Journal Subline

LNCS 8160

Ajith Abraham
Guest Editor

Transactions on Computational Science XXI

Marina L. Gavrilova · C. J. Kenneth Tan
Editors-in-Chief

Special Issue on Innovations in Nature-Inspired Computing and Applications

Springer

Marina L. Gavrilova   C.J. Kenneth Tan
Ajith Abraham (Eds.)

Transactions on Computational Science XXI

Special Issue on Innovations
in Nature-Inspired Computing and Applications

Springer

Editors-in-Chief

Marina L. Gavrilova
University of Calgary, AB, Canada
E-mail: mgavrilo@ucalgary.ca

C.J. Kenneth Tan
CloudFabriQ Ltd., London, UK
E-mail: cjtan@CloudFabriQ.com

Guest Editor

Ajith Abraham
Technical University of Ostrava, Czech Republic
and
Machine Intelligence Research Labs, Auburn, WA, USA
E-mail: ajith.abraham@ieee.org

ISSN 0302-9743 (LNCS) e-ISSN 1611-3349 (LNCS)
ISSN 1866-4733 (TCOMPSCIE) e-ISSN 1866-4741 (TCOMPSCIE)
ISBN 978-3-642-45317-5 e-ISBN 978-3-642-45318-2
DOI 10.1007/978-3-642-45318-2
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013955030

CR Subject Classification (1998): I.2, G.1.6, F.2, H.2.8, F.1, I.5, H.3

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

Printed on acid-free paper

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

- Computational Geometry
- Computational Number Theory
- Computational Physics
- Data Storage and Information Retrieval
- Data Mining and Data Warehousing
- Grid Computing
- Hardware/Software Co-design
- High-Energy Physics
- High-Performance Computing
- Numerical and Scientific Computing
- Parallel and Distributed Computing
- Reconfigurable Hardware
- Scientific Visualization
- Supercomputing
- System-on-Chip Design and Engineering

Editorial

The Transactions on Computational Science journal is part of the Springer series *Lecture Notes in Computer Science*, and is devoted to the gamut of computational science issues, from theoretical aspects to application-dependent studies and the validation of emerging technologies.

The journal focuses on original high-quality research in the realm of computational science in parallel and distributed environments, encompassing the facilitating theoretical foundations and the applications of large-scale computations and massive data processing. Practitioners and researchers share computational techniques and solutions in the area, identify new issues, and shape future directions for research, as well as enable industrial users to apply the presented techniques.

The current volume is devoted to recent advancements in the field of nature-inspired computing and applications. The issue provides an in-depth overview of current research on neurocomputing, evolutionary algorithms, swarm intelligence, artificial immune systems, membrane computing, computing with words, artificial life, and hybrid approaches. This special issue is aimed at practitioners, researchers, and post-graduate students who are engaged in developing and applying advanced nature-inspired computational techniques from both theoretical and practical points of view. Some articles are extended versions of conference papers previously published at the Nature and Biologically Inspired Computing (NaBIC) Congress, while others are direct submissions to the special issue.

We would like to extend our sincere appreciation to Guest Editor Prof. Ajith Abraham, to all of the authors for submitting their papers to this special issue, and the associate editors and referees for their valuable work. We would like to express our gratitude to the LNCS editorial staff of Springer, who supported us at every stage of the project.

It is our hope that the fine collection of papers presented in this special issue will be a valuable resource for Transactions on Computational Science readers and will stimulate further research into the vibrant area of computational science applications.

October 2013

Marina L. Gavrilova
C.J. Kenneth Tan

Special Issue on Innovations in Nature-Inspired Computing and Applications Guest Editor's Preface

Nature-inspired computation is a general term referring to computing inspired by nature. It is an emerging interdisciplinary area in computer science and due to its success in dealing with large, complex, and dynamic problems, it has become a household name for solving real-world problems. The main idea is to mimic the complex phenomena (concepts, principles, and mechanisms) occurring in nature as computational processes in order to enhance the way computation is performed from a problem solving point of view. Some of the key paradigms falling under this umbrella are neurocomputing, evolutionary algorithms, swarm intelligence, artificial immune systems, membrane computing, computing with words, artificial life, hybrid approaches, etc. Articles were selected on the basis of fundamental ideas and concepts rather than the direct usage of well-established techniques. This special issue is aimed at practitioners, researchers, and post-graduate students who are engaged in developing and applying advanced nature-inspired computational techniques from a theoretical point of view and also to solve real-world problems. It constitutes a collection of 15 articles reflecting some of the current technological innovations in the field of nature-inspired computation and its real world applications. The papers are arranged as follows.

In the first article, Veenhuis presents a novel function optimization algorithm inspired from Wikipedia, which uses a collaborative web community of authors to improve the quality of articles. The author introduces a community optimization algorithm by mimicking a collaborative web community, which edits or improves a knowledge base. The knowledge base represents the problem to be solved and the different decision variables represent different topics contained in this knowledge base. The algorithm is tested on eight well-known benchmark problems for lower as well as higher dimensions.

The diffusion of innovation theory explains how new ideas are disseminated among social system members. Sampaio et al. in the second article propose the use of evolutionary algorithms for the simulation of innovation diffusion within organizations. To overcome some of the problems inherent in the conventional evolutionary algorithm a probabilistic approach is also incorporated.

In the sequel, Jha et al. evaluate the performance of a robot by empowering it with a decision-making capability, which uses synthetic emotions. The authors attempted to make the robot perform high-profile tasks rather than menial ones so as to increase its utility.

Biogeography-based optimization is a population-based algorithm that is inspired by biogeography, which describes the immigration and emigration of species between habitats. Goel et al. in the fourth article present a land cover feature extraction technique based on the extended species abundance model of biogeography and the algorithm has been successfully tested on two different multi-spectral satellite image datasets.

In the fifth paper, Madureira et al. describe the developing issues for ant-colony system-based optimization tools to support decision-making processes and solve the problem of generating a sequence of jobs that minimizes the total weighted tardiness for a set of jobs to be processed in a single machine.

Many real-world optimization problems present themselves in a multi-objective setting (where each of the objectives portrays different aspects of the problem). Ganesan et al. in the sixth article propose the weighted sum scalarization approach using differential evolution, chaotic differential evolution, and gravitational search algorithms to generate the approximate Pareto frontier.

In the sequel, Dutta et al. present a real-coded multi-objective genetic algorithm based K- clustering method, where a genetic algorithm is exploited to search for suitable clusters and centers of clusters so that intra-cluster distance and inter-cluster distances are simultaneously optimized. The authors attempted to simultaneously tackle dimensionality reduction and optimization of objectives using the multi-objective genetic algorithm.

The scheduling problem is considered to be an NP-complete combinatorial optimization problem and during the past few decades, researchers have used different meta-heuristics to solve such complex problems. However, most of these meta-heuristic techniques require extensive parameter tuning, which is again a very hard and time-consuming task to perform. Periera et al. in the eighth article propose a case-based reasoning module to solve the parameter-tuning problem in a multi-agent scheduling system.

Díaz-Parra and Ruiz-Vanoye in the ninth paper propose a vertical transfer algorithm for solving the school bus routing problem. The vertical transfer algorithm uses the clusterization population pre-selection operator, tournament selection, crossover-k operator, and an intelligent mutation operator.

In the tenth paper, Saha et al. propose craziness-based particle swarm optimization for designing digital Infinite Impulse Response (IIR) filters. Experimental results illustrate that apart from gaining better control on cognitive and social components of the conventional particle swarm optimization algorithm, the craziness-based particle swarm optimization offers better performance.

The prisoner's dilemma game has emerged as the most promising mathematical metaphor for studying cooperation. Wang et al. conduct simulations with four different types of neighbourhood structures, and agents update their strategies by probabilistically imitating the strategies of better performing neighbours. During the evolution each agent can modify its own strategy and/or personal feature via a particle swarm optimization approach in order to improve the utility.

Polášek and Uhlár in the twelfth paper propose a method for extracting, identifying, and visualizing topics, code tiers, users, and authors in software projects. The methodology can extract topics and visualize them in 3D graphs and then developers within and outside the teams can receive and utilize visualized information from the code and apply it to their projects.

Navrat and Sabo present an approach, inspired by honey bees, that allows exploring the World Wide Web by extracting keywords relevant to current news stories. Honey bees cooperate together to select random keywords and carry them from one article to another, landing only on the articles relevant to the keyword.

In the fourteenth article, Raeesi and Kobti introduce the Variable Neighborhood Search (VNS) metaheuristic. VNS is hybridized with Differential Evolution (DE) incorporating explorative evolutionary operators and sub-populations to improve the population diversity. The algorithms are then validated on classical job shop scheduling problems.

In the final paper, Snasel et al. illustrate a growing self-organizing grid method for knowledge discovery and visualization for the analysis of emergency call-taking information systems and their data characteristics. To handle the massive data, the growing grid algorithm is implemented in a parallel environment using compute unified device architecture. Experimental results illustrate that the proposed method is very efficient.

I would like to thank our peer-reviewers for their diligent work and timely efforts. We are also grateful to the Editor-in-Chief of Springer's LNCS Transactions on Computational Science, Prof. Marina Gavrilova, University of Calgary, Canada, for her continued support and for the opportunity to organize this special issue. We hope that the readers will enjoy reading this special issue and find it useful.

Acknowledgments: This work was also performed within the framework of the IT4 Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 by operational programme 'Research and Development for Innovations' funded by the Structural Funds of the European Union and state budget of the Czech Republic, EU.

October 2013 Ajith Abraham

LNCS Transactions on Computational Science – Editorial Board

Table of Contents

Community Optimization

Christian B. Veenhuis

Berlin University of Technology
Berlin, Germany
veenhuis@googlemail.com

Abstract. In recent years a number of web-technology supported communities of humans have been developed. Such a web community is able to let emerge a collective intelligence with a higher performance in solving problems than the single members of the community. Thus, collective intelligence systems are explicitly designed to take advantage of these increased capabilities. A well-known collective intelligence system is *Wikipedia*, the web encyclopedia. It uses a collaborative web community of authors, which improves and completes the content of articles. The quality of a certain number of these articles comes close to some degree to that of a famous printed encyclopedia. Based on such successes of collective intelligence systems, the question arises, whether such a collaborative web community could also be capable of function optimization.

This paper introduces an optimization algorithm called Community Optimization (CO), which optimizes a function by simulating a collaborative web community, which edits or improves an article-base, or, more general, a knowledge-base. The knowledge-base represents the problem to be solved and is realized as a real valued vector. The different vector components (decision variables) represent different topics contained in this knowledge-base. Thus, the dimension of the problem is the number of topics to be improved by the simulated community, whereby the dimension remains static. In order to realize this, CO implements a behavioral model of collaborative human communities derived from the human behavior that can be observed within certain web communities (e.g., *Wikipedia* or *open source* communities). The introduced CO method is applied to eight well-known benchmark problems for lower as well as higher dimensions. CO turns out to be the best choice in 9 cases and the Fully Informed Particle Swarm Optimization (FIPS) as well as Differential Evolution (DE) approaches in 4 cases. Concerning the high dimensional problems, CO significantly outperformed FIPS as well as DE in 6 of 8 cases and seems to be a suitable approach for high dimensional problems.

Keywords: Swarm Intelligence, Collective Intelligence, Web Community, Collaborative Community, Human Community, Human Society, Community Optimization, Behavioral Model, Knowledge Base, Contribution Rule, Contribution Equation, Learning Rule, Learning Equation

1 Introduction

In nature, social insects, fishes, birds and land animals form communities called swarms to increase their probability of survival. These communities exhibit intelligent behavior created by the direct or indirect interplay between individuals and their environment. For instance, a bird in a flock is directly manipulated by the behavior of its neighbors, because the bird has to adapt its own direction to avoid collisions. Ants communicate indirectly via the environment by depositing pheromones, which control the behavior of other ants encountering them. Based on these different forms of communication, intelligent strategies emerge on a macroscopic level – the level of the whole swarm.

Humans also build communities, which show swarm properties. Humans solve problems not only by taking into account their own knowledge and experiences, but also by using communication to get the best experiences or beliefs of their colleagues [7]. According to [7], intelligent behavior and knowledge processing ability emerge from the interpersonal interaction in a society of humans. Kennedy and Eberhart's Particle Swarm Optimization (PSO) algorithm can be interpreted as a model of this knowledge processing ability of human societies [6,7,9]. The PSO algorithm also shows swarm properties similar to a flock of birds. This is based on the social term used in PSO, which ensures that each particle moves to its best neighbor to some degree. Keeping near to the neighbors (*Flock Centering*) is one of the rules of Reynold's flock algorithm [1] [15], which models the movement dynamics of a flock of birds. Thus, PSO's dynamic resembles that of a bird flock and this swarm analogy is widely used in literature.

Nowadays, communication technologies and networks are able to connect huge groups of humans. The result is that new types of web-technology driven communities of enormous size emerged. They can be classified into social networks and collaborative communities, whose members collaboratively work on something. One example is the *open source* community, which collaboratively develops software systems or libraries. The contributing members of such an *open source* project have different knowledge and levels of expertise and over time their contributions lead to complex results such as the *Linux* operating system [11]. Another very prominent example of a system that uses a collaborative web community is the web encyclopedia *Wikipedia* [23], where a huge number of participants develop the content of articles. Over time the articles are improved and completed. In other words, the web community optimizes the articles successively to reach a higher quality. This is achieved based on the following rules (R) and human behaviors (B):

R1: Each human is allowed to contribute to each article and topic.
R2: Conflicts based on different opinions are resolved by discussion on a special board.

[1] The flock algorithm was initially created to visualize a bird flock within virtual reality environments, because modeling the movement of each single bird by hand was too difficult [15].

B1: Typically, a human only contributes to a topic to which he feels competent.
B2: Sometimes, a human overrates his competence and contributes nonsense. Such nonsense is repaired by the competent ones after a short while.
B3: While contributing, a human also learns from the contributions done by others.
B4: Humans who are not contributing, also learn from all articles as well.

