes that returns the minimum error between observed and predicted values at the end of the run. Three (3) standard model evaluation criteria namely, mean absolute percent error (MAPE), root mean square error (RMSE) and coefficient of determination (R2) were used to investigate the performance of the monthly models developed in this study, and their mathematical expressions reported in Oyebode et al. (2014).

4 Results and Discussions

The performance of all the models developed by the ANNs (ANN-1 and ANN-2) was subjected to test by adopting the three performance evaluation criteria used for the GP models (i.e. MAPE, RMSE and R2). The performance of ANN models were evaluated against those obtained using the GP technique. The performance evaluation results are presented in Tables 4 and 5, while the optimal network architectures are presented in Table 6. The performance of the different models in predicting flows in the upper Mkomazi River is analyzed and presented with reference to each of the case studies.

Table 4. Comparison of MAPE, RMSE and R2 values between GP and ANNs during training

Month	Training Phase								
	MAPE (%)			RMSE			R^2		
	GP	ANN-1	ANN-2	GP	ANN-1	ANN-2	GP	ANN-1	ANN-2
Jan	3.9401	1.2865	3.2593	1.4968	0.6778	1.7259	0.9964	0.9992	0.9951
Feb	0.9966	0.4008	7.2021	0.4974	0.2689	5.7134	0.9994	0.9997	0.8653
Mar	3.6570	3.1207	91.2276	1.2469	1.1404	41.5205	0.9982	0.9989	0.1176
Apr	5.6798	3.4220	50.1606	1.0710	0.7181	8.4563	0.9891	0.9949	0.2966
May	2.6864	0.6667	1.7045	0.1982	0.0694	0.1627	0.9970	0.9991	0.9961
Jun	1.1854	1E-08	0.1711	0.0607	7E-10	0.0081	0.9986	1.0000	1.0000
Jul	1.2691	0.2050	27.5932	0.0558	0.0150	1.3923	0.9985	1.0000	0.6224
Aug	4.2479	4.1465	15.3046	0.1047	0.1799	0.5216	0.9972	0.9918	0.9302
Sep	11.1474	5.3100	10.9293	0.0607	0.2615	0.4853	0.9988	0.9984	0.9946
Oct	3.1860	4.2952	58.9866	0.1607	0.4268	4.4800	0.9994	0.9953	0.4610
Nov	4.2854	2.2617	34.0076	0.4855	0.2183	3.5226	0.9975	0.9995	0.8667
Dec	6.0007	1E-10	23.6360	0.8642	3E-11	5.5912	0.9972	1.0000	0.8304
Average	**4.0235**	**2.0929**	**27.0152**	**0.5252**	**0.3313**	**6.1317**	**0.9973**	**0.9981**	**0.7480**

4.1 Case Study 1

It can be observed from Table 4 that both the GP and ANN-1 models provided very competitive performance during the training phase, with the ANN-1 models having a slight edge over the GP models. During training, the ANN-1 models converged better towards zero than the GP models, while producing the lowest errors in the months of June and December. An agreement in the maximum MAPE estimates of the GP and ANN-1 models can be noticed, as both techniques (GP and ANN-1) produced maximum MAPE estimates in the month of September 5.31% and 11.1% respectively. It can also be noted that the RMSE estimates recorded by the ANN-1 models was in line with the orientation of its

Table 5. Comparison of MAPE, RMSE and R2 values between GP and ANNs during validation

Month	Validation Phase								
	MAPE (%)			RMSE			R^2		
	GP	ANN-1	ANN-2	GP	ANN-1	ANN-2	GP	ANN-1	ANN-2
Jan	3.4179	65.407	45.769	1.3535	29.1835	13.0723	0.9969	0.0992	0.8444
Feb	3.0490	22.707	61.138	1.0126	18.3385	19.1202	0.9923	0.9651	0.9509
Mar	4.3077	35.350	17.067	1.3797	20.2504	10.6233	0.9932	0.3559	0.6991
Apr	6.1725	21.983	18.862	1.1868	6.3037	5.3593	0.9741	0.5174	0.6993
May	1.8900	57.279	57.748	0.1245	3.8866	3.9262	0.9954	0.2272	0.2234
Jun	0.5932	40.584	9.4256	0.0325	1.6614	0.4419	0.9990	0.6659	0.7441
Jul	0.7575	87.680	21.531	0.0257	2.5899	0.6496	0.9999	0.5788	0.5137
Aug	7.5730	58.147	46.672	0.3401	3.9421	1.3623	0.9881	0.0080	0.9306
Sep	5.1472	100.67	107.54	0.1983	1.8091	1.8671	0.9740	0.2979	0.2917
Oct	6.1092	125.09	58.171	0.2653	3.3390	1.7698	0.9905	0.0959	0.6667
Nov	2.1735	255.03	11.610	0.2918	24.8077	1.3853	0.9978	0.1231	0.9707
Dec	2.4802	89.395	87.983	0.5242	16.0652	8.3457	0.9978	0.0917	0.9295
Average	**3.6392**	**65.407**	**45.769**	**0.5612**	**11.0148**	**5.6602**	**0.9916**	**0.3355**	**0.7053**

MAPE estimates, as the minimum RMSE values converged towards zero and were also generated in the months of June and December. The maximum RMSE of 1.14 was however produced in the March model. Although, the performance of the ANN-1 models was marginally better in most of the months during training, some dominance by the GP models were however evident in the August, September and October models. The R2 values in both techniques (GP and ANN) were found to be comparable, as high correlation between observed and predicted flows were recorded. The R2 values ranged between 0.9918–1.0000 in the ANN-1 models, and 0.9891–0.9994 in the GP models. Generally, the performance of the GP and ANN-1 models during training can be said to be comparable, but with the ANN-1 models producing better convergence in most cases. This is in correlation with the study of Ni et al. (2010) where ANN performed slightly better than GP during the training phase.

On the other hand, GP models performed considerably better than the ANN-1 models during validation, as the ANN-1 produced higher error estimates. The errors produced in the GP models were better converged towards zero, and were estimated to be 0.6%–7.6% and 0.03–1.38 for MAPE and RMSE respectively. In furtherance to that, all the GP models maintained the highly positive correlations recorded during training, with R2 values of 0.9740–0.9999. However, the error estimates in the ANN-1 models increased substantially, yielding higher MAPE and RMSE estimates, while generating lower R2 values. This may be considered as a result of poor learning which arises when small amount of datasets

are used for model training in ANNs (Zhang et al. 2010). It can be inferred that ANN-1 simply memorized the learning process as evident in the training results, and thus led to its inability to produce adequate generalization on the validation datasets as also experienced by Nourani et al. (2011). Furthermore, it was noticed that the optimization of the network architecture (number of hidden layer nodes) resulted in higher computational demands in terms of training time and computer memory. More understanding in this regard can be found from Table 6 which presents the optimal network architecture of the individual ANN models as determined by the DE algorithm, and returned at each end of the run.

Table 6. Network architecture showing number of hidden layer nodes in the ANN models.

Month	Optimal network architecture	
	ANN-1	ANN-2
Jan	9-10-1	4-4-1
Feb	9-7-1	4-3-1
Mar	9-10-1	4-3-1
Apr	9-7-1	4-3-1
May	9-10-1	4-4-1
Jun	9-5-1	4-5-1
July	9-7-1	4-3-5
Aug	9-8-1	4-4-1
Sep	9-7-1	4-5-1
Oct	9-9-1	4-4-1
Nov	9-10-1	4-3-1
Dec	9-4-1	4-5-1

It was found during the runs that the training speed became slower with increase in number of hidden layer nodes. The reduction in training speed can be considered as a function of the increment in number of synaptic connections between units and adjacent network layers. Thus, as the number of hidden layer nodes increased, greater amount of weight/load was imposed on the network. This observation agrees with Karthikeyan et al. (2013) recent submission in their groundwater level prediction study, that the number of hidden layer nodes influences computational time. Unlike the ANN-1 models, the GP models produced better generalization at a faster learning rate; a product of its ability to distribute the semi-isolated population space into multiple subpopulations, called demes (Oyebode and Adeyemo 2014b). Although, all the simulations in this study were carried out on the same computer, the average computational time of the GP and ANN-1 models were 4 and 8 h respectively. Figures 1 and 2 present plots of observed against predicted flows. The plots clearly show some

few under- and over-estimations of the observed values by the ANN-1 models. Despite ensuring that the validation datasets were within the range of the training datasets, the variations between observed and ANN-1 predicted values during the months of high flows (January, February and March) were more pronounced than that of the months characterized by low flows (June and July).

4.2 Case Study 2

As presented in Table 7, significant improvement can be noticed in the performance of the ANN-2 models compared to the ANN-1 models. Although, the error estimates produced during training were higher than those recorded by the ANN-1 models, it can be observed that the models achieved better convergence in the validation phase. This is evident as the errors produced during validation were far less than those produced during training. This implies that the overfitting and memorization problems that plagued the ANN-1 models were totally eliminated in the ANN-2 models. This is an indication that the introduction of the early stopping method in the second case study was effective in preventing the occurrence of overfitting in the ANN-2 models. Similar experience was reported in a rainfall-runoff modelling study conducted by Siou et al. (2012). In the study, early stopping was also found to have successfully prevented the occurrence of overfitting in ANN-derived models thereby resulting to improved forecast accuracy. The superiority of the ANN-2 models over ANN-1 models is quite noticeable in the R2 values, as the observed and ANN-2 predictions were reasonably correlated, with the exception of the May and September models. A computation of the relative MAPE, RMSE and R2 differences between the ANN models during validation is presented in Table 6, with the ANN-2 models producing lower MAPE and RMSE estimates, and higher R2 values in most of the months.

It is clearly evident from the results that the incorporation of the GP technique and fine-tuning of the ANN learning process led to the improved performance of the ANN-2 models. The introduction of GP as a screening tool, to reduce the dimensionality of the models obviously facilitated quick convergence and reduced over-parameterization effects. In the same vein, the subjection of the DE algorithm to sensitivity analysis invigorated the search for better solutions, and consequently resulted into improved performance with a less complex model architecture. Table 6 clearly shows that the ANN-2 models were able to produce better performance using a less complex architecture (number of hidden nodes). Therefore, it can be said that the methodologies adopted in case study 2 prevented the imposition of higher amount of load on the network (as experienced in case study 1), and consequently translated into lesser computational time. Although, the GP models exhibited dominance in terms of predictive accuracy, convergence rate and adaptation to rare occurrences of extreme events (Figs. 1 and 2), results show that the ANN-2 models produced better predictions and faster convergence when compared to the ANN-1 models. Thus results from this paper indicate that the synergistic integration of the EC techniques (GP and DE) as well as incorporation of early stopping method are promising techniques

Table 7. Relative differences between the ANN models during validation

Month	Relative differences (%)		
	MAPE	RMSE	R2
Jan	30.0	55.2	88.3
Feb	62.9	4.1	1.5
Mar	51.7	47.5	49.1
Apr	14.2	15.0	26.0
May	0.8	1.0	1.7
Jun	76.8	73.4	10.5
Jul	75.4	74.9	11.2
Aug	19.7	65.4	99.1
Sep	6.4	3.1	2.1
Oct	53.5	47.0	85.6
Nov	95.4	94.4	87.3
Dec	1.6	48.1	90.1
Average	**43.3**	**48.6**	**52.4**