Note that behaviors **B1** to **B4** also describe the case of an *open source* community. There, an article is just a source code file and contributions are pieces of code. Contributing nonsense in this context means to contribute source code with bugs or being insufficient in some way. Thus, behaviors **B1** to **B4** can be considered as a behavioral model describing collaborative human communities.

In [4] the findings of an investigation carried out by *Nature* are presented, where *Wikipedia* articles and their related *Encyclopaedia Britannica* versions were compared with respect to the number of contained errors and accuracy. For this, 42 articles from *Wikipedia* were peer reviewed by a group of *Nature* experts and compared. The given results are

	Wikipedia	*Encyclopaedia Britannica*
serious errors, misinterpretations of concepts	4	4
factual errors, misleading statements, omissions	162	123

Because it is not clear whether the chosen articles were representative [2], these results have to be considered carefully. Nevertheless, it seems that the following general conclusion can be drawn based on it:

> The quality in terms of *number of errors* of a certain number of *Wikipedia* articles comes close to some degree to that of their counterparts in *Encyclopaedia Britannica*.

This raises an interesting question: if a web community is able to collectively optimize the content of articles in order to reach high quality, *is such a community also able to optimize functions*?

[2] Giles did not give any information about how the sample of 42 articles was chosen (arbitrary?, based on their experts?, uniformly distributed over all *Wikipedia* articles?). Thus, it is not clear whether the chosen articles are representative or not. Also, a clear definition of an "error" is missing! Thus, the presented experiment seems to be a bit arbitrary.

In order to give the answer in advance: Yes, it is! In [22] an appropriate optimization algorithm called Community Optimization (CO) was firstly introduced, which is based on the behavioral model defined by **B1** to **B4**. Thus, CO optimizes a function by simulating a community of authors, which edits or improves a knowledge-base, whereby the knowledge-base is a more general view on a database of articles. The authors of the community contribute to the different topics of the knowledge-base based on their expertise. Furthermore, the authors improve their own knowledge by learning from the knowledge-base. The different expertise ratings of an author grow with the success of his contributions.

The paper at hand is an extended version of [22] and is organized as follows. Section 2 reintroduces the Community Optimization algorithm. The used testbed of benchmark functions is presented in section 3. Section 4 explains the systematic of the employed parameter exploration to determine a suitable parameter set for the used testbed. The conducted experiments with their results are presented in section 5. Finally, in section 6 some conclusions are drawn.

2 Community Optimization

The Community Optimization (CO) algorithm optimizes a function by simulating a community, which collaboratively optimizes a knowledge-base. It draws its inspiration from the collective intelligence emerged by a web community like, for instance, the *Wikipedia* authors who collaboratively edit and improve an article-base. In CO, this article-base is a more general construct called "knowledge-base" and the term author is replaced by the more general term "person". The knowledge-base is thought to represent the problem instance to be solved and is realized as a real valued vector. The different vector components (or decision variables of the problem) represent different topics contained in this knowledge-base, i.e., each single topic represents a single decision variable. Thus, the dimension of the problem is the number of topics. In the following table, the analogy between the CO model and function optimization is given:

CO	**Function Optimization**
knowledge-base	global-best vector
quality of knowledge-base	fitness of global-best
community + knowledge-base	population of vectors
topic	decision variable
knowledge	value of decision variable
number of topics	dimension of vectors
person's knowledge	vector of population
expertise ratings	-

At the core of the CO algorithm, two rules are employed. On the one hand, a community of persons improves the different topics of the knowledge-base based on the expertise ratings of the persons. The better a person's expertise, the higher the probability that he feels competent enough to edit a given piece of knowledge. After editing, the person's expertise is adapted, if he improved the quality of the given piece of knowledge. On the other hand, all persons improve their own knowledge using the knowledge-base. This way, a sort of stigmergy [3] is realized, because a person can change the knowledge-base (the environment) and other persons take this new knowledge and change their own state and thus their behavior. It can be stated that CO uses a swarm of persons, who change their environment (knowledge-base) to influence themselves mutually, while changing the environment additionally solves an optimization problem (see section 2.8).

2.1 Notation

Let D be the dimension of the problem (i.e., the dimension of the search space $\mathbb{R}^D$) and $\boldsymbol{\kappa} = (\kappa_1, \cdots, \kappa_t, \cdots, \kappa_D) \in \mathbb{R}^D$ the knowledge-base comprised of D topics with κ_t denoting the contained knowledge of topic t. Then, let $\boldsymbol{\epsilon} = (\epsilon_1, \cdots, \epsilon_t, \cdots, \epsilon_D) \in \mathbb{R}^D$ denote the levels of expertise with which the topics were improved by the last person. Topic's knowledge κ_t and expertise ϵ_t correspond, i.e., ϵ_1 belongs to κ_1, ϵ_2 to κ_2 and so on.

Furthermore, let N_{pers} denote the number of persons and $\mathcal{C}$ the set of persons $\mathcal{C} = \{P_1, \cdots, P_p, \cdots, P_{N_{pers}}\}$ forming the community. Each person $P_p = (\boldsymbol{k_p}, \boldsymbol{e_p})$ has knowledge, which is a position in the search space ($\boldsymbol{k_p} \in \mathbb{R}^D$) and expertise ratings ($\boldsymbol{e_p} \in \mathbb{R}^D$). The person's knowledge (person's expertise) w.r.t. a given topic t is denoted by k_{p_t} (e_{p_t}), whereby expertise e_{p_t} corresponds to knowledge k_{p_t} (again, e_{p_1} belongs to k_{p_1}, e_{p_2} to k_{p_2}, etc.).

Do not confuse expertise e_{p_t} with ϵ_t. Expertise e_{p_t} denotes the expertise of a person w.r.t. a given topic t, whereas ϵ_t means the expertise by which the knowledge κ_t of topic t was improved. So to speak, ϵ_t reflects the quality or level of κ_t.

For each topic t a lower and upper bound is defined by L_t (lower bound) and U_t (upper bound). Thus, for all topics $t \in \{1, \cdots, D\}$ we have that $\kappa_t, k_{p_t} \in [L_t, U_t]$ at least for the initial community.

In order to compute the objective value (or quality) of a knowledge-base, a quality function

$$Q : \mathbb{R}^D \to \mathbb{R}$$

is used.

2.2 Algorithm

The main algorithm is quite the same as for most optimizers and reads:

```
1: Initialization
2:
```

[3] The term *stigmergy* denotes the indirect communication via the environment as used, e.g., by ants, which deposit pheromones to guide their colony mates [1].

```
3: while termination criteria not fulfilled do
4:
5:     Iteration
6:
7: end while
```

There are two sets of termination criteria used by CO:

1. Run until a prespecified number of iterations or a given quality is reached. This means to run until the wished quality is reached, but no longer than the given number of iterations.
2. Run until a pre-specified number of allowed function evaluations is reached. This is used primarily to compare CO with other algorithms, which is done by setting for all algorithms the same number of evaluation steps. The typical *size of population* $\times$ *number of iterations* computation does not work with CO as will be seen in section 5.

In the following two subsections, the procedures for `Initialization` and `Iteration` are described.

2.3 Initialization

Firstly, an iteration counter G is set to 0. Then, the knowledge-base $\boldsymbol{\kappa}$ is randomly set by uniformly drawing the knowledge of topics κ_t from $\mathcal{U}(L_t, U_t)$. The corresponding expertise vector $\boldsymbol{\epsilon}$ is set to a zero-vector. This way, all persons of the community are allowed to contribute to all topics in the first iteration. After initializing the knowledge-base $\boldsymbol{\kappa}$, its quality is computed and stored in variable $Q_{\boldsymbol{\kappa}} \in \mathbb{R}$ to avoid multiple calls to $Q(\cdot)$ for the same $\boldsymbol{\kappa}$ in the `Iteration` procedure.

The community $\mathcal{C}$ is initialized with randomly created persons P_p, whereby each k_{p_t} is drawn from $\mathcal{U}(L_t, U_t)$. The initial expertise vector $\boldsymbol{e_p}$ is set to a one-vector so a person has the same degree of expertise w.r.t. each topic ($\boldsymbol{e_p} = (1, 1, \ldots, 1)$). Algorithm (1) gives the pseudo-code of initialization.

2.4 Iteration

The iteration performs the two rules of CO:

Contribution Rule: Community contributes to knowledge-base,
Learning Rule: Community learns from knowledge-base.

They are executed in sequence w.r.t. the whole community. That means that first all persons of the community follow the contribution rule. Afterwards, all persons perform the learning rule. Both steps are described in the following subsections. The iteration is also given as pseudo-code in Algorithm (2) to present the interplay of all upcoming equations in more detail.

Algorithm 1. Initialization of CO

```
1: $G \leftarrow 0$
2:
3: for all $t \in \{1, \cdots, D\}$ do
4:     $\kappa_t \sim \mathcal{U}(L_t, U_t)$
5:     $\epsilon_t \leftarrow 0$
6: end for
7:
8: $Q_{\boldsymbol{\kappa}} \leftarrow Q(\boldsymbol{\kappa})$
9:
10: for all $p \in \{1, \cdots, N_{pers}\}$ do
11:     for all $t \in \{1, \cdots, D\}$ do
12:         $k_{p_t} \sim \mathcal{U}(L_t, U_t)$
13:         $e_{p_t} \leftarrow 1$
14:     end for
15: end for
```

Contribution Rule. In the first step, all persons of the community have to contribute their knowledge w.r.t. all topics to the knowledge-base. Typically, only persons having a sufficient expertise contribute to a given topic. This is realized by setting the person's expertise in topic t into relationship to the quality of this topic's knowledge already in existence in the base (realizes behavior **B1**, page 2). For this, a roulette-wheel sampling is performed between the person's expertise e_{p_t} and the expertise level of the t^{th} topic ϵ_t (Algorithm (2), lines 5-6). If the person's expertise is higher or similar to topic's expertise level ϵ_t, the probability is also high that the person feels competent enough to contribute his knowledge to this topic. But if the expertise of the person is low, he rarely contributes. Sometimes a person with a lower expertise will contribute successfully, which is a source of diversification. But if this person contributes nonsense, his contribution is discarded (realizes behavior **B2**, page 2).

In case the expertise e_{p_t} of person P_p was selected by the roulette-wheel sampling, person P_p contributes to the topic's knowledge κ_t, whereby the new knowledge κ'_t is computed by the contribution equation as

$$\kappa'_t \sim \mathcal{N}(\ k_{p_t}\ ,\ \lambda_C \cdot |k_{p_t} - \kappa_t|\ ) \tag{1}$$

with $\mathcal{N}(\mu, \sigma)$ denoting the normal distribution with mean μ and standard deviation σ. Thus, the new knowledge κ'_t is sampled around the person's knowledge k_{p_t}, whereby the distance between the knowledge-base and the person influences the range of sampling (σ). This range can be further controlled by the positive factor λ_C, which denotes the degree of exploration while contributing to a topic. The lower λ_C is, the narrower is the normal distribution and the nearer the new knowledge κ'_t is placed to the person's knowledge k_{p_t}. If λ_C is set to 0, then $\kappa'_t = k_{p_t}$. Thus, a lower λ_C supports exploitation. The higher λ_C is, the wider is the normal distribution, which supports exploration.

After computing the new knowledge κ'_t for topic t, a greedy approach is used: κ'_t is only kept, if it is better than the previous knowledge κ_t w.r.t. a given tolerance of acceptance τ, otherwise κ'_t is discarded. In order to realize this, the knowledge-base $\boldsymbol{\kappa}$ is copied to $\boldsymbol{\kappa}'$ and Eq. (1) is applied to $\boldsymbol{\kappa}'$. Then, it is checked whether

$$Q(\boldsymbol{\kappa}') - Q(\boldsymbol{\kappa}) < \tau \tag{2}$$

(in case of minimization). If so, the new knowledge-base $\boldsymbol{\kappa}'$ replaces $\boldsymbol{\kappa}$ and the expertise ratings are updated.

The person's expertise e_{p_t} is increased, if he could successfully contribute to the given topic t. The new expertise e'_{p_t} is computed as

$$e'_{p_t} = e_{p_t} + 1 \tag{3}$$

counting this way successful contributions. After the person's expertise is updated, the topic's expertise level ϵ_t needs to be updated as well. The new expertise level ϵ'_t is set to the new expertise of the contributing person:

$$\epsilon'_t = e'_{p_t} \tag{4}$$

Note that if the successful contribution was realized by a person with an expertise being lower than ϵ_t, the new topic's expertise level can decrease. In this case more persons with lower expertise ratings get a chance to contribute to topic t.

Learning Rule. In the second step, all persons of the community have to improve their knowledge w.r.t. all topics contained in the knowledge-base (realizes behaviors **B3** and **B4**, page 2). This step enables new knowledge to spread within the community and realizes a sort of stigmergy. The person's new knowledge k'_{p_t} w.r.t. topic t is updated by the learning equation as

$$k'_{p_t} \sim \mathcal{N}(\ \kappa_t\ ,\ \lambda_L \cdot |\kappa_t - k_{p_t}|\ ) \tag{5}$$

with $\mathcal{N}(\mu, \sigma)$ again denoting the normal distribution with mean μ and standard deviation σ. Thus, the person's new knowledge k'_{p_t} is sampled around the knowledge κ_t, whereby the distance between the person and the knowledge-base influences the range of sampling (σ). This range can be further controlled by the positive factor λ_L, which denotes the degree of exploration while learning from the knowledge-base. The lower λ_L is, the narrower is the normal distribution and the nearer the person's new knowledge k'_{p_t} is placed to the knowledge κ_t. If λ_L is set to 0, then $k'_{p_t} = \kappa_t$. Thus, a lower λ_L supports exploitation. The higher λ_L is, the wider is the normal distribution, which supports exploration.

2.5 Tolerance of Acceptance

While contributing to a topic, the quality of a newly computed knowledge-base $\boldsymbol{\kappa}'$ is checked against the quality of the current knowledge-base $\boldsymbol{\kappa}$. This involves a

Algorithm 2. Iteration of CO

```
// Contribution Rule: Each person tries to contribute to each topic
for all $p \in \{1, \cdots, N_{pers}\}$ do
    for all $t \in \{1, \cdots, D\}$ do

        $r \sim \mathcal{U}(0, e_{p_t} + \epsilon_t)$            // expertise high enough?
        if $r < e_{p_t}$ then

            $\boldsymbol{\kappa}' \leftarrow \boldsymbol{\kappa}$

            $\kappa'_t \sim \mathcal{N}(\ k_{p_t}\ ,\ \lambda_C \cdot |k_{p_t} - \kappa'_t|\ )$

            $Q_{\boldsymbol{\kappa}'} \leftarrow Q(\boldsymbol{\kappa}')$

            if $Q_{\boldsymbol{\kappa}'} - Q_{\boldsymbol{\kappa}} < \tau$ then

                $e_{p_t} \leftarrow e_{p_t} + 1$

                $\epsilon_t \leftarrow e_{p_t}$

                $\boldsymbol{\kappa} \leftarrow \boldsymbol{\kappa}'$
                $Q_{\boldsymbol{\kappa}} \leftarrow Q_{\boldsymbol{\kappa}'}$

            end if
        end if
    end for
end for

// Learning Rule: Each person learns from knowledge-base
for all $p \in \{1, \cdots, N_{pers}\}$ do
    for all $t \in \{1, \cdots, D\}$ do

        $k_{p_t} \sim \mathcal{N}(\ \kappa_t\ ,\ \lambda_L \cdot |\kappa_t - k_{p_t}|\ )$

    end for
end for

$G \leftarrow G + 1$
```

greedy approach, which uses the tolerance of acceptance τ to control the strictness of accepting new knowledge. A tolerance $\tau = 0$ leads to a strict-greedy approach, which only accepts contributions with a better quality:

$$\tau = 0 \quad \Leftrightarrow \quad Q(\boldsymbol{\kappa}') < Q(\boldsymbol{\kappa}) \tag{6}$$

This is the default configuration of CO and is used in the experiments in section 5. It works well in continuous spaces, which have a lot of gradients so new positions $\boldsymbol{\kappa}'$ near a contributing person will have different qualities compared to the person's position or the current knowledge-base $\boldsymbol{\kappa}$.

In more discrete spaces, which have plateaux, the strict-greedy approach could perform sub-optimally. In Fig. 1 this is shown for a single topic t. There, the whole community is situated on a plateau. Thus, the qualities around all person's positions as well as around the current knowledge-base $\boldsymbol{\kappa}$ are exactly the same. If a person contributes (like P_2 in Fig. 1), then the new knowledge-base $\boldsymbol{\kappa}'$ is placed near to this person. But because the person's position and his neighborhood are on the same plateau as the knowledge-base $\boldsymbol{\kappa}$, the new knowledge-base $\boldsymbol{\kappa}'$ will have the same quality as well. The strict-greedy approach discards same qualities, since it expects a clearly better quality (see Eq. (6)). In such situations, no person is able to successfully contribute and the community stagnates.

In order to overcome such situations, the tolerance of acceptance τ was introduced. A tolerance $\tau > 0$ leads to a weak-greedy approach, which allows the acceptance of contributions of same qualities as well as of worse ones. This weak-greedy approach solves the problem of stagnation, because the community as well as the knowledge-base $\boldsymbol{\kappa}$ are now allowed to move on the plateau. This increases the probability that they come close enough to a border so the community can leave the plateau.

Although on a plateau the actual value of τ does not matter, because every one greater than 0 accepts contributions of same qualities, it could matter if the community has left the plateau. Then, the community could be confronted with gradients and in this situation a $\tau > 0$ would allow to accept worse contributions as well. What is needed for search spaces containing discrete regions is a τ value that is greater than 0 to allow movements on plateaux, but is additionally small enough to avoid accepting really worse qualities in gradient-like regions of the search space. A reasonable trade-off could be to use a tolerance $\tau = 1\text{e-}300$ as one of the smallest values a *double* data-type of current programming languages can represent. Such τ values lead to a lesser-strict-greedy approach, which allows the movement on plateaux while having only a minor effect in gradient-like regions. Thus, it practically approximates the $\leq$ relation:

$$\tau = 1\text{e-}300 \quad \approx> \quad Q(\boldsymbol{\kappa}') \leq Q(\boldsymbol{\kappa}) \tag{7}$$

2.6 Geometric Activity

The question arises, which geometric activity is realized by both rules of CO. In order to depict this activity, let's consider a two-dimensional problem ($D = 2$) so there are two topics in the knowledge-base $\boldsymbol{\kappa} = (\kappa_1, \kappa_2)$. Furthermore, let's have a community $\mathcal{C} = \{P_1, P_2, P_3\}$ of three persons. In Fig. (2), part (A), a possible situation in the used search space is given. Let's assume that person P_1's expertise is not high enough for it to contribute to either topic, person P_2 has a high expertise only for topic 1 (which is the x-axis) and person P_3 only for topic 2 (which is the y-axis). Then, performing the contribution rule leads to the following behavior:

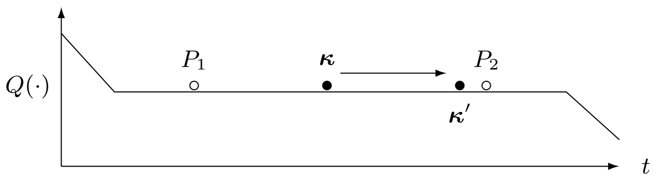


Fig. 1. A community placed on a plateau in a search space. All entities have the same quality: $Q(\boldsymbol{\kappa}') = Q(\boldsymbol{\kappa}) = Q(P_1) = Q(P_2)$. Thus, each $\boldsymbol{\kappa}'$ on the plateau will be rejected by the strict-greedy approach leading to stagnation.

1. Person P_1 does not contribute to one of both topics so the knowledge-base keeps unchanged.
2. Person P_2 contributes to topic 1. According to Eq. (1) this means that κ_1 is moved in direction to P_2's position k_{21}. In other words, person P_2 pulls κ_1 in his direction. This is depicted in Fig. (2), part (B), by the arrow captioned by "1.".
3. Person P_3 contributes to topic 2. According to Eq. (1) this means that κ_2 is moved in direction to P_3's position k_{32}. Again, person P_3 pulls κ_2 in his direction. This is depicted in Fig. (2), part (B), by the arrow captioned by "2.".

What needs to be noted here is that the persons pull the knowledge-base only for those dimensions for which they have an expertise being high enough. Also, the persons contribute in their given order and not in parallel (first P_2 changed the base, then P_3). In cases where several persons contribute to the same topic, the later ones can eliminate the contribution of previous ones. This is not a weak point of CO, but designed by full intention. The inspiration is a *Wikipedia*-like system. There, the same effect takes place, because authors overwrite the work done by others before. Even oscillations are possible in *Wikipedia*-like systems, if two authors permanently change back the same piece of knowledge to their personal opinion, if this piece was changed by the other one. This is called an "edit war" [24]. Since in CO the persons do not have egos, because they are simple entities, no mechanism against "edit wars" is included yet, but one is imaginable [4].

Performing the learning rule, depicted in Fig. (2), part (C), lets all persons move to the position of the new knowledge-base. Note that this is also true for person P_1, although he did not contribute to the base. Thus, his movement (his behavior) is influenced by the other persons who changed the base. Finally, the following statements can be given:

[4] This could be realized by rejecting further contributions of a person for a given topic for a certain time (number of iterations), after his last contribution. But this would need a centralized control, because a person would not respect such a rule in reality. In section 2.8 it is argued that the person community is a swarm. This property would get lost, if a centralized control would be established.

S1: All persons try to pull the knowledge-base to their own position for as many dimensions as possible.
S2: A person can specialize to subsets of topics, based on his expertise vector.
S3: All persons contribute in a sequence.
S4: All persons move to the new knowledge-base in all dimensions.

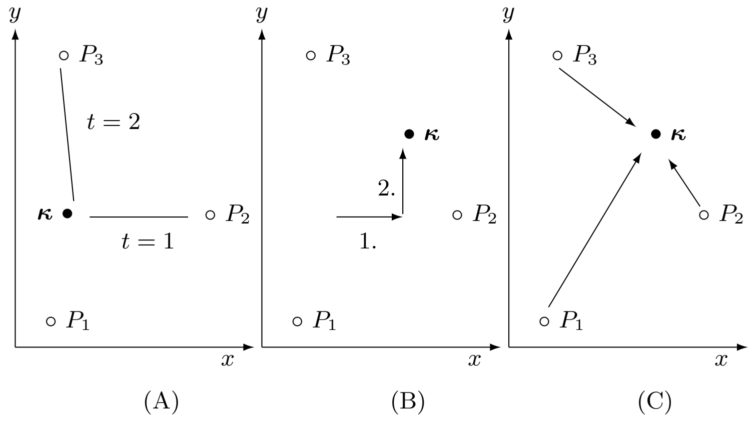


Fig. 2. The geometric meaning of the contribution rule (A,B) and learning rule (C)

2.7 Boundaries

All knowledge of persons k_{p_t} as well as all topics of the knowledge-base κ_t are initialized within the $[L_t, U_t]$ bounds. But they can leave these boundaries while optimizing. If the actual optimum of the problem at hand lies outside of the chosen boundaries, the region near a boundary could become more attractive than the actual region within the boundaries. Thus, the community of persons would try to move to this boundary. Both, the community and the knowledge-base could move beyond this boundary, because of the following reasons:

Knowledge-Base. Assumed the community is located near a boundary. If a contributing person is placed close enough to this boundary so that the normal distribution set over the person's knowledge would reach out beyond the boundary, then the knowledge-base could be placed outside the boundary, because the normal distribution is symmetric.

Knowledge of Persons. Because the knowledge-base can move beyond the boundaries by the contribution rule, a person's knowledge can leave the bounded region as well by the learning rule. Learning from the knowledge-base means to place the person's knowledge somewhere within the normal distribution set over the knowledge-base. If the knowledge-base is located outside the boundaries, then the new person's knowledge is also placed outside the boundaries near to the knowledge-base. Even in case the knowledge-base is still within the bounded region, but close enough to a boundary so that its normal distribution reaches out beyond this boundary, a certain number of persons could be placed outside this boundary.

This property of CO that the community can move over the whole search space is an interesting feature, if CO shall be applied to a dynamic optimization problem, where the actual optimum moves in the search space, too. But if, for some reasons, it is necessary to keep the community within its boundaries, one could use the clamping approach for velocities as used by PSO [9]. Note that it is important not only to clamp all person's knowledge (k_{p_t}), but also the knowledge-base (κ_t). This could be done at line 35 in Algorithm (2). But in the conducted experiments (see section 5) there was no need for this.

2.8 Is It Swarm Intelligence?

The question arises, whether the CO algorithm is (or uses) swarm intelligence or not. A swarm can be defined as follows:

Definition 1 (Swarm). *A swarm is a decentralized group of simply behaving entities, which mutually influence themselves by some sort of communication.*

There are two classes of communication distinguished in literature: the direct communication between swarm mates and the indirect communication, e.g., by changing the environment [1]. Thus, a swarm mate can change the state or behavior of another swarm mate through direct interaction or by changing their environment. The environment of the community in CO is the search space and the contained knowledge-base as a movable point. The knowledge-base is an inactive object and does not belong to the actual community, since it is merely used by the persons and does not perform own activities, as can be seen in Algorithm (2).

Furthermore, the community is a group of persons with rather simple behavior: they can pull the knowledge-base object to their direction (statement S1, page 12) and they can move to its location (statement S4, page 12). If a person changes the position of this knowledge-base object, he changes the environment. Since all persons move to the knowledge-base's position by the learning rule, the position of the knowledge-base influences the movements of the persons. Thus, a person can influence the state and behavior of all others by pulling the knowledge-base to his position. This is an indirect communication by changing the environment and termed as stigmergy in literature [1]. Finally, the persons are not under the influence of a centralized control. The whole concept is person-related, i.e., the persons decide by themselves whether to contribute or not.

From all this follows that the community of persons is a stigmergy-based swarm and thus CO can be classified as a swarm intelligence algorithm.

3 Benchmark Functions

In order to evaluate the capabilities of CO, eight benchmark functions from a test suite for large scale continuous optimization problems [5] were chosen. Since many population-based approaches like PSO tend to perform best near the origin (zero-vector) of a search space [13], only the shifted versions of these benchmark functions were used, because it is not clear whether CO has this affinity to the origin, too. The benchmarks out of [5], which are not shifted (Schwefel 1.2, Schaffer) were transformed to shifted versions for this work. In Table 1 (page 14), the properties of all eight benchmarks are presented as given in [5], whereby the last row summarizes the numbers of different properties.