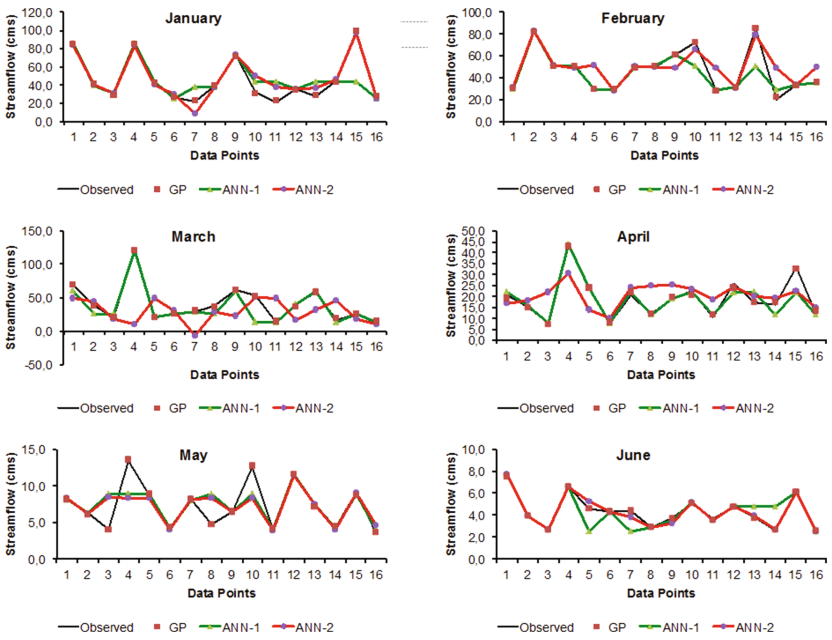


Fig. 1. Observed and predicted flows by the GP and ANN models (January–June)

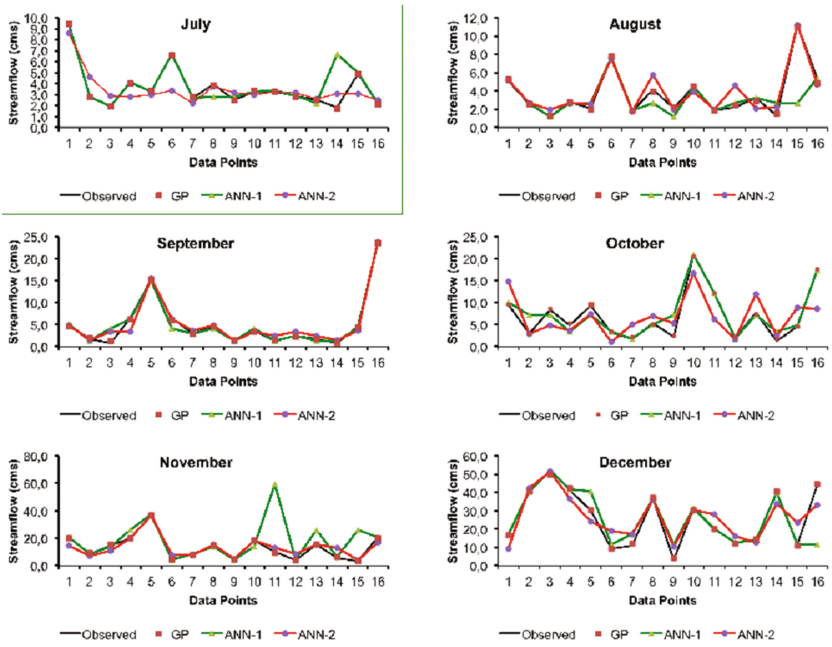


Fig. 2. Observed and predicted flows by the GP and ANN models (July–December)

that could be adopted for improving the performance of ANNs in river flow forecasting, and by extension in water-related modelling studies.

5 Conclusions

In this paper, the performance of two data-driven modelling techniques namely genetic programming (GP) and differential evolution (DE)-trained artificial neural networks (ANNs) were investigated comparatively for monthly river flow prediction using limited amount of datasets. Two case studies were considered in the application of ANNs. Correlation analysis, used for determining the predictor variables in the GP technique was adopted in the first case study. However, in the second case study, GP, DE and early stopping methods were integrated for optimization purposes. The ANN models in the first case study were found to be plagued by the problems of overfitting and over-parameterization, which is typical of ANNs when confronted with small-sized datasets. The ANN models developed in the second case study however show-cased improved performance through the conjunctive effort of early stopping method, GP and DE. Although high difficulty exists in modelling hydrological processes with limited datasets, this study demonstrates the benefits that can be derived from the incorporation of evolutionary computation techniques such as GP and DE. Considering the ease at which they can be integrated into other modelling techniques, they can

be adopted in the development of modular and hybrid models. This will further assist decision makers in addressing issues relating to planning and effective management of water resources.

References

Abdul-Kader, H.: Neural networks training based on differential evolution algorithm compared with other architectures for weather forecasting34. IJCSNS **9**(3), 92–99 (2009)

Adeyemo, J., Otieno, F.: Differential evolution algorithm for solving multi-objective crop planning model. Agric. Water Manage. **97**(6), 848–856 (2010)

Babovic, V., Keijzer, M.: Rainfall runoff modelling based on genetic programming. Nordic Hydrol. **33**(5), 331–346 (2002)

Coulibaly, P., Evora, N.: Comparison of neural network methods for infilling missing daily weather records. J. Hydrol. **341**(1), 27–41 (2007)

Elshorbagy, A., Corzo, G., Srinivasulu, S., Solomatine, D.: Experimental investigation of the predictive capabilities of data driven modeling techniques in hydrology part 1: concepts and methodology. Hydrol. Earth Syst. Sci. **14**(10), 1931–1941 (2010)

Flugel, W., Marker, M.: The response units concept and its application for the assessment of hydrologically related erosion processes in semiarid catchments of Southern Africa. ASTM Spec. Tech. Publ. **1420**, 163–177 (2003)

Jha, G.K.: Artificial neural networks and its applications (2007). http://www.iasri.res.in/ebook/ebadat/5-Modeling

Karthikeyan, L., Kumar, D.N., Graillot, D., Gaur, S.: Prediction of ground water levels in the uplands of a tropical coastal riparian wetland using artificial neural networks. Water Resour. Manage. **27**(3), 871–883 (2013)

Krse, B., van der Smagt, P.: An Introduction to Neural Networks, 8th edn. The University of Amsterdam, Amsterdam (1996)

Maier, H.R., Dandy, G.C.: Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. Environ. Model. Softw. **15**(1), 101–124 (2000)

Ni, Q., Wang, L., Ye, R., Yang, F., Sivakumar, M.: Evolutionary modeling for streamflow forecasting with minimal datasets: a case study in the West Malian River. Environ. Eng. Sci. **27**(5), 377–385 (2010)

Nourani, V., Kisi, Ö., Komasi, M.: Two hybrid artificial intelligence approaches for modeling rainfall runoff process. J. Hydrol. **402**(1), 41–59 (2011)

Olofintoye, O., Adeyemo, J., Otieno, F.: A combined pareto differential evolution approach for multi-objective optimization. In: EVOLVE-A Bridge Between Probability, Set Oriented Numerics, and Evolutionary Computation III, pp. 213–231. Springer (2014)

Oyebode, O., Adeyemo, J.: Reservoir inflow forecasting using differential evolution trained neural networks. In: Tantar, A.-A., et al. (eds.) EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V, Advances in Intelligent systems and Computing, vol. 288, pp. 307–319. Springer, Switzerland (2014a)

Oyebode, O., Adeyemo, J., Otieno, F.: Monthly streamflow prediction with limited hydro-climatic variables in the upper Mkomazi River, South Africa using genetic programming. Fresenius Environ. Bull. **23**(3), 708–719 (2014)

Oyebode, O.K., Adeyemo, J.A.: Genetic programming: principles, applications and opportunities for hydrological modelling. Int. J. Environ. Ecol. Geomatics Earth Sci. Eng. **8**(6), 305–311 (2014b)

Pal, S., Qu, B., Das, S., Suganthan, P.: Optimal synthesis of linear antenna arrays with multi-objective differential evolution. Prog. Electromagnet. Res. PIER B **21**, 87–111 (2010)

Piotrowski, A.P., Napiorkowski, J.J.: Optimizing neural networks for river flow forecasting Evolutionary Computation methods versus the Levenberg Marquardt approach. J. Hydrol. **407**(1), 12–27 (2011)

Price, K., Storn, R.: Differential Evolution (DE) for Continuous Function Approximation (2013). http://www1.icsi.berkeley.edu/~storn/code.html. Accessed 19 July 2013

Qian, G., Zhao, X.: On time series model selection involving many candidate ARMA models. Comput. Stat. Data Anal. **51**(12), 6180–6196 (2007)

Siou, L.K.A., Johannet, A., Valrie, B.E., Pistre, S.: Optimization of the generalization capability for rainfall runoff modeling by neural networks: the case of the Lez aquifer (Southern France). Environ. Earth Sci. **65**(8), 2365–2375 (2012)

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)

Taylor, V., Schulze, R., Jewitt, G.: Application of the indicators of hydrological alteration method to the Mkomazi River, KwaZulu-Natal, South Africa. African J. Aquatic Sci. **28**(1), 1–11 (2003)

Zhang, L., Mernyi, E., Grundy, W.M., Young, E.F.: Inference of surface parameters from near-infrared spectra of crystalline H2OH2O ice with Neural Learning. Publ. Astron. Soc. Pacific **122**(893), 839–852 (2010)

Evolutionary Cost-Sensitive Balancing: A Generic Method for Imbalanced Classification Problems

Camelia Lemnaru(✉) and Rodica Potolea

Computer Science Department, Technical University of Cluj-Napoca, Cluj-Napoca, Romania
{Camelia.Lemnaru,Rodica.Potolea}@cs.utcluj.ro

Abstract. Efficient classification under imbalanced class distributions is currently of interest in data mining research, considering that traditional learning methods often fail to achieve satisfying results in such domains. Also, the correct choice of the metric is essential for the recognition effort. This paper presents a new general methodology for improving the performance of classifiers in imbalanced problems. The method, *Evolutionary Cost-Sensitive Balancing* (*ECSB*), is a meta-approach, which can be employed with any error-reduction classifier. It utilizes genetic search and cost-sensitive mechanisms to boost the performance of the base classifier. We present evaluations on benchmark data, comparing the results obtained by ECSB with those of similar recent methods in the literature: *SMOTE* and *EUS*. We found that ECSB boosts the performance of traditional classifiers in imbalanced problems, achieving ∼45% relative improvement in true positive rate (TP_{rate}) and around 16% in F-measure (FM) on the average; also, it performs better than sampling strategies, with ∼35% relative improvement in TP_{rate} and ∼12% in FM over SMOTE (on the average), similar $textTP_{rate}$ and geometric mean (GM) values and slightly higher area under de curve (AUC) values than EUS (up to ∼9% relative improvement).

Keywords: Imbalanced classification · Meta-approach · Hybrid methodology · Genetic algorithms · Cost-sensitive

1 Introduction

One of the current important challenges in data mining research is classification under an imbalanced data distribution. This issue appears when a classifier has to identify a rare, but important case. Domains in which class imbalance is prevalent include fraud or intrusion detection, medical diagnosis, risk management, text classification and information retrieval [7], unexploded ordnance detection [1], or mine detection [33].

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_14

A classification problem is imbalanced if, in the available data, a certain class is represented by a very small number of instances compared to the other classes [16]. In practice, the problem is addressed with 2-class problems; multi-class problems are translated to binary. As the minority instances are of greater interest, they are referred to as positive instances (positive class).

This paper presents a new general methodology for improving the performance of classifiers under imbalanced conditions. The method, Evolutionary Cost-Sensitive Balancing (ECSB), is a hybrid meta-approach which combines genetic search mechanisms with cost sensitive classification strategies. It involves the identification of the optimal cost matrix and parameter settings for the given problem, selected classifier (inducer) and evaluation metric. The method has been evaluated on benchmark data and compared to recently proposed methods for dealing with class imbalance, yielding significant performance improvements.

The rest of the paper is organized as follows: the next section reviews related work in this area. Section 3 details the proposed ECSB method, which is followed by its experimental validation in Sect. 4. Concluding remarks and future work are discussed in the last section.

2 Learning in Imbalanced Scenarios

Establishing how to assess performance is essential in imbalanced problems. The selection of an inappropriate measure may lead to unexpected predictions, which are not in agreement with the problem goals. This section presents the main evaluation metrics considered in imbalanced domains, a brief analysis of the limitations of traditional algorithms and an overview of existing techniques to tackle the imbalance.

2.1 Measuring Performance in Imbalanced Domains

The most employed evaluation measure for classification problems, the overall accuracy, is unfit in imbalanced domains [8,32], since the minority class contributes very little to its value. In highly imbalanced problems, a good recognition of the majority class translates into a high accuracy, regardless of how well the model identifies minority cases: for a data set with 99% examples for one class and 1% for the other, a model which classifies everything as belonging to the majority class yields 99% accuracy, while failing to identify any minority example.

For an imbalanced problem, the *true positive rate*, (TP_{rate}), also referred to as *recall* or *sensitivity*, is usually more important. However, there are other metrics, derived from the confusion matrix, which may also be relevant for assessing the performance in certain problems. A series of composite measures have been suggested by the scientific community for evaluating the performance in imbalanced problems: in [3,5,11] the *area under the ROC curve* (AUC) is employed; the *geometric mean* (GM) is proposed in [2] and employed in several other studies [11,13]; the *balanced accuracy* (BAcc) is another symmetric measure which

is more suited for imbalanced problems [4]; the *f-measure*, or *f-score* [8,13], and its generalization – the *f_β-measure* – provide a trade-off between the correct identification of the positive class and the cost of false alarms (in number of false positive errors). In [12] it is suggested that, in imbalanced problems, more attention should be given to sensitivity (TP_{rate}) than to specificity (TN_{rate}). In [8], the strategy to follow in imbalanced problems is to maximize the recall while keeping the *precision* under control. Both statements hold true in most imbalanced problems.

We argue that metric selection in imbalanced problems is essential for both model quality assessment and guiding the learning process. The metric should reflect the goal of the specific classification process, not just focus on the imbalance. Thus, if we are additionally dealing with imbalance at the level of the error costs, then associating a cost parameter to account for such disproportions is appropriate. If, on the other hand, the focus is on identifying both classes correctly, then an equidistant metric provides a fair estimation.

2.2 Existing Approaches for Dealing with Imbalance

The existing approaches for dealing with imbalanced problems can be split into: data-centered, algorithm-centered and hybrid solutions.

1. *Data-centered techniques* focus on altering the distribution of the training data: either randomly, or by making an informed decision on which instances to eliminate or add (by multiplying existing examples, or artificially generating new cases). Under this category we find random over- and under-sampling, or more elaborated approaches, such as Synthetic Minority Over-sampling Technique (SMOTE) [5], Tomek links [27], the Condensed Nearest Neighbor Rule (CNN) [14], One-Sided Selection (OSS) [17], the Neighborhood Cleaning Rule (NCL) [18], or Evolutionary Under-Sampling (EUS) [11]. In order to maximize the classification performance in the mining step, one should carefully match the appropriate sampling technique to the learning algorithm employed at that stage. Also, some methods require the analyst to set the amount of re-sampling needed, and this is not always easy to establish. It is acknowledged that the naturally occurring distribution is not always the best for learning [31]. A balanced class distribution may yield satisfactory results, but is not always optimal either. The optimal class distribution is highly dependent on the particularities of the data at hand.
2. *Algorithm-centered techniques*, also known as internal approaches, refer to strategies which adapt the inductive bias of classifiers, or newly proposed methods for tackling the imbalance. For decision trees, such strategies include adjusting the decision threshold at leaf nodes [24], adapting the attribute selection criterion [22], or changing the pruning strategy [36]. For classification rule learners, using a strength multiplier or different algorithms for learning the rule set for the minority class is proposed in [12], while for association rule learners, multiple minimum supports are employed in rule generation [21]. In [23], confidence weights are associated to attribute values (given a class

label) in a kNN approach. For SVMs, class boundary alignment is proposed in [35] and the use of separate penalty coefficients for different classes is investigated in [20]. Newly proposed methods, which deal with the imbalance intrinsically, include the biased minimax probability machine (BMPM) [15], or the infinitely imbalanced logistic regression (IILR) [33].

3. *Hybrid approaches* combine data- and algorithm-centered strategies. A number of approaches in this category consist of ensembles built via boosting, which also employ replication on minority class instances to second the weight update mechanism. Also, the base classifiers may be modified to tackle imbalanced data. Such approaches include SMOTEBoost [6], DataBoost-IM [13], and a complex SVM ensemble [26]. Another hybrid strategy is the one employed in cost-sensitive problems, to bias the learning process according to the different costs of the errors involved [10,25,37]. The method we propose in this paper falls into this category.

2.3 Limitations of Traditional Techniques

It is widely acknowledged that the nature of imbalanced problems is manifold. The essential data characteristic in such areas is the *imbalance ratio* (IR), i.e. the ratio between the number of instances in the majority (m_{Maj}) and minority classes (m_{Min}) – Eq. (1). Other data meta-features which have been shown to influence the behavior of classifiers in such domains are the *size* and the *complexity* of the data [16] and the *instances per attribute ratio* (IAR), i.e. the ratio between the total number of instances (m) and the number of attributes recorder per instance (n), which combines size and complexity information [19] – Eq. (2):

$$IR = \frac{m_{Maj}}{m_{Min}} \tag{1}$$

$$IAR = \frac{m}{n} \tag{2}$$

Also, particularities related to the distribution of the minority samples, such as too many "special cases" in the minority class, may affect the classifiers' capability to recognize all cases of interest (*within-class rarity*, *small disjuncts problem* [32]).

Several studies [16,29] indicate that most traditional classifiers are affected by the class imbalance problem to some extent. This is mainly because the assumptions followed in the training process don't usually hold in imbalanced problems. First of all, classifiers attempt to maximize accuracy, which is not an appropriate measure in imbalanced domains. Moreover, they assume the same distribution in the training and test samples, meaning that the model is customized for a certain distribution which is not the actual occurring distribution. Such a situation appears, for example, when dealing with dynamic distributions (such as the distribution of flu cases, which changes according to the season). Also, the rare cases may be very costly to obtain (in terms of time required, economic costs and/or pain). Moreover, even if the actual distribution is known, it may not be optimal for learning [30].

In [19], the authors perform a systematic analysis of the effect of class imbalance on the performance of six different classifiers, using 32 binary (or binarized) real-life benchmark data sets. The performance of all the classifiers evaluated seemed to be affected by the imbalance. Another conclusion of the study refers to the factors affecting classifier performance. The reduction in performance becomes more severe as the IR increases. However, for the same IR, larger IAR values are associated with improved classifier performance. Therefore, techniques for increasing the value of IAR (i.e. larger data set size and/or smaller complexity) may lead to an improved behavior.

3 Evolutionary Cost-Sensitive Balancing (ECSB)

The objective of the ECSB method is to improve the performance of a classifier (inducer) in imbalanced domains. It is a meta-methodology, which can be employed with any error-reduction classifier. Two strategies are simultaneously followed by the method: (1) use a cost-sensitive meta-classifier to adapt to the imbalance and (2) tune the base classifier's parameters. The outcome of the method is a tuple <M, S> for the triple <p, i, m>, where M is a cost matrix and S is the set of resulting parameter settings for the given problem (p), selected inducer (i) and evaluation metric (m). M is employed in conjunction with the cost-sensitive classifier, in order to build a more efficient classification model, focused on better identifying the underrepresented/interest cases. The search for M and S is performed through evolutionary mechanisms. The cost-sensitive component employs a meta-classifier to make its base classifier cost-sensitive, taking into account the misclassification costs. The main mechanisms for wrapping cost-sensitivity around traditional classifiers usually focus on employing a larger penalty for the errors on classes with higher misclassification cost, or modifying the training data such that the costly cases are proportionally better represented than the others.

The general flow of the method is presented in Fig. 1. The inputs are: the problem (p), translated in terms of a set of labeled examples (i.e. the training set), the base inducer (i) and the metric (m) to use for assessing the performance of i. The result of the method is a <M, S> tuple, which is used by a (meta-) cost-sensitive classifier to build the final classification model.

3.1 The Cost-Sensitive Component

Cost-sensitive learning encompasses several algorithms which focus on minimizing the total expected cost instead of the classifier error. A taxonomy of the types of costs involved in inductive concept learning can be found in [28], the most important being the *misclassification* and the *test costs*. The first category includes the costs which are conventionally considered by most cost-sensitive classifiers, and attempts to quantify the different impact that distinct errors produce. Several solutions address the second category also, which models different types of costs involved in acquiring the data (time, physical pain, money, etc.).

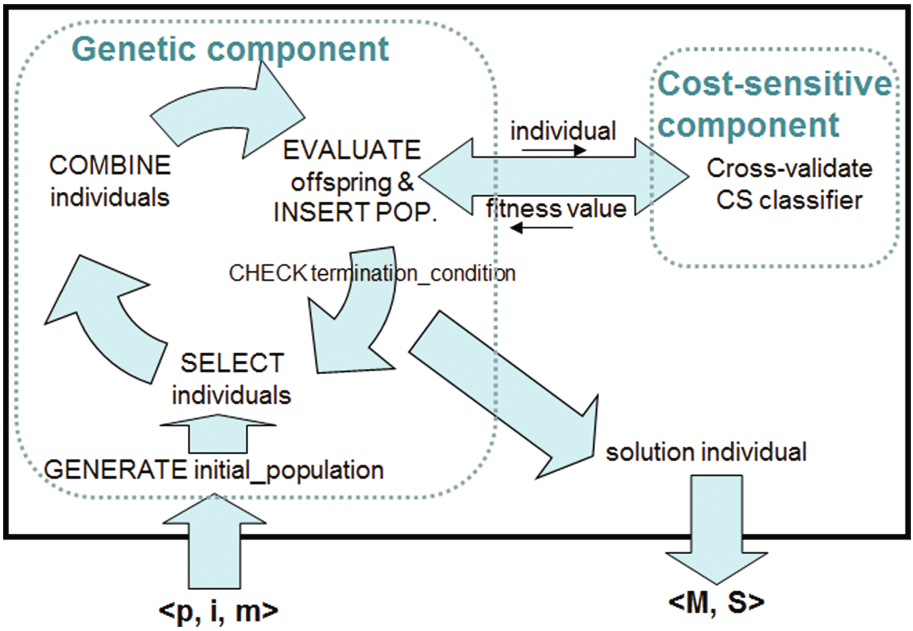


Fig. 1. General ECSB flow

We focus only on misclassification costs, since they can be employed to bias the learning process such as to provide a better identification for the minority class instances. The misclassification costs are represented via a cost matrix $(c_{ij})_{nxn}$, where c_{ij} represents the cost of misclassifying an instance of class j as being of class i. For imbalanced problems, we usually focus on binary classification, i.e. $n = 2$:

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \tag{3}$$

The main diagonal elements (c_{11} and c_{22}) represent the costs of correct identification and are normally smaller than or equal to 0 (i.e. reward or no penalty); c_{12} is the cost of a false negative (i.e. failing to identify a positive) and c_{21} captures the reverse situation. One of the most important difficulties when dealing with different error costs is to quantify misclassification costs. Even if it is relatively easy to determine which errors are more severe than others (e.g. in medical diagnosis $c_{12} > c_{21}$), it is difficult to quantify the gravity of an error exactly, since this may translate, indirectly, to more serious social/moral dilemmas, such a putting a price tag on human life.

In our approach, the cost matrix (M) for the given imbalanced problem is determined indirectly, following a genetic search. We can influence the result of the search by tuning the fitness function employed, which can be more easily translated, given a specific problem, than directly setting the cost matrix.

For example, it is more reasonable to state that the objective is to maximize both TP_{rate} and TN_{rate} in medical diagnosis, or to maximize precision in online advertising, than it is to set specific error costs.