Table 1. The properties of all used benchmark functions

f_i Name	**U**nimodal **M**ultimodal	**S**eparable **N**on-separable	**D**imension-wise **N**ot **D**imension-wise
f_1 Shifted Sphere	U	S	D
f_2 Shifted Rosenbrock	M	N	D
f_3 Shifted Rastrigin	M	S	D
f_4 Shifted Griewank	M	N	ND
f_5 Shifted Ackley	M	S	D
f_6 Shifted Schwefel 2.21	U	N	ND
f_7 Shifted Schaffer	U	N	D
f_8 Shifted Schwefel 1.2	U	N	ND
	$4 \times$ U $4 \times$ M	$3 \times$ S $5 \times$ N	$5 \times$ D $3 \times$ ND

The eight benchmark functions were chosen in a way to get a good balance between unimodal / multimodal, separable / non-separable as well as dimension-wise / not dimension-wise optimizable ones. They are defined in the following:

f_1: Shifted Sphere

$$\begin{aligned}
f_1(< x_1, \cdots, x_D >) &= \sum_{t=1}^{D} z_t^2 + f_{bias} \\
& z_t = x_t - c \\
f_{bias} &= -450 \\
[L_t, U_t] &:= [-100, 100] \quad (1 \leq t \leq D) \\
\text{Global minimum } &: \ f_1(< c, \cdots, c >)
\end{aligned}$$

f_2: Shifted Rosenbrock

$$\begin{aligned}
f_2(< x_1, \cdots, x_D >) &= \sum_{t=1}^{D-1}[100 \cdot (z_{t+1} - z_t^2)^2 + (z_t - 1)^2] + f_{bias} \\
& z_t = x_t - c \\
f_{bias} &= 390 \\
[L_t, U_t] &:= [-100, 100] \quad (1 \le t \le D) \\
\text{Global minimum} &: f_2(< c+1, \cdots, c+1 >)
\end{aligned}$$

f_3: Shifted Rastrigin

$$\begin{aligned}
f_3(< x_1, \cdots, x_D >) &= 10 \cdot D + \sum_{t=1}^{D}[z_t^2 - 10 \cdot \cos(2 \cdot \pi \cdot z_t)] + f_{bias} \\
& z_t = x_t - c \\
f_{bias} &= -330 \\
[L_t, U_t] &:= [-5, 5] \quad (1 \le t \le D) \\
\text{Global minimum} &: f_3(< c, \cdots, c >)
\end{aligned}$$

f_4: Shifted Griewank

$$\begin{aligned}
f_4(< x_1, \cdots, x_D >) &= 1 + \sum_{t=1}^{D} \frac{z_t^2}{4000} - \prod_{t=1}^{D} \cos(\frac{z_t}{\sqrt{t}}) + f_{bias} \\
& z_t = x_t - c \\
f_{bias} &= -180 \\
[L_t, U_t] &:= [-600, 600] \quad (1 \le t \le D) \\
\text{Global minimum} &: f_4(< c, \cdots, c >)
\end{aligned}$$

f_5: Shifted Ackley

$$\begin{aligned}
f_5(< x_1, \cdots, x_D >) &= -20 \exp(-0.2\sqrt{\frac{1}{D}\sum_{t=1}^{D} z_t^2}) - \exp(\frac{1}{D}\sum_{t=1}^{D}\cos(2\pi z_t)) \\
& +20 + e + f_{bias} \\
& \\
& z_t = x_t - c \\
f_{bias} &= -140 \\
[L_t, U_t] &:= [-32, 32] \quad (1 \le t \le D) \\
\text{Global minimum} &: f_5(< c, \cdots, c >)
\end{aligned}$$

f_6: Shifted Schwefel 2.21

$$\begin{aligned}
f_6(< x_1, \cdots, x_D >) &= \max_{t \in \{1,\cdots,D\}} \{|z_t|\} + f_{bias} \\
& z_t = x_t - c \\
f_{bias} &= -450 \\
[L_t, U_t] &:= [-100, 100] \quad (1 \leq t \leq D) \\
\text{Global minimum } &: f_6(< c, \cdots, c >)
\end{aligned}$$

f_7: Shifted Schaffer

$$\begin{aligned}
f_7(< x_1, \cdots, x_D >) &= \sum_{t=1}^{D-1} [(z_t^2 + z_{t+1}^2)^{0.25} \cdot (\sin^2(50 \cdot (z_t^2 + z_{t+1}^2)^{0.1}) + 1)] + f_{bias} \\
& z_t = x_t - c \\
f_{bias} &= 100 \\
[L_t, U_t] &:= [-100, 100] \quad (1 \leq t \leq D) \\
\text{Global minimum } &: f_7(< c, \cdots, c >)
\end{aligned}$$

f_8: Shifted Schwefel 1.2

$$\begin{aligned}
f_8(< x_1, \cdots, x_D >) &= \sum_{t=1}^{D} (\sum_{u=1}^{t} z_u)^2 + f_{bias} \\
& z_u = x_u - c \\
f_{bias} &= 100 \\
[L_t, U_t] &:= [-100, 100] \quad (1 \leq t \leq D) \\
\text{Global minimum } &: f_8(< c, \cdots, c >)
\end{aligned}$$

4 Parameter Exploration

Developing new algorithms often includes the introduction of parameters and CO is no exception to this. Thus, the question arises, which parameter set would be the best for a given problem at hand. The best ways of determining suitable parameter sets for a given problem are either using an optimizer that optimizes the new algorithm's parameters or to perform a systematic parameter exploration over suitable intervals. The strategy used in this work was the second way: a systematic parameter exploration. But we are not interested in specialised parameters that work only for a single benchmark. We want a more general parameter set, which can be applied to all benchmark functions used in the experiments. Thus, the parameter set was determined by considering all eight benchmark functions together by performing the following steps:

1. For each benchmark function $f_1, \cdots, f_8$ a parameter exploration was conducted by using the following intervals:

Parameter	Values	Numbers
N_{pers}	5 , 10 , $\cdots$, 30	6
λ_C	0.1 , 0.2 , $\cdots$, 2.0	20
λ_L	0.1 , 0.2 , $\cdots$, 2.0	20
Experiments:		2400

   That means, for each of the $n = 2400$ parameter combinations an experiment with 30 dimensions averaged over 10 independent runs was conducted for each benchmark function. This led to eight archives $A_1, \cdots, A_8$ containing the parameter combinations ϕ_i with their corresponding mean quality values μ_{Q_i}:

   $$A_j := < (\mu_{Q_1}, \phi_1), \cdots, (\mu_{Q_n}, \phi_n) >$$

   A parameter combination ϕ_i is an element of the set of all parameter combinations Φ:

   $$\phi_i \in \Phi = \{(5, 0.1, 0.1), (5, 0.1, 0.2), \cdots, (30, 2.0, 2.0)\}$$

   Each archive contains entries for all n parameter combinations.

2. After all eight archives were computed, they were sorted according to their mean qualities. In a sorted archive, the (μ_{Q_i}, ϕ_i) tupel with the lowest mean quality μ_{Q_i} is now placed at the top of the archive, that is, it has the lowest index or rank 0. The one with the second lowest mean quality gets rank 1 and so on.

3. We are interested in the parameter combination ϕ^* with the best performance over all eight benchmark functions. Thus, what needs to be searched for is a parameter combination that is as close to the top of a sorted archive as possible and this for all archives in parallel. The strategy used in this work was to compute the mean rank for each parameter combination over all sorted archives. The best parameter combination ϕ^* is the one with the lowest mean rank (minimum of mean ranks), since in average it is nearest to the top of all sorted archives.
   More formally, let $r_j(\phi_i)$ denote the sorted rank of parameter combination ϕ_i within archive A_j. Then,

   $$\phi^* = \arg\min_{\phi_i \in \Phi} \{ \ \frac{1}{8} \sum_{j=1}^{8} r_j(\phi_i) \ \} \qquad (8)$$

After performing all above steps, the best parameter combination ϕ^* was obtained and the result is:

Mean Rank	N_{pers}	λ_C	λ_L
170.0	25	0.3	1.6

The mean rank over all sorted archives is quite high with about 170. But note that this is rank 170 in archives containing 2400 entries. Nevertheless, the obtained mean rank shows that this parameter set is quite general and by no means too specialised to one of the eight benchmark functions. This was the aim of this parameter exploration. Of course, by using a more general parameter set, CO can not realize the best possible performance for one of the eight benchmark functions used.

4.1 Parameter Sensitivity

After obtaining parameter set ϕ^*, one may wonder how sensitive are the parameters to changes, if for a new problem a more specialised parameter set is searched. Thus, based on the parameter set ϕ^*, the different parameters were analysed within their intervals of the parameter exploration, whereby one parameter was varied and the other both were kept. For example, the parameter λ_C was varied over its interval $\{0.1, \cdots, 2.0\}$, while the other two parameters were kept to their values from ϕ^*. In Figs. 3 (page 20), 4 (page 21) and 5 (page 21) the results are illustrated. The used vertical dashed lines indicate the value of the parameter set ϕ^*. Note that the quality values were normalized to be able to present the graphs of all benchmark functions together in one diagram. Unfortunately, for the λ_C parameter the diagram gets too crowded if it contains all graphs. Thus, for the sake of readability, the graphs for this parameter were spread over three diagrams. In the following, the observations out of the diagrams are given:

λ_C: The three diagrams of Fig. 3 (page 20) show that four benchmark functions (f_3, f_4, f_6, f_7) exhibit quite chaotic graphs, where neighboring points show high differences in their quality values. For them, the λ_C parameter can be considered as sensitive. The graphs of four benchmark functions (f_1, f_2, f_5, f_8) are quite smooth up to approximately $\lambda_C \approx 1$. But then, they get as chaotic as the other four functions mentioned above (except f_1). Thus, for them, the λ_C parameter can be considered as partly sensitive. Furthermore, the lowest quality values of half of the benchmark functions (f_1, f_2, f_5, f_8) are concentrated on the left side of the diagrams with the lower λ_C values, which support exploitation. Unfortunately, these four benchmarks are well balanced unimodal / multimodal as well as separable / non-separable functions. Thus, nothing can be drawn from these properties. But three functions (f_1, f_2, f_5) can be optimized dimension-wise and just one (f_8) not. Although both types of this property are covered, there seems to be a tendency that dimension-wise optimizable functions benefit from a lower λ_C value.

λ_L**:** As revealed by Fig. 4 (page 21), the graphs of most benchmark functions are quite smooth. Little changes of this parameter do not produce chaotic jumps within the quality values. Thus, the λ_L parameter is not very sensitive. Furthermore, the lowest quality values of all benchmark functions are concentrated on the right side of the diagram with the higher λ_L values, which support exploration.

N_{pers}**:** As can be seen in Fig. 5 (page 21), the graphs of most benchmark functions descent with an increase of the number of persons. Since increasing the population size also increases the performance of other optimizers like PSO or DE, CO behaves similarly to them. Two functions (f_2, f_8) show more sensitive graphs. Their property in common is that both are non-separable. But other non-separable functions (f_4, f_6, f_7) show non-sensitive graphs so nothing can be derived from it. The other six of the eight graphs of the benchmark functions descent quite smoothly. Changing this parameter a bit does not produce chaotic jumps within the quality values. Thus, the N_{pers} parameter is not very sensitive.

All in all, it can be stated that the λ_C parameter is sensitive, the λ_L parameter is non-sensitive and N_{pers} is mostly non-sensitive to changes.

Because the lowest quality values of half of the benchmark functions are obtained with lower λ_C values and the lower quality values of the other half of the benchmark functions are spread over the whole interval, it is recommended to set $\lambda_C \in [0, 1]$.

Since the performance of all used benchmark functions is best with higher λ_L values, it seems that CO is best used with the learning rule performing an exploration. Hence, the recommendation is to set $\lambda_L \geq 1$.

It seems that in most cases the N_{pers} parameter follows the intuitive rule: the higher, the better. But for some functions it can be necessary to perform a parameter exploration.

5 Experiments

For each of the benchmark functions as introduced in section 3, the CO algorithm, the Fully Informed Particle Swarm (FIPS) method [12], the Differential Evolution (DE) approach [17,18,19] as well as its self-adaptive variant SaDE [14] were conducted.

FIPS is based on Clerc's constricted PSO [2] and incorporates the information of all members of a neighborhood so a particle is "fully informed". Because this variant exhibits a better performance than the standard PSO method, the standard PSO was omitted.

DE performs best for separable problems with a lower crossover rate (CR), whereas a higher CR is better for non-separable problems [20]. There is no optimal standard value for CR, which works well in all situations. Thus, the