The implementation of the cost-sensitive component has been carried out within the Waikato Environment for Knowledge Analysis (WEKA) framework [34]. Three cost-sensitive strategies have been considered:

(1) use an ensemble method to re-label the training instances according to the Bayes optimal prediction principle, which minimizes the conditional risk (*MC*) [10];
(2) reweight training instances according to the total cost assigned to each class (*CSr*) [34];
(3) predict the class with minimum expected misclassification cost, instead of the most likely class (*CS*) [34].

3.2 The Genetic Component

We have utilized the General Genetic Algorithm Tool for implementing the genetic component [9]. It provides the traditional genetic algorithms (GA) search organization, parent selection and recombination techniques. The specificity of our implementation is the problem representation and the fitness function(s) employed.

The search process starts with the initial population, i.e. a set of potential solutions, generated randomly (lines 1 and 2 in the pseudocode snippet below). By repeatedly applying recombination operators to some of the individuals in the population over a number of cycles, an element (or group of elements) is expected to emerge as a good quality approximate solution to the given problem (the loop between lines 3 and 9). Following a strategy similar to steady state evolution, in each cycle a number of new offspring is generated (additional pool). After evaluating their fitness (line 7), the fittest *p_size* individuals out of the old population and the additional pool (the newly generated offspring) will constitute the new population (line 8):

```
(1) population = generate_initial_population(p_size)
(2) evaluate_fitness (population)
(3) repeat
(4)    parents = select(population)
(5)    offspring = crossover(parents)
(6)    mutate(offspring)
(7)    evaluate_fitness (offspring)
(8)    insert (offspring, population)
(9) until (termination_condition)
(10) return best_individual
```

This strategy considers elitism implicitly. The search process stops when one of the following occurs: the optimal fitness value is reached, the difference between the fitness values of the best and the worst individuals in the current

population is 0, or a fixed (pre-determined) number of crossover cycles have been performed.

Each individual consists of four chromosomes: the first two representing each a misclassification cost (elements of M), and the last two representing parameters for the base classifier (elements of S). Although we have considered only two parameters for S – since most base classifiers used in the experiments have only two important learning parameters – the method can be extended to search for a larger number of parameters, depending on the tuned classifier. The first two chromosomes in the individual represent the meaningful coefficients of the 2×2 cost matrix. We assume the same reward (i.e. zero cost) for the correct classification of both minority and majority classes. Each chromosome consists of 7 genes, meaning that each cost is an integer between 0 and 127. We considered this to be sufficient to account even for large IRs. Gray coding is employed to ensure that similar genotypes produce close manifestations (phenotypes).

Fitness ranking is used to avoid premature convergence to a local optimum, which can occur if in the initial pool some individuals dominate, having a significantly better fitness than the others. Since establishing how to assess performance is essential in imbalanced problems and there is no universally best metric, which captures efficiently any problem's goals, we have implemented several different fitness functions, both balanced and (possibly) imbalanced. For consistency with the literature, we sometimes employ $\mathrm{TP}_{\mathrm{rate}}$ and sometimes recall for referring to the same measure:

$$\textbf{GM}(\text{geometric mean}) = \sqrt{TP_{rate} * TN_{rate}} \tag{4}$$

$$\textbf{BAcc}(\text{balanced accuracy}) = \frac{TP_{rate} + TN_{rate}}{2} \tag{5}$$

$$\textbf{FM}(f_{\beta}\text{-measure } = (1+\beta^2)\frac{prec * recall}{prec + recall} \tag{6}$$

$$\textbf{LIN}\,(\text{linear combination between}\mathrm{TP}_{\mathrm{rate}}, \mathrm{TN}_{\mathrm{rate}}) = \alpha * TP_{rate} + (1-\alpha) * TN_{rate} \tag{7}$$

$$\textbf{PLIN}\,(\text{linear combination between recall, prec.}) = \alpha * Recall + (1-\alpha) * Prec \tag{8}$$

4 Experimental Work

This section presents the experiments performed to validate the ECSB method and to compare it with recent proficient strategies. Subsect. 4.2 presents the general setup: it includes the evaluation methodology employed throughout the experiments, as well as the mechanisms and settings employed. Two different evaluation suites are then presented, with discussions of the results. A first set of tests evaluates comparatively the performance of different specializations of ECSB on large IR, small IAR data sets, since previous analyses [19] have shown that classifiers are most affected on such problems; the second presents a comparison between ECSB and a prominent under-sampling strategy for imbalanced data: Evolutionary Under-Sampling [11].

4.1 Experimental Setup

Experiments have been carried using 2-fold cross-validation. Generally, we have compared (1) the results of the base classifier with default settings (*Base*) with (2) the results obtained by the same classifier following data pre-processing with SMOTE [5] and default settings (*Base+SMOTE*), (3) the results obtained by the classifier following a parameter tuning stage, performed with the genetic component of ECSB (*ECSBT*) and (4) the results obtained by a classifier wrapped in our ECSB method (ECSB).

The specific mechanisms and setting values employed for the genetic component are presented in Table 1. Several fitness functions have been considered. No tuning has been performed on settings of the component so far. Five classifiers have been included in the experimental study, belonging to different categories: lazy methods (k-nearest neighbor, kNN), Bayesian methods (Naive Bayes, NB), decision trees (C4.5), support vector machines (SVM) and ensemble methods (AdaBoost.M1, AB). Table 2 describes the parameters considered for the base classifiers (in ECSB and ECSBT).

4.2 General Validation on Large IR, Small IAR Data Sets

We have performed a first analysis on benchmark data sets having large IR and small IAR values, as considered in [19] – Table 3. This combination of imbalance-related factors has a strong negative influence on the performance of classifiers. All three cost-sensitive strategies were considered (MC, CS and CSr), and five different fitness functions (GM, BAcc, FM with $\beta = 1$, LIN and PLIN, the last two having $\alpha = 0.7$).

This results in 15 combinations for the ECSB method, compared with the results obtained by the classifier alone (*Base*), the classifier with SMOTE (Base+*SMOTE*) and the classifier with tuned parameter values (*ECSBT*).

Table 1. Specific genetic mechanisms employed

Setting	Value
Population type	Single, similar to steady state
Initial population	Random
Population size	20
Additional pool	10
Crossover cycles	200
Parent selection	Roulette wheel
Recombination operators	Crossover: random crossover, 4 points
	Mutation: single bit uniform mutation, 0.2 rate
Fitness functions	GM; BAcc; FM; LIN; PLIN
Other	Fitness ranking
	Elitism, implicit with use of single population

Table 2. Base classifiers parameter ranges

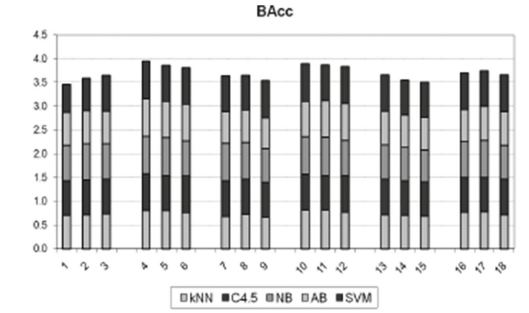
Classifier	Parameters	Type and range
kNN	K – number of neighbors	Integer between 1 and 10
C4.5	C – confidence ratio	Real, between 0 and 0.4
	M – min. number of instances per leaf	Integer, between 1 and 5
NB	n.a	n.a.
AB	P – weight threshold for weight pruning	Integer, between 1 and 127
	I – number of iterations	Integer, between 1 and 30
SVM	C – complexity	Real, between 1 and 100
	E – exponent	Integer, between 1 and 11

Table 3. Large IR, small IAR data sets

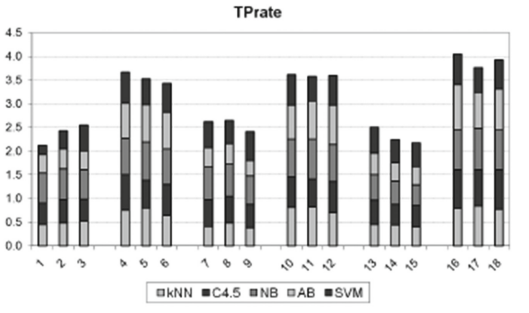
Dataset	#Examples	#Attributes	IR	IAR
Chess_IR5	2002	37	5	54
Ecoli_om_remainder_binary	336	8	15.8	42
Ecoli_imu_remainder_binary	336	8	8.6	42
Glass_VWFP_binary	214	10	11.59	21
German_IR10	769	21	10.14	37

The results are presented in Fig. 2. For viewing purposes, we have numbered the different methods from 1 to 18; please refer to the legend for identification. Each bar in the diagrams represents the overall average score (under the specific metric) obtained by all five classifiers, using the corresponding method. For example – in diagram (b), the first bar represents the overall average TP_{rate} obtained by all five classifiers on all data sets, under imbalance conditions, while the fourth bar represents the overall average TP_{rate} obtained by all five classifiers on all data sets obtained by ECSB using BAcc as fitness measure and CS as cost-sensitive strategy.

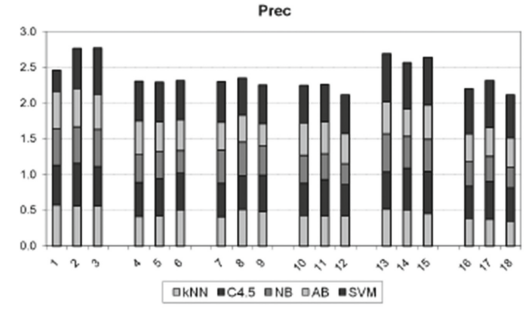
Several remarks can be made regarding these results: (1) using balanced metrics as fitness measures, such as GM or BAcc, produces significant improvements in the TP_{rate} (second and fourth groups in Fig. 2(b); (2) FM is not effective as fitness measure (third group in all diagrams); (3) the linear combination between TP_{rate} and TN_{rate} ($\alpha = 0.7$) as fitness function does not improve TP_{rate} significantly (fifth group in Fig. 2(b)), but instead it improves Prec (fifth group in Fig. 2(c)); (4) the linear combination between recall and precision ($\alpha = 0.7$) as fitness score yields the most important improvement in TP_{rate} (last group in Fig. 2(b)), but it degrades precision (Fig. 2(c)) – since $\alpha = 0.7$, more importance is given to improving recall than to precision; (5) for the SVM, both the TP_{rate} and the precision are significantly improved through the ECSB method (Fig. 2(b) and (c), the top portion of the bars); (6) out of the three cost-sensitive strategies



(a)



(b)



(c)

1 – Base
2 – Base+SMOTE
3 – ECSBT
4 – ECSB(CS, BAcc)
5 – ECSB(CSr, BAcc)
6 – ECSB(MC, BAcc)
7 – ECSB(CS, FM)
8 – ECSB(CSr, FM)
9 – ECSB(MC, FM)
10 – ECSB(CS, GM)
11 – ECSB(CSr, GM)
12 – ECSB(MC, GM)
13 – ECSB(CS, LIN)
14 – ECSB(CSr, LIN)
15 – ECSB(MC, LIN)
16 – ECSB(CS, PLIN)
17 – ECSB(CSr, PLIN)
18 – ECSB(MC, PLIN)

Fig. 2. Balanced accuracy, TP_{rate} and Precision obtained by the various methods on the large IR, small IAR data

evaluated, the most successful is CS (the first bar in each group from the second to the last), i.e. predict the class with minimum expected misclassification cost.

Therefore, balanced metrics (except FM) are generally appropriate as fitness measures for ECSB in imbalanced problems; when the recall is of utmost importance (e.g. medical diagnosis), using the linear combination between recall and precision, with a high value for α, is appropriate; this is also suitable when both precision and recall (TP_{rate}) are important (e.g. credit risk assessment), but with a lower value for α. Cost-sensitive prediction is the most appropriate strategy to employ.

4.3 Comparative Analysis with Evolutionary Under-Sampling

A second analysis was performed on a set of 28 imbalanced benchmark problems from [11], in order to compare our results with the performance of the Evolutionary Under-Sampling (EUS) strategy presented there. EUS has been shown to produce superior results when compared to state-of-the-art under-sampling methods, making it a good candidate for imbalanced data sets, especially with a high imbalance ratio among the classes. In this set of experiments, we have employed CS as cost-sensitive strategy and GM as fitness function – because it is the function employed in the most successful EUS model. We have also considered in the comparison the classifier with default settings (*Base*), the classifier with SMOTE and default settings (Base+*SMOTE*) and the classifier with tuned parameter values (*ECSBT*).

The results of this second analysis are shown in Tables 4 and 5. It can be observed that ECSB significantly boosts the performance of classifiers when compared to their behavior on the original problem (except for the AUC for AdaBoost.M1 – Table 5); on the average, there is ∼25% relative improvement on the GM and ∼5% on the AUC; the most significant improvements have been obtained for the SVM classifier (∼86% relative improvement on GM and 16% on AUC). Also, it yields significant improvements over SMOTE and ECSBT (∼17% and ∼14%, respectively, relative improvement on GM and ∼5% and ∼2%, respectively, on AUC). Slight improvements over the best EUS method

Table 4. Average GM (with standard deviations) obtained by the various methods

Geometric Mean (GM)										
	Best EUS		Base		Base+SMOTE		ECSBT		ECSB	
	mean	*stddev*	*mean*	*stddev*	*mean*	*stddev*	*mean*	*stddev*	*mean*	*stddev*
kNN	.797	.169	.731	.225	.744	.218	.762	.230	.817	.173
C4.5			.660	.317	.716	.254	.635	.307	.796	.179
NB			.754	.202	.771	.164	.754	.202	.814	.129
AB			.640	.314	.658	.306	.619	.323	.798	.188
SVM			.431	.401	.558	.358	.750	.213	.803	.184

Table 5. AUC (with standard deviations) obtained by the various methods

Area Under tin-Curve (AUC)										
	Best EUS		Base		Base+ SMOTE		ECSBT		ECSB	
	mean	*stddev*	*mean*	*stddev*	*mean*	*stddev*	*mean*	*stddev*	*mean*	*stddev*
kNN	.809	.170	.803	.144	.803	.144	.848	.140	.867	.128
C4.5			.797	.147	.797	.147	.786	.157	.830	.125
NB			.873	.110	.873	.110	.874	.111	.874	.111
AB			.892	.15	892	.105	.891	.098	.878	.121
SVM			.714	.175	.714	.175	.790	.143	.830	.132

have also been observed (i.e. the specialization of EUS which achieved the best performance in the above cited work): up to 9% relative improvement in AUC.

5 Conclusions and Future Work

Classification under imbalanced conditions is one of the current challenges in data mining research, triggered by the needs of specific application domains. All traditional algorithms are affected to some extent by the class imbalance problem. Also, the correct choice of the metric (or combination of metrics) to assess – and ultimately improve, is essential for the success of a data mining effort in such areas, since most of the time improving one metric degrades others.

A series of methods which deal with the class imbalance have been proposed in the literature over the last years. Sampling strategies are important because they can be used as pre-processing strategies. However, some approaches are difficult to employ by a less experienced user – e.g. some require to set the amount of sampling. Most importantly, to maximize their effect, they need to be matched to the specific classifier employed. Modifications to basic algorithms have also been proposed in the literature, with good performance improvements, but each is restricted to a specific class of techniques.

In this paper we propose a general hybrid strategy for improving the performance of classifiers in imbalanced problems. The method, Evolutionary Cost-Sensitive Balancing (ECSB), is a meta-approach, which can be employed with any error-reduction classifier. Two strategies are followed by the method simultaneously: tune the base classifier's parameters and use a cost-sensitive meta-classifier to adapt to the imbalance. A great advantage of the method, besides its generality, is that it needs little knowledge of the base classifier; instead, it requires specific knowledge of the domain to select the appropriate fitness measure.

We have performed several evaluations on benchmark data, to compare ECSB with current state of the art strategies for imbalanced classification. The results have demonstrated the following:

- ECSB significantly improves the performance of the base classifiers, achieving superior results to sampling with SMOTE or adapting the algorithm to the imbalance via evolutionary parameter selection;
- ECSB achieves superior results to current prominent approaches in literature: SMOTE and Evolutionary Under-Sampling;
- the most successful cost-sensitive strategy is predicting the class with minimum expected misclassification cost, instead of the most likely class (CS);
- balanced metrics are generally appropriate as fitness functions (except for the F-measure).

Our current focus is on adding an extra layer to the genetic search component, which will focus on finding the most suitable GA parameters for the given problem.

Acknowledgement. The work of the authors is supported by European Social Fund, via Programme POSDRU, DMI 1.5, ID 137516 – PARTING

References

1. Aliamiri A: Statistical Methods for Unexploded Ordnance Discrimination. PhD Thesis. Department of Electrical and Computer Engineering. Northeastern University. Boston, MA (2006)
2. Barandela, R., Sanchez, J.S., Garcia, V., Rangel, E.: Strategies for learning in class imbalance problems. Pattern Recogn. **36**(3), 849–85 (2003)
3. Batista, G.E.A.P.A, Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor. Newslett. **6**(1), 20—29 (2004). doi:10.1145/1007730.1007735
4. Brodersen, K.H., Ong, C.S., Stephen, K.E., Buhmann, J.M.: The balanced accuracy and its posterior distribution. In: Proceedings of the 20th International Conference on Pattern Recognition, pp. 3121–3124 (2010)
5. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
6. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEboost: improving prediction of the minority class in boosting. In: Proceedings of the Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 107—119 (2003)
7. Chawla, N.V., Japkowicz, N., Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. SIGKDD Explor. **6**(1), 1–6 (2004)
8. Chawla, N.: Data Mining from Imbalanced Data Sets: An Overview. Data Mining and Knowledge Discovery Handbook. Springer, US (2006)
9. Derderian, K.: General Genetic Algorithm Tool (2002), http://www.karnig.co.uk/ga/ggat.html
10. Domingos, P.: MetaCost: a general method for making classifiers cost-sensitive. In: Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, pp. 155–164. ACM Press (1999)
11. Garcia, S., Herrera, F.: Evolutionary undersampling for classification with imbalanced datasets: proposals and taxonomy. Evol. Comput. **17**(3), 275–306 (2009)

12. Grzymala-Busse, J.W., Stefanowski, J., Wilk, S.: A comparison of two approaches to data mining from imbalanced data. J. Intell. Manuf. **16**, 565–573 (2005)
13. Guo, H., Viktor, H.L.: Learning from imbalanced data sets with boosting and data generation: the databoost-IM approach. Sigkdd Explor. **6**, 30–39 (2004)
14. Hart, P.E.: The condensed nearest neighbor rule. IEEE Trans. Inf. Theory **IT-14**, 515—516 (1968)
15. Huang, K., Yang, H., King, I., Lyu, M.R.: Imbalanced learning with a biased minimax probability machine. IEEE Trans. Syst. Man Cybern. B Cybern. **36**(4), 913–923 (2006)
16. Japkowicz, N., Stephen, S.: The class imbalance problem: a systematic study. Intell. Data Anal. J. **6**(5), 429–449 (2002)
17. Kubat, M., Matwin, S.: Addressing the course of imbalanced training sets: one-sided selection. In: ICML, pp. 179—186 (1997)
18. Laurikkala, J.: Improving Identification of Difficult Small Classes by Balancing Class Distribution. Technical Report A-2001-2. University of Tampere (2001)
19. Lemnaru, C., Potolea, R.: Imbalanced Classification Problems: Systematic Study. Issues and Best Practices. LNBIP, vol. 102, pp. 35–50 (2012)
20. Lin, Y., Lee, Y., Wahba, G.: Support vector machines for classification in nonstandard situations. Mach. Learn. **46**, 191–202 (2002)
21. Liu, B., Ma, Y., Wong, C.K.: Improving an association rule based classifier. In: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, pp. 504–509 (2000)
22. Liu, W., Chawlam, S., Cieslak, D., Chawla, N.: A robust decision tree algorithms for imbalanced data sets. In: Proceedings of the Tenth SIAM International Conference on Data Mining, pp. 766–777 (2010)
23. Liu, W., Chawla, S.: Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets. Advances in Knowledge Discovery and Data Mining. LNCS, vol. 6635, pp. 345–356 (2011)
24. Quinlan, J.R.: Improved estimates for the accuracy of small disjuncts. Mach. Learn. **6**, 93–98 (1991)
25. Sun, Y., Kamel, M.S., Wong, A.K.C., Wang, Y.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recogn. **40**(12), 3358–3378 (2007)
26. Tian, J., Gu, H., Liu, W.: Imbalanced classification using support vector machine ensemble. Neural Comput. Appl. **20**(2), 203–209 (2011)
27. Tomek, I.: Two modifications of CNN. IEEE Trans. Syst. Man Commun. **SMC-6**, 769—772 (1976)
28. Turney, P.: Types of cost in inductive concept learning. In: Proceedings of the Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning. Stanford University, California (2000)
29. Visa, S., Ralescu, A.: Issues in mining imbalanced data sets-a review paper. In: Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference, pp. 67–73 (2005)
30. Weiss, G.M., Provost, F.: The Effect of Class Distribution on Classifier Learning: An Empirical Study. Technical Report ML-TR-44. Department of Computer Science, Rutgers University (2001)
31. Weiss, G.M., Provost, F.: Learning when training data are costly: the effect of class distribution on tree induction. J. Artif. Intell. Res. **19**, 315–354 (2003)
32. Weiss, G.: Mining with rarity: a unifying framework. SIGKDD Explor. **6**(1), 7—19 (2004)
33. Williams, D., Myers, V., Silvious, M.: Mine classification with imbalanced data. IEEE Geosci. Remote Sens. Lett. **6**(3), 528–532 (2009)

34. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
35. Wu, G., Chang, E.Y.: Class-boundary alignment for imbalanced dataset learning. In: Proceedings of the ICML 2003 Workshop on Learning from Imbalanced Data Sets (2003)
36. Zadrozny, B., Elkan, C.: Learning and making decisions when costs and probabilities are both unknown. In: Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, pp. 204–213 (2001)
37. Zhou, Z.H., Liu, X.Y.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans. Knowl. Data Eng. **18**(1), 63–77 (2006)

Balancing the Subtours for Multiple TSP Approached with ACS: Clustering-Based Approaches Vs. MinMax Formulation

Raluca Necula[1(✉)], Madalina Raschip[2], and Mihaela Breaban[1]

[1] Alexandru Ioan Cuza University, Iasi, Romania
{raluca.necula,pmihaela}@info.uaic.ro
[2] University of Neuchatel, Neuchatel, Switzerland
madalina.raschip@unine.ch

Abstract. The algorithms belonging to the Ant Colony Optimization metaheuristic can be naturally applied to shortest path related problems. Although not so intensively studied as TSP, the multiple traveling salesman problem (multiple-TSP) is a straightforward extension of TSP of practical importance, requiring more than one salesman to be used for covering the whole set of cities. This work tackles the MinMax formulation of multiple-TSP, which aims at obtaining balanced subtours for the salesmen. An Ant Colony System that follows the MinMax formulation is proposed. Two other algorithms combining K-Means and Fuzzy C-Means clustering with Ant Colony Systems are described. The experimental analysis investigates the performance of the proposed algorithms from a bi-objective perspective, counting for the total length/cost of a solution and its balancing degree measured as the amplitude of its subtours.

Keywords: Multiple-TSP · Ant colony optimization · Fuzzy clustering · Hybridization

1 Introduction

The single-depot multiple-TSP problem can be stated as follows. There are m salesmen who must visit a number of cities, n. Each city must be visited exactly once by a salesman and each salesman must start and end its subtour at the same depot. The objective is to minimize the total distance traveled by the salesmen.