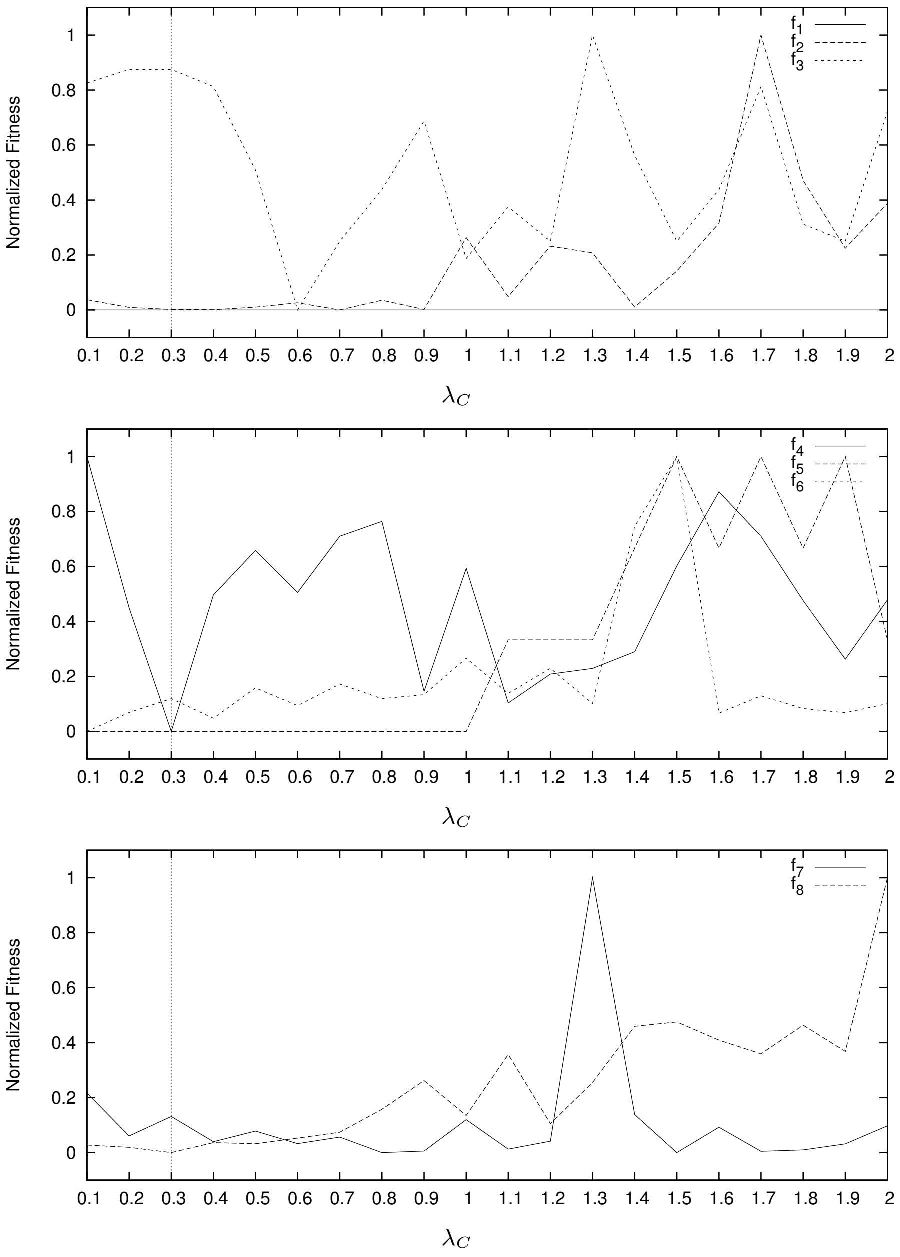


Fig. 3. The quality values for the λ_C parameter over all benchmark functions

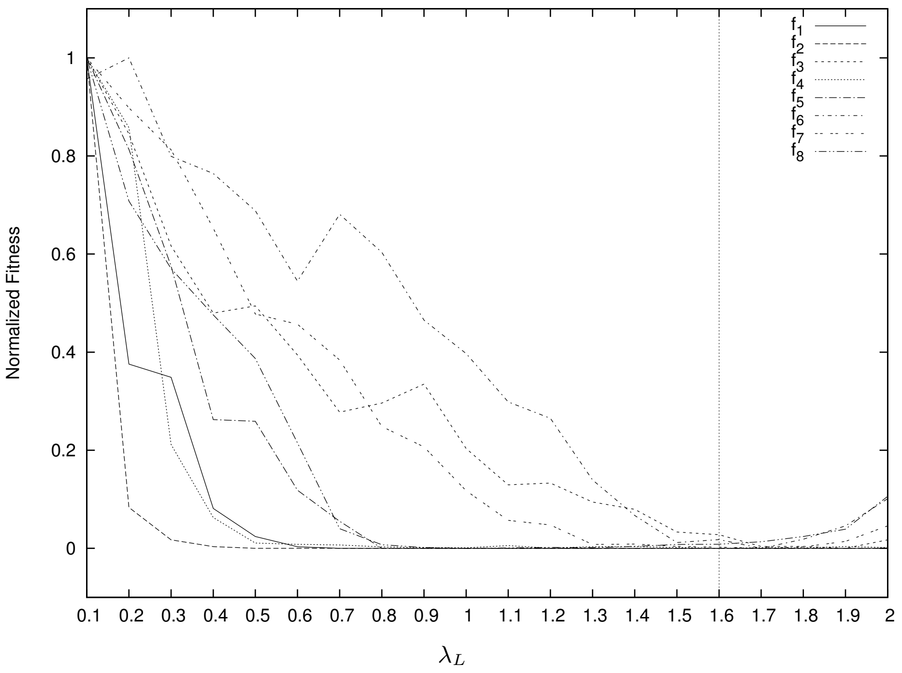


Fig. 4. The quality values for the λ_L parameter over all benchmark functions

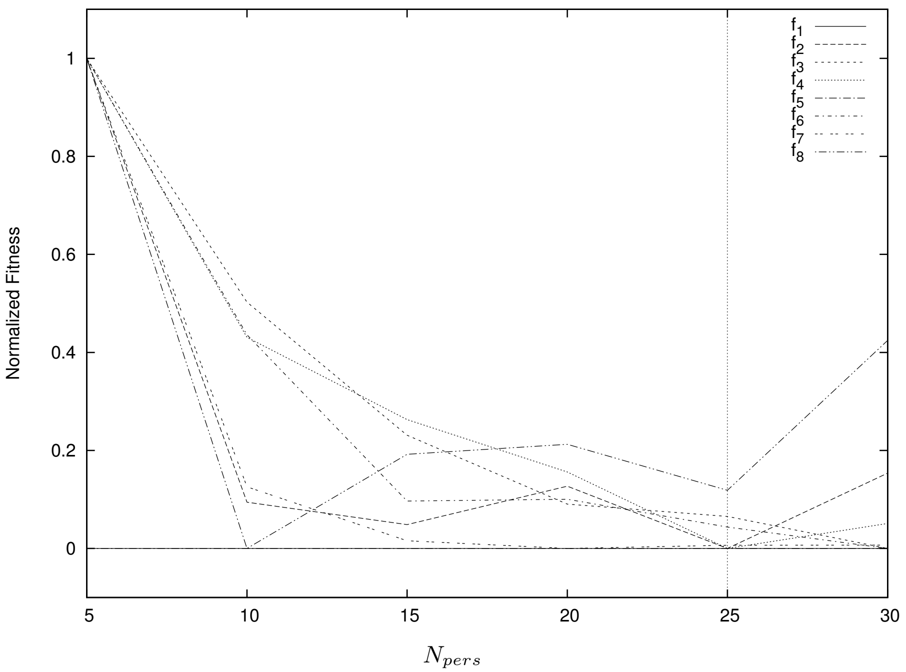


Fig. 5. The quality values for the N_{pers} parameter over all benchmark functions

DE experiments were conducted with CR=0.5 as a trade-off. Additionally, the self-adaptive differential evolution approach was used to allow for an automatic adaptation of the CR and F parameters.

In order to get an impression of the performance, experiments were conducted for the lower dimensions $D = 10\ , 20\ , \cdots\ , 100$ as well as for the higher dimensions $D = 200\ , 400\ , \cdots\ , 1000$. For each benchmark function and dimension, 100 independent runs with 100000 evaluation steps were performed. In FIPS, the number of evaluations is constant per iteration. Thus, the total number can be computed by *size of population* × *number of iterations*. The same holds for DE and SaDE. Unfortunately, in CO the number of evaluations per iteration is not known *a priori*. It depends on the expertise of the person, whether for a given topic a contribution is done or not. And the quality function is only used, if a person contributes. Thus, the experiments were fixed to 100000 evaluation steps in total for all used methods. In order to keep as comparable as possible, all methods used the same size of population. The numbers of iterations were adapted to realize the needed number of evaluation steps. Furthermore, the rates of perfect-hits were determined, whereby a perfect-hit is defined as reaching a threshold of nearness to the global optimum. Some authors like [19] call this threshold also VTR (value to reach) and their VTR=1e-6 was adopted as threshold of nearness.

Since all used benchmark functions are shifted, the c parameter needs to be adjusted. Following [3], c is set to be halfway between the origin and the upper bound: $c = \frac{U_t}{2}$.

All methods used standard parameters. The lower and upper bounds of all methods were set according to the benchmark functions (see section 3). In Table 2 the parameters are given in detail. The parameters of CO were the ones determined by the systematic parameter exploration as presented in section 4.

In the following subsections, the "winners" of the different benchmark functions are collected. A method is considered to be a winner, if it has the best performance for the highest number of dimensions. For example, let A and B be two methods and A be the best in 3 cases (10-30 dimensions) and B be the best for the remaining 7 cases (40-100 dimensions). Then, method B is the winner. In cases where the two best methods are the best for the same number of dimensions (e.g., both for 5), both are considered to be the winners. The performance of a method is measured in terms of the mean quality. In cases where several methods have equivalent mean qualities, the method with the smallest standard deviation is chosen as winner.

5.1 Shifted Sphere (f_1)

Figure 6 (page 26) reveals that the lower dimensions of the Shifted Sphere benchmark can be solved well by all four methods. Their qualities only differ w.r.t. the standard deviation and the best two are DE followed by FIPS. Except SaDE, all reached perfect-hit rates of 100% over all lower dimensions as illustrated in Fig. 8 (page 26). As can be seen in Fig. 7 (page 26) and Table 4 (page 27), the higher

Table 2. The used parameters for CO, FIPS and all DE variants

CO:		**FIPS:**	
N_{pers}	= 25	particles	= 25
λ_C	= 0.3	χ	= 0.729843788
λ_L	= 1.6	V_{max}	= 4.1
τ	= 0	c_1	= 2.05
evaluations	= 100000	c_2	= 2.05
		Φ_{max}	= 4.1
		topology	= *l*-best
		radius	= 1
		iterations	= 4000
DE:		**SaDE:**	
vectors	= 25	vectors	= 25
F	= 0.5	F_{mean}	= 0.5 (sd: 0.3)
CR	= 0.5	CR_{mean}	= 0.5 (sd: 0.1)
scheme	= DE/rand/1/bin	iterations	= 4000
iterations	= 4000		

dimensions are best solved by CO and second best by FIPS. They significantly outperformed both DE variants.

5.2 Shifted Rosenbrock (f_2)

As presented in Fig. 9 (page 28), Shifted Rosenbrock is best solved by FIPS and second best by SaDE for the lower dimensions up to 100. CO was outperformed by all other methods. The obtained rates of perfect-hits as shown in Fig. 11 (page 28) reveal that the optimum of this benchmark was not reached by any method but SaDE for 10 dimensions. The higher dimensions are best solved by CO and second best by FIPS as depicted in Fig. 10 (page 28) and Table 5 (page 29).

5.3 Shifted Rastrigin (f_3)

As depicted in Figs. 12 (page 30) and 14 (page 30), Shifted Rastrigin is best solved by CO and second best by SaDE for the lower dimensions. As can be seen in Fig. 13 (page 30) and Table 6 (page 31), the higher dimensions are best solved by CO and second best by SaDE, too, whereby CO significantly outperformed all other methods.

5.4 Shifted Griewank (f_4)

Figs. 15 (page 32) and 17 (page 32) reveal that Shifted Griewank is best solved by DE and second best by FIPS for the lower dimensions. The higher dimensions are best solved by CO and second best by FIPS, as shown in Fig. 16 (page 32).

5.5 Shifted Ackley (f_5)

According to Table 8 (page 35) and Fig. 18 (page 34), the first half of the lower dimensions (10-50) is best solved by DE and the second half (60-100) is best solved by CO. The second best is FIPS. As shown in Fig. 20 (page 34), CO reached always a 100% perfect-hit rate over the lower dimensions, whereby DE and FIPS reached this performance only up to 60 dimensions. The higher dimensions are best solved by CO and second best by SaDE as revealed by Fig. 19 (page 34). In fact, CO significantly outperformed all other methods, whereas FIPS is the worse of all.

5.6 Shifted Schwefel 2.21 (f_6)

According to Table 9 (page 37) and Fig. 21 (page 36), CO is the clear winner for lower dimensions as well as for the best perfect-hit rates (see Fig. 23 (page 36)). The second best is FIPS. As revealed by Fig. 22 (page 36), the higher dimensions are best solved by FIPS and second best by SaDE.

5.7 Shifted Schaffer (f_7)

As shown in Fig. 24 (page 38) and Table 10 (page 39), the lower dimensions are best solved by DE and second best by CO. The FIPS approach was outperformed by all other methods. As depicted in Fig. 25 (page 38), the higher dimensions are best solved by CO, which significantly outperformed all other methods. The second best was DE.

5.8 Shifted Schwefel 1.2 (f_8)

According to Table 11 (page 41) and Fig. 27 (page 40), FIPS is the winner and SaDE the second winner for lower dimensions. Although CO has the better graph, SaDE is the second winner, because SaDE wins more dimensions (2) than CO (0). As revealed by Fig. 28 (page 40), the higher dimensions are best solved by FIPS and second best by CO.

5.9 Summary

In Table 3 (page 25) the winners of all eight benchmark functions are collected. Additionally to the winners, the appropriate second bests are also contained in the form like: (2nd: DE).

Table 3. The winners of all benchmark functions. A winner is a method that reached the best performance over the most dimensions. The methods written like (2nd: DE) are the appropriate second bests. The properties of the functions are given like (U - S - D), whereby the first letter can be **U**nimodal / **M**ultimodal, the second one can be **S**eparable / **N**on-separable and the last one can be **D**imension-wise / Not dimension-wise (**ND**).

f_i Name	Lower Dimensions (10-100)	Higher Dimensions (200-1000)
f_1 Shifted Sphere	**DE**	**CO**
(U - S - D)	(2nd: FIPS)	(2nd: FIPS)
f_2 Shifted Rosenbrock	**FIPS**	**CO**
(M - N - D)	(2nd: SaDE)	(2nd: FIPS)
f_3 Shifted Rastrigin	**CO**	**CO**
(M - S - D)	(2nd: SaDE)	(2nd: SaDE)
f_4 Shifted Griewank	**DE**	**CO**
(M - N - ND)	(2nd: FIPS)	(2nd: FIPS)
f_5 Shifted Ackley	**CO , DE**	**CO**
(M - S - D)	(2nd: FIPS)	(2nd: SaDE)
f_6 Shifted Schwefel 2.21	**CO**	**FIPS**
(U - N - ND)	(2nd: FIPS)	(2nd: SaDE)
f_7 Shifted Schaffer	**DE**	**CO**
(U - N - D)	(2nd: CO)	(2nd: DE)
f_8 Shifted Schwefel 1.2	**FIPS**	**FIPS**
(U - N - ND)	(2nd: SaDE)	(2nd: CO)
$\sum$ CO	3	6
$\sum$ FIPS	2	2
$\sum$ DE	4	0

For the lower dimensions, DE is the winner considered over all benchmark functions used. But the differences to CO and FIPS are that small that actually no one of the three methods is a clear winner. The performance of SaDE is only good enough for three second places. It seems that the advantage of SaDE that one does not need to adjust the F and CR parameters comes at the cost of performance.

For the higher dimensions, CO is the clear winner considered over all used benchmark functions. The second best is the FIPS approach. It seems that CO is of high value for high dimensional problems.

Considered over all 16 different problems (eight benchmark functions for lower as well as higher dimensions), the following results are obtained: CO is the winner in 9 cases (3 low and 6 high dimensional problems), FIPS wins in 4 cases (2 low and 2 high dimensional problems) and DE also wins in 4 cases (4 low and 0 high dimensional problems).