Derived from the well-known TSP problem, multiple-TSP is more difficult to solve, since in addition it requires to determine the optimal allocation of cities to the subtours of salesmen such that each city is visited only once by one salesman and the total cost of the traveled subtours is minimized. Multiple-TSP gained less interest in the literature than TSP, although it is more suited for real-world problems related to scheduling and routing. In [1], a comprehensive review of applications for multiple-TSP is presented, ranging from school bus

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9_15

routing problem to mission planning, which arises in the case of autonomous mobile robots.

Since multiple-TSP is a NP-hard problem, which is difficult to solve, exact methods cannot be successfully applied for solving large size instances of multiple-TSP in a reasonable time. Numerous approaches were developed in literature based on deterministic or probabilistic heuristics [1]. Encouraged by their initial success when applied to the standard TSP, metaheuristics like ant colony optimization were proposed in the literature for solving multiple-TSP. Such a first method is introduced in [2], where the authors propose an Ant System to solve the multiple-TSP with ability constraints, in which the number of cities visited by each salesman is limited.

The criterion of balancing the workloads amongst salesmen, desired in real-world scenarios, but less addressed in the literature, is considered in [3]. A tabu search heuristic as well as two exact algorithms are the approaches proposed for solving this variant of multiple-TSP. Besides these methods, other approaches like adaptive neural networks [4] or ant colony [5] were proposed in literature. In [5] an ant colony system algorithm is developed, where a team of ants construct in parallel solution to the problem. The member of a team that will make a move is chosen according to the route lengths, most of the time being the one with the minimum route length. The main idea is to move always the member of a team with the minimum route length to guarantee approximately even route lengths.

There are also papers that considers both objectives, minimizing the total distance traveled by all the salesmen, respectively minimizing the maximum distance traveled by a salesman, and supply results for both of them. A genetic algorithm is proposed in [6], where the authors present a new chromosome representation that works with the classical genetic operators for the TSP. The representation dramatically reduces the number of redundant solutions. In [7] two new metaheuristics based on artificial bee colony and invasive weed optimization are proposed, whilst in [8] a market-based solution is employed, where the problem is of assigning mobile agents to tasks. In [9] the two objectives are treated independently. The approach is based on a hybrid Max-Min Ant System algorithm, which is combined with a local improvement procedure.

A good balance of workloads among salesmen is also obtained by using clustering algorithms. In [10] a clustering algorithm which minimizes the variation of distances traveled within each cluster is proposed. A variety of evolutionary computation algorithms and paradigms for the euclidean multiple TSP are compared in [11]. The first level of optimization namely, the optimal subdivision of cities into groups is considered. The neighborhood attractor schema which is a variation of k-means influences the chromosome representation.

In this study we consider the variant of multiple-TSP, denoted as MinMax multiple-TSP, in which we aim to minimize the longest subtour of a salesman, such that to achieve fairness among salesmen. This formulation leads to obtaining balanced subtours, in which the workload among salesmen is evenly distributed. This paper is organized as follows. In Sect. 2 an integer linear programming formulation for the MinMax multiple-TSP is presented, that can be used for solving

the problem with exact methods. Section 3 summarizes the Ant Colony System, as the underlying algorithm for the proposed methods that will be described in Sect. 4. Section 5 presents the conducted experiments and the obtained results, whilst the final remarks will be concluded in Sect. 6.

2 The Single-Depot Multiple Traveling Salesman Problem

Several integer linear programming formulations for the multiple-TSP were proposed in the literature. A review of them is presented in [1]. We will refer only to the formulation for the MinMax variant of the multiple-TSP.

The multiple-TSP can be defined over a directed graph $G = (V, A)$, with V being the set of nodes and A the set of arcs, and the graph has associated a cost matrix $C = (c_{ij})$ for each arc $(i, j) \in A$. The problem is to find the optimal assignment of cities to the salesmen, such that each salesman starts and ends its subtour from the depot and the longest subtour of a salesman is minimized. Initially, all salesmen are located at the depot, considered to be the first city.

The integer linear programming formulation for MinMax multiple-TSP with n cities and m salesmen, used when solving the model with dedicated software is the following:

$$\min \quad T \tag{1}$$

$$\text{s.t.} \quad \sum_{j=2}^{n} x_{1jk} = 1,\ k = 1, .., m \tag{2}$$

$$\sum_{j=2}^{n} x_{j1k} = 1,\ k = 1, .., m \tag{3}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{m} x_{ijk} = 1,\ j = 2, .., n,\ i \neq j \tag{4}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{m} x_{ijk} = 1,\ i = 2, .., n,\ i \neq j \tag{5}$$

$$\sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{jik},\ j = 2, .., n,\ k = 1, .., m,\ i \neq j \tag{6}$$

$$u_i - u_j + (n - m) \cdot \sum_{k=1}^{m} x_{ijk} \leq n - m - 1,\ 2 \leq i \neq j \leq n \tag{7}$$

$$\sum_{(i,j) \in A} c_{ij} x_{ijk} \leq T,\ k = 1, .., m \tag{8}$$

$$x_{ijk} \in \{0, 1\},\ \forall (i, j) \in A,\ k = 1, .., m. \tag{9}$$

where x_{ijk} is a binary variable that is equal to 1 if the arc (i, j) is used in the optimal solution by the salesman k and 0 otherwise. u_i denotes the number of

nodes visited on that salesman's path from the origin to node i, for any salesman, i.e. the position of node i in a subtour.

Constraints (2) and (3) ensure that each salesman leaves, and respectively returns to the depot city. Constraint (4) denotes that in each city, except the depot, we enter only once, whilst constraint (5) denotes that from each city, except the depot, we exit only once. Constraint (6) indicates that in order to form subtours, the salesman that enters the city j, must exit from it, for each j. Constraint (7) corresponds to the subtour elimination constraint, whilst constraint (8) reflects that variable T to be higher than any subtour. Thus T represents the longest subtour, that should be minimized according to the objective function from (1).

3 The Ant Colony System

The Ant Colony Optimization metaheuristic belongs to the class of swarm intelligence methods and draws its inspiration from the way real ants succeed in finding the paths between their nest and food sources. This is achieved by ants laying down pheromone trails throughout their path, used as a form of indirect communication. The variation in the amount of pheromone deposited in the paths is an indicator of the quality of the route selected to reach the food. Although initially designed to solve shortest-path problems in graphs, the application of ant colony algorithms is not restricted to this class of problems; they have been applied successfully to other various combinatorial optimization problems as well. The proposed algorithms described in the next section are based on the Ant Colony System (ACS) [12], initially designed for solving the TSP.

In ACS, several ants are placed in the nodes of the graph representing the TSP instance. At each iteration, each ant builds a tour in the graph; on each edge it traverses, the ant lays a constant quantity of pheromone. At the end of each iteration, after all ants built their tours, the best solution recorded so far, called global solution, is updated (if necessary). Then, the edges on the path given by the global solution receive an extra quantity of pheromone which is inverse proportional with the cost of the global solution. Three important phases are thus involved in this optimization process: node selection at route construction, local pheromone update performed each time an ant traverses an edge and global pheromone update that highlights the route with the minimum cost identified so far.

4 Algorithms Investigated for Multiple-TSP

4.1 Problem Decomposition with K-Means Followed by ACS for TSP (kM-ACS)

A first algorithm that resorts to clustering for solving the multiple-TSP is $kM - ACS$ and it was introduced in a previous work of the authors [13]. The main idea behind this approach is to first divide the initial problem into several

groups of cities by using K-Means clustering algorithm, then solve each formed subgroup as an individual TSP problem with the Ant Colony System. In this way the original multiple-TSP instance is transformed into several smaller subproblems, that represent disjoint subset of cities to be visited in a certain order by the m salesmen. The two coordinates in the cartesian plan that each city has, serves as two numerical attributes to be used when performing the cluster analysis. The K-Means algorithm was chosen due to the fact that it is one of the most known clustering algorithms and it generates groups of equal volumes. This characteristic is desirable since we aim to obtain subtours of approximately equal cost. The final solution to the initial multiple-TSP instance is obtained by aggregating the solutions found during the m independently runs of the ACS, which will represent subtours to be assigned to the m salesmen.

4.2 Fuzzy C-Means Combined with ACS with Global-Solution Pheromone Update (fuzzy g-ACS)

Another clustering based algorithm uses a global-solution pheromone update and it incorporates the fuzzy C-Means algorithm. Like the previous algorithm, it employs the similar idea of using clustering for the grouping of cities.

Fuzzy C-Means clustering algorithm does not compute disjoint partitions of data points - as in the case of k-means algorithm - but rather a degree of flexibility is introduced, by allowing an observation to belong to more than one group with a certain probability. The output of the fuzzy C-Means is a matrix of membership levels (i.e. probabilities), giving for each data point the probability of belonging to every cluster. In the case of our studied problem, the number of clusters is equal to the number of salesmen from the multiple-TSP instance. Given the cost matrix encoding the distances between each pair of cities from the multiple-TSP instance, each city can be seen as an item in the data set, having two numerical attributes corresponding to its two coordinates.

Since the depot city need to be included in the subtour of each salesman, the Fuzzy C-Means algorithm need to be adapted so as to take into account the position of the depot city. To this end, a change was done when computing the position of the centroid for each partition, such that all the centroids will be biased towards the location of the depot city.

The corresponding nearest neighbour subtours, needed in order to compute the value for the initial pheromone level, are constructed by using the crisp partitions obtained after running Fuzzy C-Means. To get these crisp partitions each element is assigned to the cluster for which it has the highest membership level, so that each element will belong to only one partition. Therefore, each subtour is constructed within the cities belonging to the same partition and by always choosing next the closest unvisited city. For each ant, initially all the salesmen are positioned in the depot city. Then for each possible partition corresponding to a salesman, a probability is computed according to the crisp clusters obtained at a precedent step. For the k salesman, this probability is calculated as the division between the number of cities assigned to the k crisp cluster and the total number of cities from the multiple-TSP instance.

This probability will be used when deciding which salesman should perform the next move, such that each salesman will be chosen in a probabilistic way by some type of roulette wheel selection. The rationale behind this selection mechanism is to reflect for each salesman the number of expected cities that it should visit. In this way, a salesman with a greater number of assigned cities will have a higher probability of being chosen. Subsequently, when the selected salesman decides which node to traverse next based on its current position, to the standard ACS transition rule it is added the membership level obtained after running Fuzzy C-Means algorithm. More exactly, in the subtour construction process, the next node to be visited according to the pseudo-random-proportional rule is given by the following equation:

$$s = \begin{cases} \arg\max_{s \in C} \tau(r,s) \cdot \eta^{\beta}(r,s) \cdot mem^{\beta}(s,k), & \text{if } rand(0,1) < q_0 \\ S, & \text{otherwise} \end{cases}$$

where q_0 is a parameter and $s \in C$ denotes that node s is from the candidate set C, that consists of nodes not visited yet. $mem(s,k)$ expresses the membership level of the city s to the k cluster corresponding to the subtour of the k salesman which is under construction, and S is a random variable with the probability distribution given by equation:

$$p(r,s) = \frac{\tau(r,s) \cdot \eta^{\beta}(r,s) \cdot mem^{\beta}(s,k)}{\sum_{u \in C} \tau(r,u) \cdot \eta^{\beta}(r,u) \cdot mem^{\beta}(u,k)} \tag{10}$$

The pheromone level on the traversed connections is updated regardless of the salesman which visited the edge. The construction process of the subtours ends when the candidate set of nodes is empty, meaning there are no left unvisited cities to be chosen in the route selection step. The global best ant, for which the global pheromone update will be applied, is chosen as the one with the minimum value for the total cost of its m subtours. Then the edges (r,s) belonging to the subtours of the globally best ant receives an additional amount of pheromone like in the standard ACS:

$$\tau(r,s) = (1-\rho) \cdot \tau(r,s) + \rho \cdot \Delta\tau(r,s) \tag{11}$$

where

$$\Delta\tau(r,s) = \begin{cases} (L_{gb})^{-1}, & \text{if } (r,s) \in \text{ subtours of globally best ant} \\ 0, & \text{otherwise} \end{cases}$$

with ρ being the pheromone evaporation rate and L_{gb} being the total cost of the global solution. In the experimental section this algorithm is denoted as $fuzzy\ g-ACS$.

4.3 Fuzzy C-Means Combined with MinMax ACS with Global-Solution Pheromone Update (fuzzy g-MinMaxACS)

This ACS based algorithm, denoted as $fuzzy\ g-MinMaxACS$, is similar with the prior described one, using as well Fuzzy C-Means clustering, but differs in

the criterion used for deciding the best ant. More exactly, the ant with the smallest cost for its longest subtour, which corresponds to the MinMax criterion, is considered as the global best ant when performing the global pheromone update. Therefore, during the global pheromone update, all the edges visited by the global best ant will receive the same deposit of pheromone, equal to the length of the longest subtour from the global best solution.

4.4 MinMax ACS with Global-Solution Pheromone Update (g-MinMaxACS)

Another investigated algorithm relying on ACS, uses a global-solution pheromone update, along with considering the MinMax criterion for achieving balanced subtours. Initially, for a problem instance with m salesmen, m salesmen are placed at the depot city. Then at each iteration, several ants construct solutions for the multiple-TSP instance, by employing a team of m ants. In this way, the solution built by an ant will be composed of m individual subtours, one corresponding to each salesman.

Since a city should be included only in the subtour of one salesman, several salesmen compete towards building its subtour in an iteration of the algorithm. To this end, at each step, the next salesman that will visit a city is chosen randomly from the set of m available salesmen. After being chosen, the salesman is replaced in the set of possible salesmen so that next time it can get another chance of being selected (i.e. selection with replacement). The selected salesman chooses the city to visit in its subtour according to the pseudo-random-proportional rule from the standard ACS algorithm. The next city is selected among the nodes from the candidate set, comprising of nodes not visited yet in any of the m subtours under construction. This process continues until there are no remaining unvisited nodes, meaning that a complete candidate solution consisting of m subtours was constructed. During the local pheromone update, the pheromone level on the visited edges is updated regardless of which salesman traverses it.

For computing the value of the initial pheromone level, the equivalent nearest neighbour tour for multiple-TSP is constructed by randomly choosing the salesman that will perform the next move. Then the city to be visited by the salesman is selected by a greedy choice of the closest unvisited node, relative to the current node. After all the ants finish to construct a complete solution, a global pheromone update takes place, reinforcing with additional pheromone the connections that belong to the global best ant. The amount of deposited pheromone is given by the cost of the longest subtour of the global best solution. Among the ants from the current iteration, the best iteration ant is chosen as being the one with the smallest cost for its longest subtour. Based on the best iteration ant, the global best so far ant is updated if necessary.

This version of ACS algorithm will be referred in the experimental section as $g - MinMaxACS$.

5 Experiments

5.1 Problem Instances

Since there are no publicly available benchmarks for multiple-TSP, as it is TSPLIB [1] for TSP, we had to construct our own set of multiple-TSP instances. The experimental analysis in this paper is conducted on 8 problem instances which were created based on 2 TSPLIB instances: for each TSPLIB instance we specified 4 different values for the number of salesmen. The 8 multiple-TSP instances, were solved with exact methods by using CPLEX.[2] The description of the multiple-TSP instances [3] and the obtained results are publicly available as multiple-TSPLIB.[4]

As mentioned in Sect. 2, where we introduced the linear programming formulation for the MinMax variant of multiple-TSP, the objective is to minimize the cost of the longest subtour. This is different from the formulation of the standard multiple-TSP, which aims to minimize the total cost of the traveled subtours. The formulation we considered for the multiple-TSP is necessary such as to obtain balanced subtours as much as possible. In fact, when no such constraint is imposed, the solutions obtained by CPLEX are highly unbalanced: most of the cities are visited by a salesman, while the rest of the salesmen visit only a small fraction of cities. This corresponds to an uneven distribution of work among the salesmen, which is undesirable in real-world scenarios. At the same time, the MinMax version of multiple-TSP is more difficult to solve by CPLEX than the standard multiple-TSP. For the eil51-m5 problem instance, in case of the standard multiple-TSP formulation, CPLEX found an optimal solution in at most 1 min, while in case of the MinMax multiple-TSP formulation the reported solution was obtained with CPLEX after 120 hours. In comparison, one run of any algorithm presented in this paper required on average 10 seconds in a single-threaded implementation. All the runs were performed on a PC with 8 GB RAM, processor Intel Core i5-4590S 3.00 GHz using 4 processing units (multi-threaded implementation of CPLEX). However our purpose is not to compare our investigated algorithms with the CPLEX solver, since for the execution of the ACS algorithms we set as limit a maximum number of iterations, whilst in the case of the CPLEX solver we set a high time limit as stopping criterion. In the experimental section of our paper we report the solutions obtained by CPLEX, just to have an overview of the quality of solutions found by our proposed approaches.

5.2 Parameters Settings

For the parameters of the ACS algorithm we used the recommended values for the standard ACS [12], more specifically we used $q_0 = 0.9$, $\alpha = 0.1$, $\rho = 0.1$, $\beta = 2.0$ in all the algorithms presented in Sect. 4. For the $kM-ACS$, the number

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.
[2] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.
[3] www.infoiasi.ro/~mtsplib.
[4] www.infoiasi.ro/~mtsplib/MinMaxMTSP.

of used ants was 10, for each of the m TSP subproblems being solved, leading to a total of $10 \cdot m$ ants employed for solving the initial multiple-TSP instance. For all the other algorithms we used the same number of $10 \cdot m$ ants as follows: 10 groups of ants of size m build independently in a single run of the algorithm complete solutions to the multiple-TSP.

As a stopping criterion for the ACS based algorithms we set a maximum number of iterations. More exactly, each algorithm was run for 1400 iterations in case of eil51 and 1800 iterations for eil76 problem instances. For the conducted experiments we performed 50 runs for each investigated algorithm.

K-Means is an iterative method, which aims to minimize the within-cluster variance and the obtained clusters are highly dependent on the initialization step. Thus, we performed 30 runs of K-Means on the same multiple-TSP instance to split the n cities into m groups and selected the solution which provided the lowest sum of the within-cluster variances. However, in the case of fuzzy clustering algorithms, we didn't impose a certain grouping of cities to be used in all the runs, but instead different membership probabilities were used in the conducted runs, such that to exploit the random nature of Fuzzy C-Means.

5.3 Results

For assessing the performance of the investigated algorithms, we report their obtained values for the total cost of the traveled subtours and for the cost of the longest subtour. Though we report the obtained results for both objectives, the main focus is on the MinMax criterion. We specify the values for the classic objective (i.e. total cost) only for a better comparison. As a measure for the balancing degree of the subtours, we used amplitude, computed as the difference between the cost of the longest subtour and the cost of the shortest subtour of the solution.

Comparing Fuzzy g-ACS and Fuzzy g-MinMaxACS. Although both $fuzzy\ g-ACS$ and $fuzzy\ g-MinMaxACS$ algorithms resort to clustering, they exhibit different behaviour according to the multiple-TSP objectives: total cost and the balancing of subtours. For Fig. 1, we post-processed the 50 solutions obtained by the two fuzzy clustering based algorithms and extracted only the Pareto non-dominated solutions according to the two objectives. For the considered multiple-TSP instances we illustrate the values for the two objectives: the values on the horizontal axis corresponds to the total cost of the traveled subtours, whilst the values on the vertical axis corresponds to the amplitude of the traveled subtours. It can be noticed that for both eil51 and eil76 instances, $fuzzy\ g-ACS$ obtains lower values for the total cost, but that are highly unbalanced, whereas, on the other hand, $fuzzy\ g-MinMaxACS$ obtains good balanced solutions but of higher total costs. This observation actually indicate the fact that these two objectives, of minimizing the total cost of the traveled subtours and of minimizing the cost of the longest subtour are conflicting ones: improving one objective, degrades the obtained values for the other objective. This leads to

multiple-TSP being an inherently bi-criteria problem. As a matter of fact, our future studies will be focused on tackling the multiple-TSP from a bi-criteria perspective. Since $fuzzy\ g-ACS$ obtained worse results regarding the MinMax criterion, in the following we will only refer to $fuzzy\ g-MinMaxACS$ as a fuzzy clustering based algorithm.

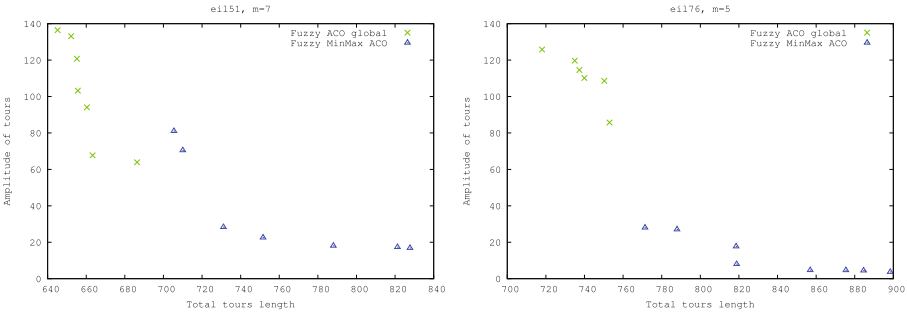


Fig. 1. A comparison between the fuzzy ACS global and fuzzy MinMax ACS algorithms for eil51-m7 and eil76-m5 instances

From our experiments it resulted that in most of the multiple-TSP instances, $fuzzy\ g-ACS$ obtains better values for the total cost of the traveled subtours, whilst $fuzzy\ g-MinMaxACS$ succeeds in attaining good values for the amplitude of the subtours, achieving balanced subtours. This fact can be explained by their different mechanism of selecting the best ant and enforcing the global best so far solution. While in $fuzzy\ g-ACS$ the best ant is selected to be the one with the minimum total cost of the traveled subtours, in $fuzzy\ g-MinMaxACS$ the global best solution and the additional pheromone deposit take into account the cost of the longest subtour.

Algorithms Comparison. Table 1 reports the optimal values for the longest subtour of minimal cost (MinMax) obtained by CPLEX, along with the values obtained by our investigated algorithms. For the ACS based algorithms the MinMax value represents the minimum value obtained for the longest subtour from the found solutions. When CPLEX was stopped before reaching an optimal solution, the upper (corresponding to the best known solution) and the lower bound on the cost of the longest subtour are given. The best value from a row is highlighted with boldface. From the analysis of these values, among all the ACS algorithms, $g-MinMaxACS$ achieves the best values for the longest subtour, corresponding to the MinMax criterion. In case of eil76-m5 multiple-TSP instance it even gets a better value for MinMax than the best known solution found by CPLEX.

Figures 2, 3, 4 and 5 depicts for eil51 and eil76 multiple-TSP instances the distribution for the total costs and for the cost of the longest subtour, and

Table 1. The performance of the ACS variants for the eil51 and eil76 instances using MinMax as measure

m	CPLEX optimum	kM-ACS	g-MinMaxACS	fuzzy g-MinMaxACS
eil51				
2	222.73	254.42	**226.54**	231.05
3	[150.70, 159.57]	176.16	**164.64**	167.06
5	[96.91, 123.96]	169.03	**127.86**	133.12
7	[72.42, 112.46]	129.80	**116.05**	118.26
eil76				
2	280.85	301.74	**288.96**	290.37
3	[186.34, 197.34]	266.92	**214.92**	221.05
5	[116.02, 173.18]	176.31	**160.49**	167.24
7	[88.35, 139.62]	158.03	**145.28**	150.37

confidence intervals for the mean of the total costs and for the mean of the longest subtour, for the solutions obtained by each algorithm. On each plot there are four groups corresponding to the different number of m (2,3,5 and 7). It can be noticed that with the increase in the number of salesmen, the total cost increases, whilst the length of longest subtour decreases, reflecting the division of work among more salesmen. For each of these four groups there are indicated the obtained values for each of the three considered algorithms ($kM-ACS$, $g-MinMaxACS$ and $fuzzy\ g-MinMaxACS$) together with the best known solution, obtained with CPLEX, that serves as lower bound for the ACS-based algorithms.