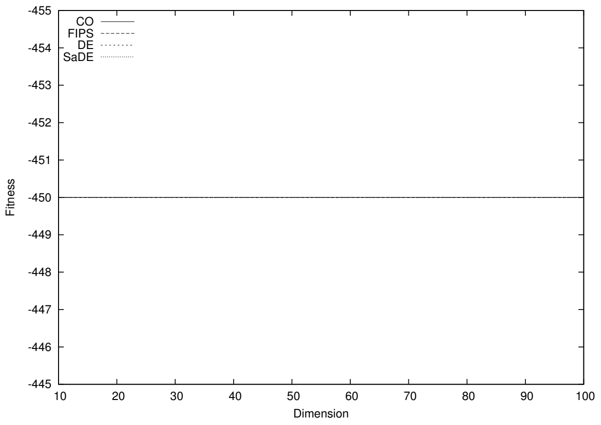


Fig. 6. The quality for the **Shifted Sphere** Benchmark over all **low** dimensions

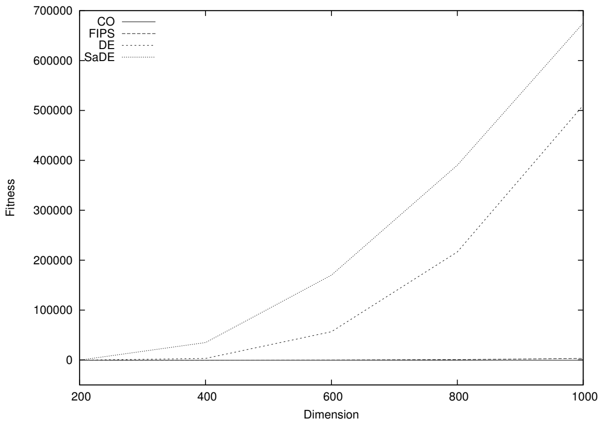


Fig. 7. The quality for the **Shifted Sphere** Benchmark over all **high** dimensions

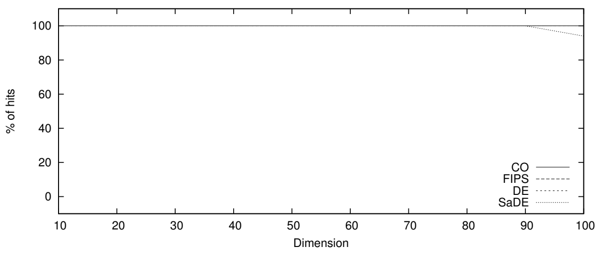


Fig. 8. The perfect-hit rates for the **Shifted Sphere** Benchmark over all **low** dimensions

Table 4. The obtained results for the **SHIFTED SPHERE** benchmark averaged over 100 independent runs. The columns 'Avg. Quality' are the best quality values reached on average and 'sd' are the appropriate standard deviations.

	CO	**FIPS**	**DE**	**SaDE**
D	Avg. Quality	Avg. Quality	Avg. Quality	Avg. Quality
10	-4.500000e+002	**-4.500000e+002**	**-4.500000e+002**	**-4.500000e+002**
	(sd:7.190187e-014)	(sd:0)	(sd:0)	(sd:0)
20	-4.500000e+002	-4.500000e+002	**-4.500000e+002**	-4.500000e+002
	(sd:7.958079e-014)	(sd:2.126886e-014)	(sd:0)	(sd:2.273737e-014)
30	-4.500000e+002	**-4.500000e+002**	-4.500000e+002	-4.500000e+002
	(sd:1.523149e-013)	(sd:9.845569e-015)	(sd:3.457650e-014)	(sd:2.784747e-014)
40	-4.500000e+002	**-4.500000e+002**	-4.500000e+002	-4.500000e+002
	(sd:2.239371e-013)	(sd:9.845569e-015)	(sd:5.511174e-014)	(sd:6.938624e-014)
50	-4.500000e+002	-4.500000e+002	**-4.500000e+002**	-4.500000e+002
	(sd:2.924537e-013)	(sd:4.439613e-014)	(sd:0)	(sd:2.055815e-013)
60	-4.500000e+002	-4.500000e+002	**-4.500000e+002**	-4.500000e+002
	(sd:3.442666e-013)	(sd:5.824717e-014)	(sd:0)	(sd:1.031331e-011)
70	-4.500000e+002	-4.500000e+002	**-4.500000e+002**	-4.500000e+002
	(sd:3.194883e-013)	(sd:8.658132e-014)	(sd:1.969114e-014)	(sd:1.287300e-012)
80	-4.500000e+002	-4.500000e+002	**-4.500000e+002**	-4.500000e+002
	(sd:3.815714e-013)	(sd:1.215174e-013)	(sd:4.987990e-014)	(sd:6.487610e-011)
90	-4.500000e+002	**-4.500000e+002**	-4.500000e+002	-4.500000e+002
	(sd:4.375467e-013)	(sd:1.615794e-013)	(sd:2.474485e-013)	(sd:2.666406e-008)
100	-4.500000e+002	**-4.500000e+002**	-4.500000e+002	-4.500000e+002
	(sd:4.776200e-013)	(sd:2.381325e-013)	(sd:6.145837e-011)	(sd:2.286373e-004)
200	**-4.500000e+002**	-4.499999e+002	-4.498360e+002	-2.614953e+002
	(sd:1.119360e-007)	(sd:1.238909e-004)	(sd:2.815927e-001)	(sd:5.131776e+002)
400	**-4.499929e+002**	-4.387593e+002	2.996741e+003	3.493584e+004
	(sd:1.546788e-002)	(sd:5.856080e+000)	(sd:1.714725e+003)	(sd:1.371262e+004)
600	**-4.492890e+002**	-1.959629e+002	5.658269e+004	1.700624e+005
	(sd:1.182685e+000)	(sd:4.298016e+001)	(sd:1.095679e+004)	(sd:3.763572e+004)
800	**-4.388396e+002**	7.729272e+002	2.165145e+005	3.903596e+005
	(sd:9.009993e+000)	(sd:1.347133e+002)	(sd:2.545970e+004)	(sd:5.627727e+004)
1000	**-3.890554e+002**	3.049089e+003	5.103793e+005	6.750752e+005
	(sd:1.928671e+001)	(sd:2.565870e+002)	(sd:7.437225e+004)	(sd:7.924662e+004)

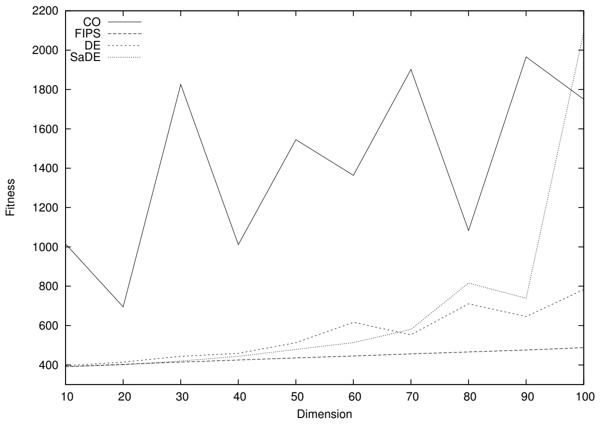


Fig. 9. The quality for the **Shifted Rosenbrock** Benchmark over all **low** dimensions

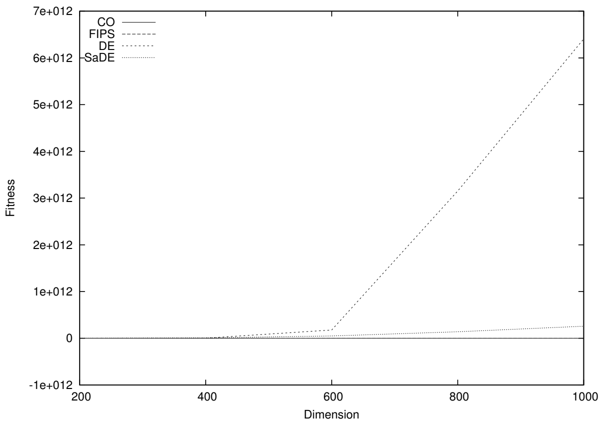


Fig. 10. The quality for the **Shifted Rosenbrock** Benchmark over all **high** dimensions

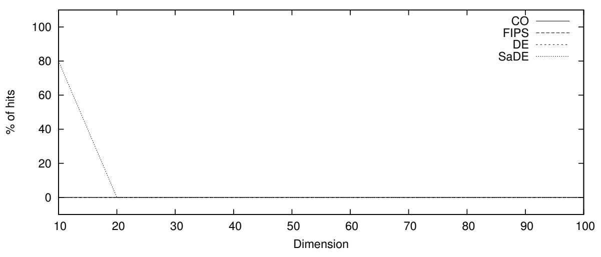


Fig. 11. The perfect-hit rates for the **Shifted Rosenbrock** Benchmark over all **low** dimensions

Table 5. The obtained results for the **SHIFTED ROSENBROCK** benchmark averaged over 100 independent runs. The columns 'Avg. Quality' are the best quality values reached on average and 'sd' are the appropriate standard deviations.

	CO	**FIPS**	**DE**	**SaDE**
D	Avg. Quality	Avg. Quality	Avg. Quality	Avg. Quality
10	1.014814e+003	3.911184e+002	3.960845e+002	**3.905231e+002**
	(sd:3.683167e+003)	(sd:3.318371e-001)	(sd:5.392485e+000)	(sd:1.339300e+000)
20	6.952487e+002	4.032622e+002	4.143899e+002	**4.007489e+002**
	(sd:1.083385e+003)	(sd:1.016404e+000)	(sd:2.293293e+001)	(sd:1.192593e+001)
30	1.825208e+003	**4.142498e+002**	4.439706e+002	4.201108e+002
	(sd:7.383815e+003)	(sd:8.674111e-001)	(sd:6.625716e+001)	(sd:2.458557e+001)
40	1.011501e+003	**4.248906e+002**	4.587187e+002	4.439911e+002
	(sd:2.788764e+003)	(sd:1.409797e+000)	(sd:6.957866e+001)	(sd:3.591328e+001)
50	1.545018e+003	**4.355404e+002**	5.130391e+002	4.789255e+002
	(sd:5.616539e+003)	(sd:1.615432e+000)	(sd:3.015181e+002)	(sd:3.952624e+001)
60	1.363647e+003	**4.457168e+002**	6.167651e+002	5.135169e+002
	(sd:2.685179e+003)	(sd:1.473175e+000)	(sd:8.651087e+002)	(sd:8.084122e+001)
70	1.902235e+003	**4.563187e+002**	5.532552e+002	5.809877e+002
	(sd:5.126389e+003)	(sd:5.428972e+000)	(sd:3.877393e+002)	(sd:2.096486e+002)
80	1.083035e+003	**4.660917e+002**	7.105638e+002	8.158947e+002
	(sd:2.096649e+003)	(sd:1.444370e+000)	(sd:9.494556e+002)	(sd:1.679032e+003)
90	1.965833e+003	**4.759197e+002**	6.462474e+002	7.389693e+002
	(sd:5.883145e+003)	(sd:1.140886e+000)	(sd:4.314112e+002)	(sd:2.659029e+002)
100	1.750359e+003	**4.877016e+002**	7.832532e+002	2.091274e+003
	(sd:3.899240e+003)	(sd:8.371906e+000)	(sd:1.326696e+003)	(sd:1.248989e+004)
200	2.391920e+003	**6.969935e+002**	1.878450e+006	4.994059e+006
	(sd:4.617019e+003)	(sd:6.243538e+001)	(sd:8.892477e+006)	(sd:1.405947e+007)
400	**2.608685e+003**	1.483319e+005	3.128679e+009	8.055967e+009
	(sd:3.656561e+003)	(sd:5.184172e+004)	(sd:1.768106e+009)	(sd:4.765960e+009)
600	**6.528941e+003**	9.058607e+006	1.774271e+011	4.886980e+010
	(sd:1.029473e+004)	(sd:1.153290e+006)	(sd:5.517890e+011)	(sd:1.473563e+010)
800	**2.373496e+004**	1.297131e+008	3.150051e+012	1.393014e+011
	(sd:6.546513e+004)	(sd:1.421733e+007)	(sd:2.446747e+012)	(sd:2.700521e+010)
1000	**8.005813e+004**	7.837178e+008	6.404477e+012	2.583907e+011
	(sd:1.091941e+005)	(sd:6.927078e+007)	(sd:1.748920e+012)	(sd:3.563682e+010)

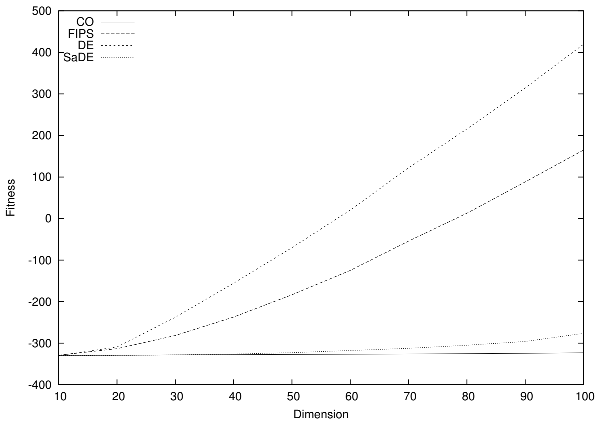


Fig. 12. The quality for the **Shifted Rastrigin** Benchmark over all **low** dimensions

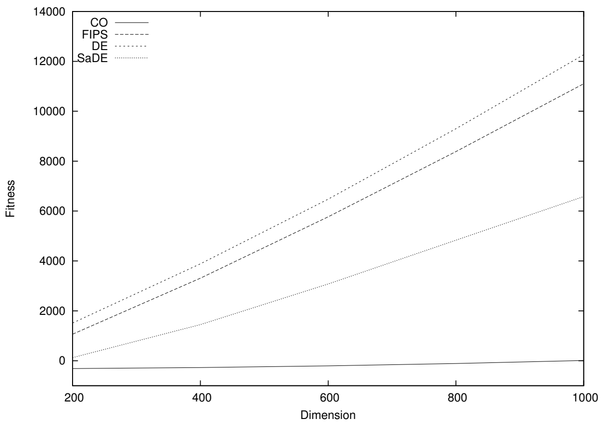


Fig. 13. The quality for the **Shifted Rastrigin** Benchmark over all **high** dimensions

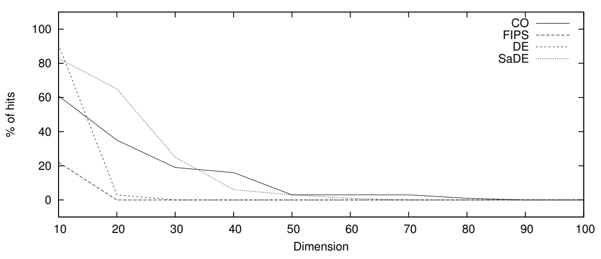


Fig. 14. The perfect-hit rates for the **Shifted Rastrigin** Benchmark over all **low** dimensions

Table 6. The obtained results for the **SHIFTED RASTRIGIN** benchmark averaged over 100 independent runs. The columns 'Avg. Quality' are the best quality values reached on average and 'sd' are the appropriate standard deviations.

	CO	**FIPS**	**DE**	**SaDE**
D	Avg. Quality	Avg. Quality	Avg. Quality	Avg. Quality
10	-3.295324e+002 (sd:6.363850e-001)	-3.290894e+002 (sd:1.042462e+000)	**-3.299105e+002** (sd:2.847391e-001)	-3.298209e+002 (sd:4.073261e-001)
20	-3.289752e+002 (sd:1.028761e+000)	-3.132639e+002 (sd:5.453842e+000)	-3.092145e+002 (sd:8.789442e+000)	**-3.294528e+002** (sd:1.047073e+000)
30	**-3.282688e+002** (sd:1.446083e+000)	-2.814818e+002 (sd:1.038143e+001)	-2.376072e+002 (sd:1.052276e+001)	-3.281494e+002 (sd:2.063283e+000)
40	**-3.277514e+002** (sd:1.668568e+000)	-2.371213e+002 (sd:2.009253e+001)	-1.556737e+002 (sd:1.168647e+001)	-3.263087e+002 (sd:3.144112e+000)
50	**-3.269057e+002** (sd:1.628179e+000)	-1.835899e+002 (sd:2.187269e+001)	-7.010259e+001 (sd:1.618482e+001)	-3.226970e+002 (sd:5.393398e+000)
60	**-3.269111e+002** (sd:1.857835e+000)	-1.247690e+002 (sd:2.885090e+001)	2.052051e+001 (sd:1.823677e+001)	-3.176725e+002 (sd:8.298072e+000)
70	**-3.260639e+002** (sd:1.978296e+000)	-5.434455e+001 (sd:3.262892e+001)	1.219691e+002 (sd:1.846041e+001)	-3.122997e+002 (sd:8.243492e+000)
80	**-3.252322e+002** (sd:2.228190e+000)	1.270795e+001 (sd:3.420864e+001)	2.157519e+002 (sd:2.144012e+001)	-3.050664e+002 (sd:1.069179e+001)
90	**-3.243360e+002** (sd:2.469812e+000)	8.816808e+001 (sd:4.155467e+001)	3.143240e+002 (sd:2.229105e+001)	-2.958154e+002 (sd:1.358818e+001)
100	**-3.232379e+002** (sd:2.703923e+000)	1.647245e+002 (sd:4.785285e+001)	4.196112e+002 (sd:2.865505e+001)	-2.765022e+002 (sd:1.920225e+001)
200	**-3.119726e+002** (sd:4.164903e+000)	1.064771e+003 (sd:8.369567e+001)	1.512219e+003 (sd:5.653973e+001)	1.251842e+002 (sd:3.985319e+001)
400	**-2.715822e+002** (sd:6.916448e+000)	3.308359e+003 (sd:1.577501e+002)	3.883808e+003 (sd:1.204154e+002)	1.447845e+003 (sd:9.001163e+001)
600	**-2.041250e+002** (sd:1.146317e+001)	5.770185e+003 (sd:2.440953e+002)	6.475049e+003 (sd:2.105878e+002)	3.077570e+003 (sd:1.247022e+002)
800	**-1.089592e+002** (sd:1.420523e+001)	8.383482e+003 (sd:2.960762e+002)	9.298387e+003 (sd:2.677412e+002)	4.832295e+003 (sd:2.455763e+002)
1000	**1.012936e+001** (sd:2.036126e+001)	1.110240e+004 (sd:2.874027e+002)	1.226412e+004 (sd:3.597017e+002)	6.580697e+003 (sd:3.099531e+002)

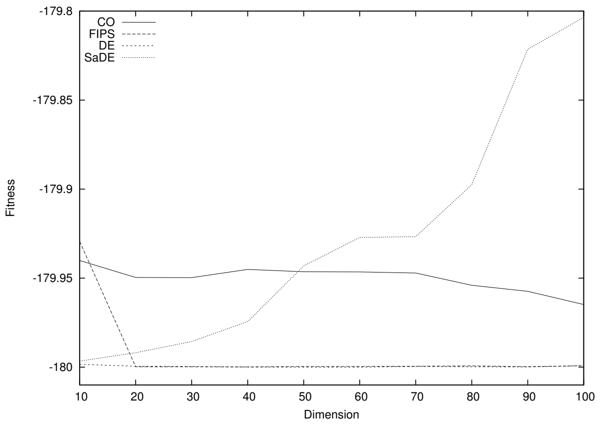


Fig. 15. The quality for the **Shifted Griewank** Benchmark over all **low** dimensions

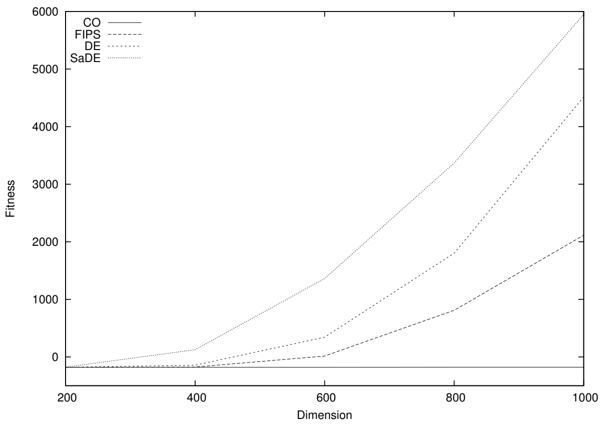


Fig. 16. The quality for the **Shifted Griewank** Benchmark over all **high** dimensions

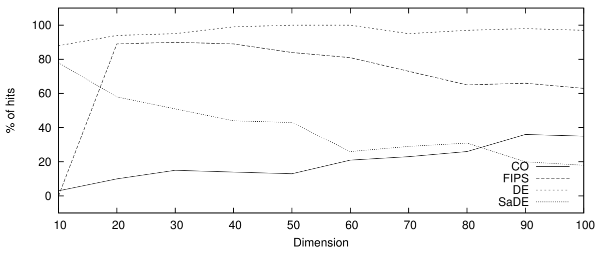


Fig. 17. The perfect-hit rates for the **Shifted Griewank** Benchmark over all **low** dimensions

Table 7. The obtained results for the **SHIFTED GRIEWANK** benchmark averaged over 100 independent runs. The columns 'Avg. Quality' are the best quality values reached on average and 'sd' are the appropriate standard deviations.

	CO	**FIPS**	**DE**	**SaDE**
D	Avg. Quality	Avg. Quality	Avg. Quality	Avg. Quality
10	-1.799401e+002	-1.799294e+002	**-1.799984e+002**	-1.799966e+002
	(sd:4.134539e-002)	(sd:4.411139e-002)	(sd:4.873407e-003)	(sd:7.466981e-003)
20	-1.799496e+002	**-1.799998e+002**	-1.799995e+002	-1.799919e+002
	(sd:4.460841e-002)	(sd:1.037936e-003)	(sd:2.099964e-003)	(sd:1.337185e-002)
30	-1.799497e+002	**-1.799998e+002**	-1.799996e+002	-1.799856e+002
	(sd:4.902744e-002)	(sd:1.037581e-003)	(sd:1.846288e-003)	(sd:2.118928e-002)
40	-1.799451e+002	-1.799999e+002	**-1.799999e+002**	-1.799744e+002
	(sd:5.152321e-002)	(sd:7.463596e-004)	(sd:7.358967e-004)	(sd:3.878824e-002)
50	-1.799464e+002	-1.799996e+002	**-1.800000e+002**	-1.799431e+002
	(sd:5.054002e-002)	(sd:2.271104e-003)	(sd:2.842171e-015)	(sd:1.043619e-001)
60	-1.799465e+002	-1.799996e+002	**-1.800000e+002**	-1.799271e+002
	(sd:5.852028e-002)	(sd:1.817196e-003)	(sd:0)	(sd:1.212855e-001)
70	-1.799471e+002	-1.799995e+002	**-1.799996e+002**	-1.799267e+002
	(sd:6.146186e-002)	(sd:2.185108e-003)	(sd:1.611930e-003)	(sd:1.201846e-001)
80	-1.799540e+002	-1.799993e+002	**-1.799998e+002**	-1.798975e+002
	(sd:5.818864e-002)	(sd:2.049870e-003)	(sd:1.261670e-003)	(sd:1.666943e-001)
90	-1.799574e+002	-1.799997e+002	**-1.799999e+002**	-1.798214e+002
	(sd:6.310732e-002)	(sd:1.455247e-003)	(sd:1.035446e-003)	(sd:3.817112e-001)
100	-1.799648e+002	**-1.799992e+002**	-1.799992e+002	-1.798035e+002
	(sd:5.386163e-002)	(sd:2.465578e-003)	(sd:6.849497e-003)	(sd:3.111143e-001)
200	-1.799546e+002	**-1.799783e+002**	-1.799668e+002	-1.769174e+002
	(sd:8.186290e-002)	(sd:2.442928e-002)	(sd:3.697388e-002)	(sd:6.219243e+000)
400	**-1.799596e+002**	-1.765999e+002	-1.435568e+002	1.242539e+002
	(sd:9.081599e-002)	(sd:5.284919e-001)	(sd:2.044130e+001)	(sd:1.286640e+002)
600	**-1.798772e+002**	1.577781e+001	3.425431e+002	1.361870e+003
	(sd:1.091613e-001)	(sd:2.337765e+001)	(sd:9.775039e+001)	(sd:3.537492e+002)
800	**-1.791875e+002**	8.092041e+002	1.801778e+003	3.366892e+003
	(sd:2.087905e-001)	(sd:6.377247e+001)	(sd:2.603267e+002)	(sd:5.027938e+002)
1000	**-1.784840e+002**	2.113870e+003	4.515258e+003	5.946979e+003
	(sd:2.061974e-001)	(sd:1.032748e+002)	(sd:7.227539e+002)	(sd:6.796047e+002)

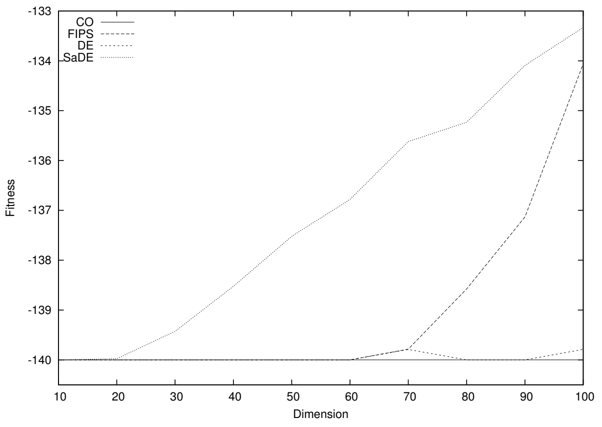


Fig. 18. The quality for the **Shifted Ackley** Benchmark over all **low** dimensions

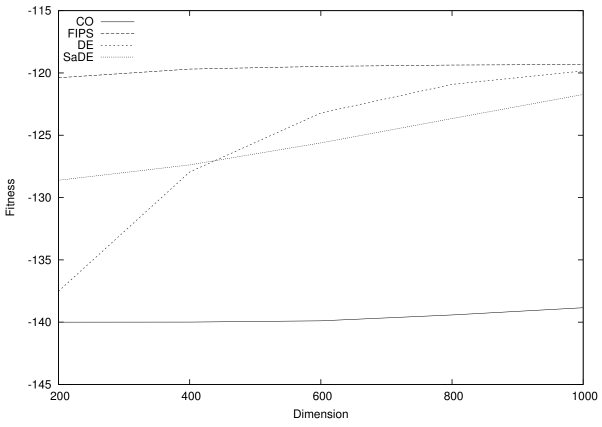


Fig. 19. The quality for the **Shifted Ackley** Benchmark over all **high** dimensions

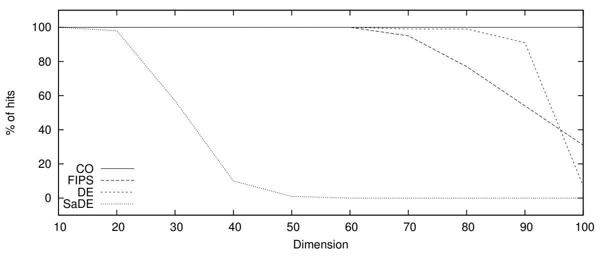


Fig. 20. The perfect-hit rates for the **Shifted Ackley** Benchmark over all **low** dimensions

Table 8. The obtained results for the **SHIFTED ACKLEY** benchmark averaged over 100 independent runs. The columns 'Avg. Quality' are the best quality values reached on average and 'sd' are the appropriate standard deviations.