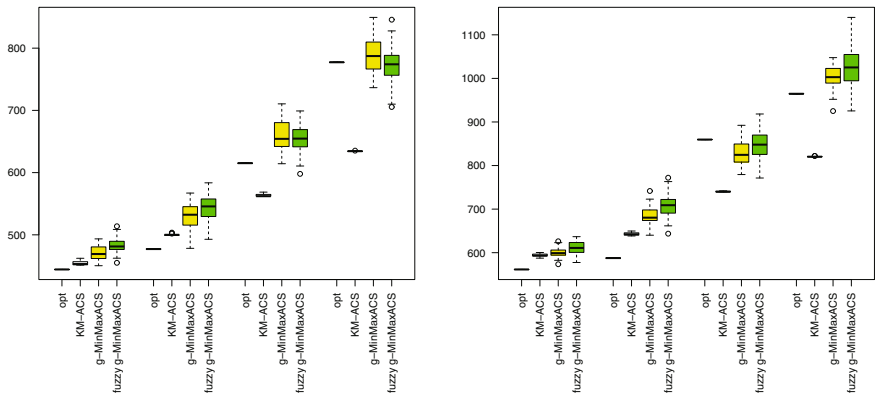


Fig. 2. Results obtained in 50 runs for the total cost: (a) eil51, (b) eil76; the groups correspond to different settings for m: 2, 3, 5, 7

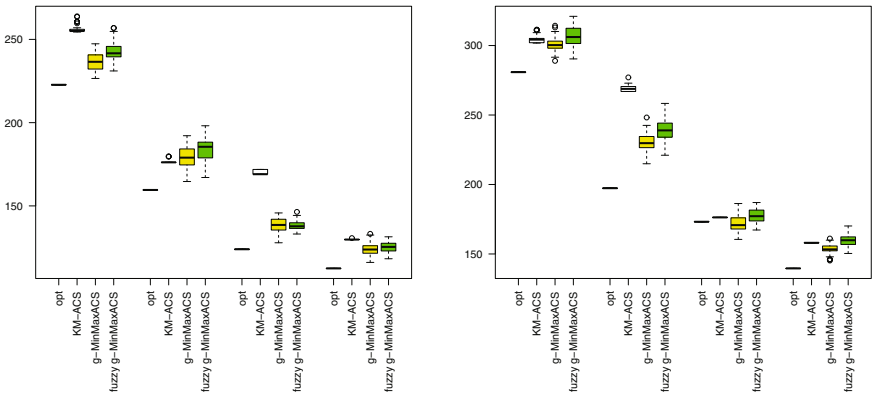


Fig. 3. Results obtained in 50 runs for the longest subtour: (a) eil51, (b) eil76; the groups correspond to different settings for m: 2, 3, 5, 7

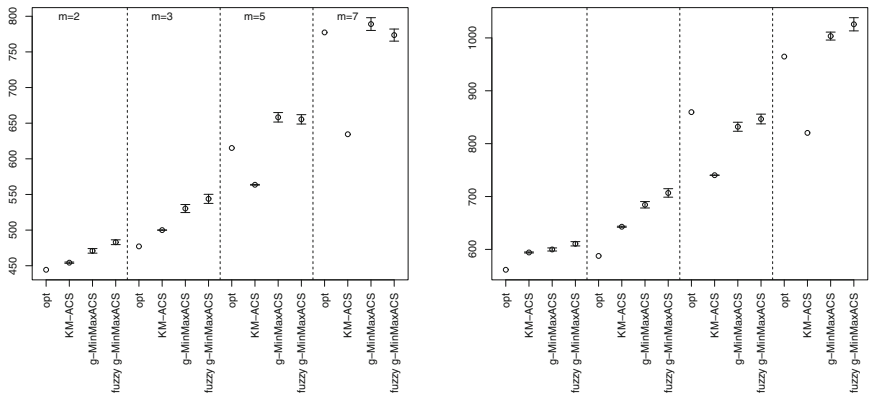


Fig. 4. Results obtained in 50 runs for the (a) eil51, (b) eil76: confidence intervals for the means computed for total costs; the groups correspond to different settings for m: 2, 3, 5, 7

Analyzing the boxplots, but also also from the size of the confidence intervals, $kM-ACS$ is the most stable algorithm. However, it does not achieve such good values as $g-MinMaxACS$ and $fuzzy\ g-MinMaxACS$ for the length of the longest subtour. Figure 2 that illustrates the boxplots for the total cost, indicates the low values obtained for the standard deviation in case of K-Means, such that the obtained solutions are not so diverse. Figure 5 shows that between the two clustering algorithms, $fuzzy\ g-MinMaxACS$ obtains better values than $kM-ACS$ regarding the cost of the longest subtour. Although both these algorithms attempts to create groups of cities of equal volumes, it seems that the flexible nature of Fuzzy C-Means helps in that sense. In contrast to this, $kM-ACS$ seems to loss diversity when a higher number of clusters is considered, proven by the overall lower values obtained for the standard deviation of the total cost.

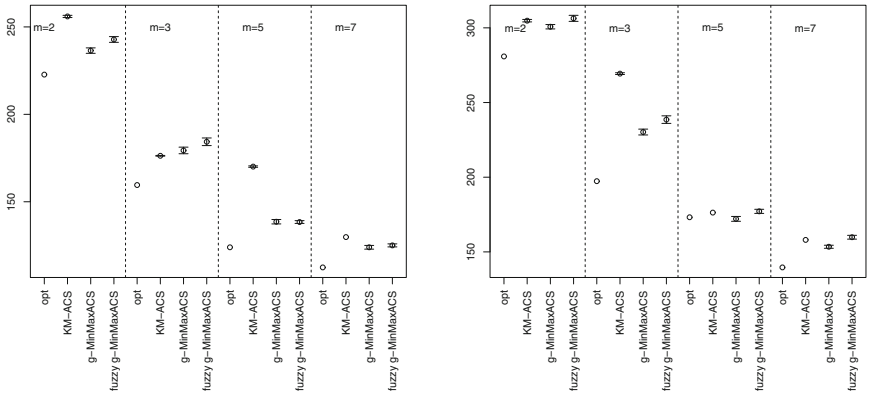


Fig. 5. Results obtained in 50 runs for the (a) eil51, (b) eil76: confidence intervals for the means computed for the longest subtour; the groups correspond to different settings for m: 2, 3, 5, 7

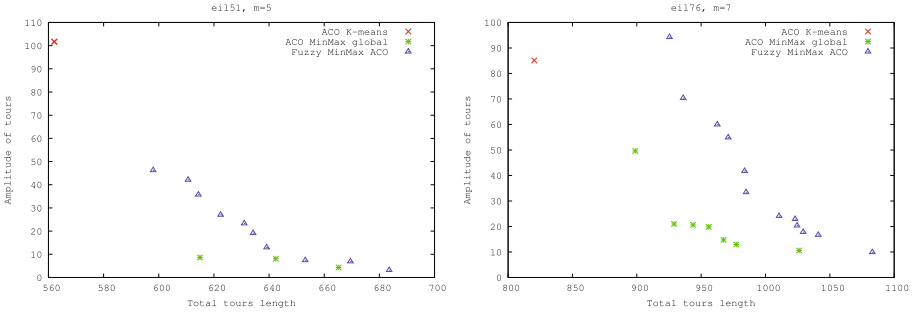


Fig. 6. A comparison between the investigated algorithms for eil51-m5 and eil76-m7 instances

A comparison of the investigated algorithms on eil51-m5 and eil76-m7 instances is illustrated in Fig. 6, that plots, like in Fig. 1, the Pareto non-dominated solutions according to the two objectives. It can be noticed the good values obtained by $g - MinMaxACS$ and $fuzzy\ g - MinMaxACS$ for the amplitude, taken as a measure for the balancing degree of the subtours. In contrast to these two algorithms, there are fewer solutions corresponding to $kM-ACS$, which are highly unbalanced. On most of the multiple-TSP instances, $g - MinMaxACS$ achieves better performance both in terms of total cost and MinMax criterion when compared to $fuzzy\ g - MinMaxACS$.

6 Conclusions

Ant colony algorithms can be easily adapted and applied to shortest path related problems as multiple-TSP. Unlike exact methods, ACO based approaches can

obtain acceptable solutions, both in matter of time and performance, which makes them more suited for real-world applications. In this paper, we tackled the MinMax variant of multiple-TSP, which aims to minimize the cost of the longest subtour. To this end, we proposed four ACS based algorithms, three of them employing clustering methods for the division of cities into equal groups, and the other one following a MinMax approach. Between the two versions of fuzzy clustering algorithms, the one that takes into account the MinMax criterion achieves better values for the cost of the longest subtour. The approach involving K-Means is the most stable algorithm, but due to its decomposition scheme, it imposes rigid boundaries on the candidate solution space and the obtained solutions are not so balanced. Among all the investigated algorithms, $g - MinMaxACS$, which follows a MinMax approach, achieves the best performance, succeeding in finding balanced subtours. As future work, our study will be directed towards applying multi-objective ACO algorithms for the bi-criteria multiple-TSP and evaluate their relative performance.

References

1. Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega **34**(3), 209–219 (2006)
2. Junjie, P., Dingwei, W.: An ant colony optimization algorithm for multiple travelling salesman problem. In: ICICIC 2006, vol. 1, pp. 210–213 (2006)
3. França, P.M., Gendreau, M., Laporte, G., Müller, F.M.: The m-traveling salesman problem with minmax objective. Transp. Sci. **29**(3), 267–275 (1995)
4. Somhom, S., Modares, A., Enkawa, T.: Competition-based neural network for the multiple travelling salesmen problem with minmax objective. Comput. Oper. Res. **26**(4), 395–407 (1999)
5. Vallivaara, I.: A team ant colony optimization algorithm for the multiple travelling salesmen problem with minmax objective. In: MIC 2008, pp. 387-392 (2008)
6. Carter, A.E., Ragsdale, C.T.: A new approach to solving the multiple traveling salesperson problem using genetic algorithms. EJOR **175**(1), 246–257 (2006)
7. Venkatesh, P., Singh, A.: Two metaheuristic approaches for the multiple traveling salesperson problem. Appl. Soft Comput. **26**, 74–89 (2015)
8. Kivelevitch, E., Cohen, K., Kumar, M.: A market-based solution to the multiple traveling salesmen problem. JIRS J. **72**(1), 21–40 (2013)
9. Liu, W., Li, S., Zhao, F., Zheng, A.: An ant colony optimization algorithm for the multiple traveling salesmen problem. In: ICIEA 2009, pp. 1533–1537 (2009)
10. Chandran, N., Narendran, T.T., Ganesh, K.: A clustering approach to solve the multiple travelling salesmen problem. Int. J. Ind. Syst. Eng. **1**(3), 372–387 (2006)
11. Sofge, D., Schultz, A., De Jong, K.: Evolutionary computational approaches to solving the multiple traveling salesman problem using a neighborhood attractor schema. In: Applications of Evolutionary Computing, pp. 153-162 (2002)
12. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
13. Necula, R., Breaban, M., Raschip, M., Performance evaluation of ant colony systems for the single-depot multiple traveling salesman problem. In: HAIS 2015, vol. 9121, pp. 257-268 (2015)

Author Index

A.-A. Tantar et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Advances in Intelligent Systems and Computing 674,
https://doi.org/10.1007/978-3-319-69710-9