	CO	**FIPS**	**DE**	**SaDE**
D	Avg. Quality	Avg. Quality	Avg. Quality	Avg. Quality
10	-1.400000e+002 (sd:9.671737e-014)	-1.400000e+002 (sd:6.355287e-015)	**-1.400000e+002** (sd:0)	-1.400000e+002 (sd:4.019437e-015)
20	-1.400000e+002 (sd:1.594405e-013)	**-1.400000e+002** (sd:0)	**-1.400000e+002** (sd:0)	-1.399769e+002 (sd:1.617208e-001)
30	-1.400000e+002 (sd:2.291255e-013)	-1.400000e+002 (sd:3.492518e-014)	**-1.400000e+002** (sd:5.684342e-015)	-1.394264e+002 (sd:6.963515e-001)
40	-1.400000e+002 (sd:1.904042e-013)	-1.400000e+002 (sd:6.272125e-014)	**-1.400000e+002** (sd:2.770206e-014)	-1.385232e+002 (sd:6.670704e-001)
50	-1.400000e+002 (sd:2.337848e-013)	-1.400000e+002 (sd:1.405014e-011)	**-1.400000e+002** (sd:5.092168e-014)	-1.375221e+002 (sd:1.063325e+000)
60	**-1.400000e+002** (sd:1.999240e-013)	-1.400000e+002 (sd:2.748646e-011)	-1.400000e+002 (sd:8.965047e-012)	-1.367800e+002 (sd:1.342065e+000)
70	**-1.400000e+002** (sd:2.691822e-013)	-1.397847e+002 (sd:1.879027e+000)	-1.397910e+002 (sd:2.079796e+000)	-1.356196e+002 (sd:1.593818e+000)
80	**-1.400000e+002** (sd:1.065314e-012)	-1.385754e+002 (sd:4.275715e+000)	-1.400000e+002 (sd:3.102383e-005)	-1.352316e+002 (sd:1.625549e+000)
90	**-1.400000e+002** (sd:3.009068e-011)	-1.371323e+002 (sd:6.091536e+000)	-1.399999e+002 (sd:1.231362e-003)	-1.340940e+002 (sd:1.771462e+000)
100	**-1.400000e+002** (sd:2.695251e-010)	-1.340593e+002 (sd:7.895980e+000)	-1.397890e+002 (sd:2.098857e+000)	-1.333304e+002 (sd:1.752582e+000)
200	**-1.400000e+002** (sd:9.158325e-006)	-1.203831e+002 (sd:4.315355e-001)	-1.375227e+002 (sd:4.997207e+000)	-1.286100e+002 (sd:1.944420e+000)
400	**-1.399934e+002** (sd:3.520323e-003)	-1.196892e+002 (sd:1.240444e-001)	-1.279517e+002 (sd:4.159970e+000)	-1.273760e+002 (sd:1.164288e+000)
600	**-1.398914e+002** (sd:3.555859e-002)	-1.194773e+002 (sd:7.444760e-002)	-1.232114e+002 (sd:2.031766e+000)	-1.256143e+002 (sd:7.905553e-001)
800	**-1.394189e+002** (sd:9.910390e-002)	-1.193695e+002 (sd:4.414739e-002)	-1.209179e+002 (sd:1.025148e+000)	-1.236581e+002 (sd:4.525474e-001)
1000	**-1.388388e+002** (sd:9.619811e-002)	-1.193211e+002 (sd:4.799817e-002)	-1.198350e+002 (sd:6.157377e-001)	-1.217219e+002 (sd:7.267552e-001)

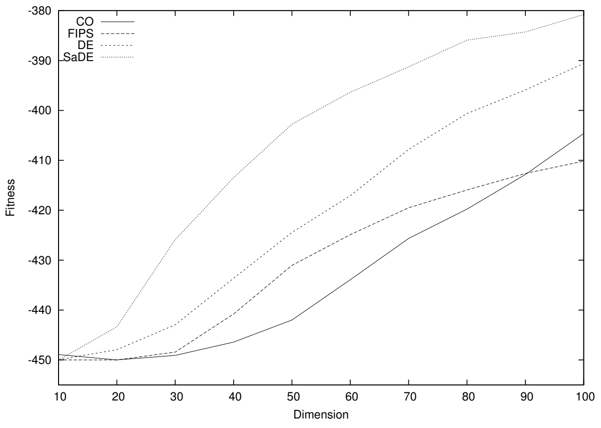


Fig. 21. The quality for the **Shifted Schwefel 2.21** Benchmark over all **low** dimensions

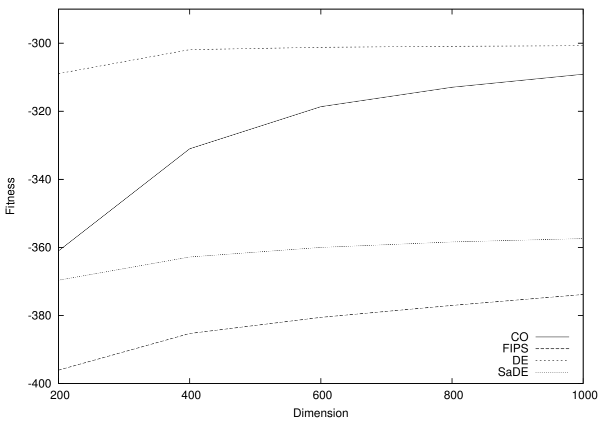


Fig. 22. The quality for the **Shifted Schwefel 2.21** Benchmark over all **high** dimensions

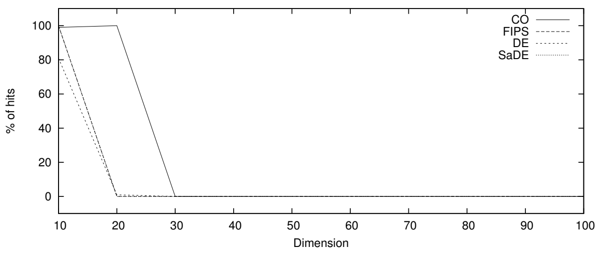


Fig. 23. The perfect-hit rates for the **Shifted Schwefel 2.21** Benchmark over all **low** dimensions

Table 9. The obtained results for the **SHIFTED SCHWEFEL 2.21** benchmark averaged over 100 independent runs. The columns 'Avg. Quality' are the best quality values reached on average and 'sd' are the appropriate standard deviations.

	CO	**FIPS**	**DE**	**SaDE**
D	Avg. Quality	Avg. Quality	Avg. Quality	Avg. Quality
10	-4.489104e+002 (sd:1.084125e+001)	**-4.500000e+002** (sd:4.755862e-014)	-4.499584e+002 (sd:2.611637e-001)	-4.500000e+002 (sd:1.488578e-008)
20	**-4.500000e+002** (sd:1.723247e-007)	-4.499937e+002 (sd:1.650810e-002)	-4.479376e+002 (sd:4.454956e+000)	-4.433669e+002 (sd:5.096988e+000)
30	**-4.490855e+002** (sd:8.569079e+000)	-4.484360e+002 (sd:1.044253e+000)	-4.429598e+002 (sd:7.161370e+000)	-4.258889e+002 (sd:6.500815e+000)
40	**-4.464303e+002** (sd:1.140033e+001)	-4.407844e+002 (sd:3.544000e+000)	-4.336153e+002 (sd:1.046324e+001)	-4.134643e+002 (sd:5.906191e+000)
50	**-4.419887e+002** (sd:1.202361e+001)	-4.310389e+002 (sd:2.889775e+000)	-4.244601e+002 (sd:1.138035e+001)	-4.027587e+002 (sd:5.703359e+000)
60	**-4.339265e+002** (sd:1.561768e+001)	-4.249005e+002 (sd:2.679366e+000)	-4.170774e+002 (sd:1.278055e+001)	-3.963494e+002 (sd:6.363289e+000)
70	**-4.256561e+002** (sd:1.544347e+001)	-4.195058e+002 (sd:2.645905e+000)	-4.078050e+002 (sd:1.151013e+001)	-3.912661e+002 (sd:5.473383e+000)
80	**-4.197817e+002** (sd:1.149723e+001)	-4.159261e+002 (sd:2.659169e+000)	-4.006189e+002 (sd:1.244508e+001)	-3.859185e+002 (sd:5.194362e+000)
90	**-4.128266e+002** (sd:9.896689e+000)	-4.126387e+002 (sd:2.503374e+000)	-3.958927e+002 (sd:9.457541e+000)	-3.842713e+002 (sd:5.239893e+000)
100	-4.046237e+002 (sd:1.790226e+001)	**-4.101617e+002** (sd:2.395554e+000)	-3.906002e+002 (sd:1.222260e+001)	-3.807847e+002 (sd:4.366847e+000)
200	-3.610515e+002 (sd:1.712589e+001)	**-3.960883e+002** (sd:2.005508e+000)	-3.089553e+002 (sd:7.035707e+000)	-3.696587e+002 (sd:3.775774e+000)
400	-3.310405e+002 (sd:1.242192e+001)	**-3.853219e+002** (sd:1.688922e+000)	-3.019135e+002 (sd:5.938256e-001)	-3.628508e+002 (sd:2.878341e+000)
600	-3.186619e+002 (sd:9.011174e+000)	**-3.805799e+002** (sd:1.519170e+000)	-3.012224e+002 (sd:4.026564e-001)	-3.600298e+002 (sd:3.102008e+000)
800	-3.129529e+002 (sd:6.862525e+000)	**-3.770601e+002** (sd:1.584371e+000)	-3.009255e+002 (sd:3.127022e-001)	-3.584252e+002 (sd:2.538747e+000)
1000	-3.091179e+002 (sd:5.541339e+000)	**-3.738526e+002** (sd:1.802341e+000)	-3.007410e+002 (sd:2.260200e-001)	-3.574285e+002 (sd:3.087123e+000)

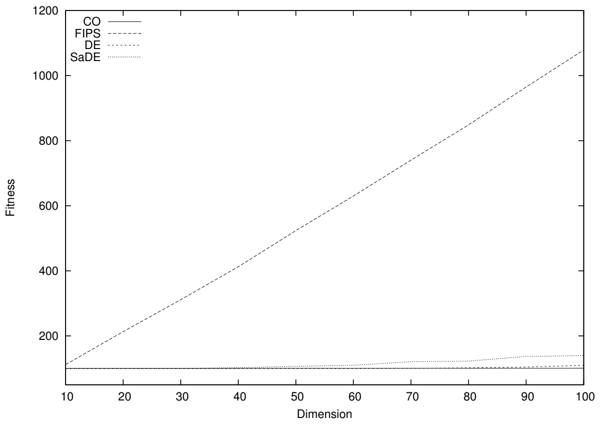


Fig. 24. The quality for the **Shifted Schaffer** Benchmark over all **low** dimensions

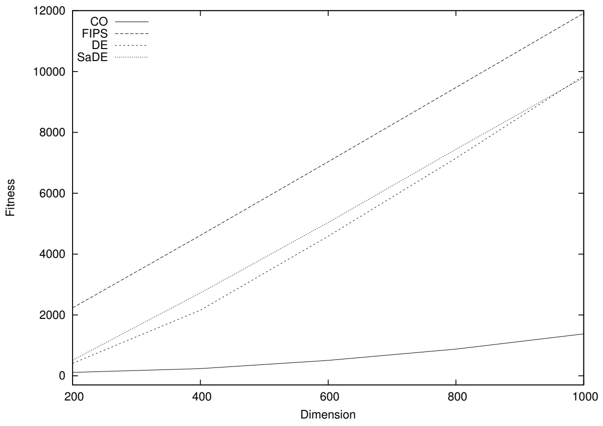


Fig. 25. The quality for the **Shifted Schaffer** Benchmark over all **high** dimensions

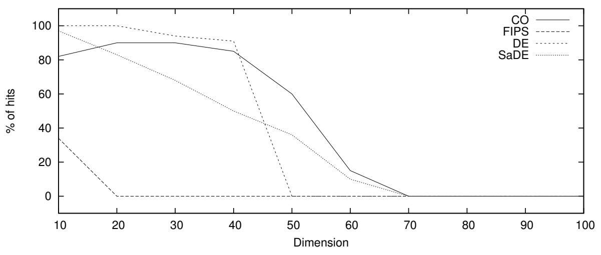


Fig. 26. The perfect-hit rates for the **Shifted Schaffer** Benchmark over all **low** dimensions

Table 10. The obtained results for the **SHIFTED SCHAFFER** benchmark averaged over 100 independent runs. The columns 'Avg. Quality' are the best quality values reached on average and 'sd' are the appropriate standard deviations.

	CO	**FIPS**	**DE**	**SaDE**
D	Avg. Quality	Avg. Quality	Avg. Quality	Avg. Quality
10	1.000380e+002	1.120205e+002	**1.000000e+002**	1.000003e+002
	(sd:1.502848e-001)	(sd:1.409892e+001)	(sd:0)	(sd:1.830224e-003)
20	1.000622e+002	2.133233e+002	**1.000000e+002**	1.000072e+002
	(sd:2.490667e-001)	(sd:1.434267e+001)	(sd:0)	(sd:3.038411e-002)
30	1.001193e+002	3.114448e+002	**1.000018e+002**	1.004810e+002
	(sd:4.704398e-001)	(sd:1.439638e+001)	(sd:8.129451e-003)	(sd:1.857596e+000)
40	1.001512e+002	4.126514e+002	**1.000035e+002**	1.025308e+002
	(sd:4.489124e-001)	(sd:1.457026e+001)	(sd:1.383280e-002)	(sd:5.334459e+000)
50	1.007304e+002	5.237790e+002	**1.000241e+002**	1.065268e+002
	(sd:3.080524e+000)	(sd:1.841096e+001)	(sd:1.230472e-001)	(sd:8.330295e+000)
60	1.002823e+002	6.299084e+002	**1.001393e+002**	1.099324e+002
	(sd:6.556228e-001)	(sd:1.795887e+001)	(sd:7.861875e-002)	(sd:1.104592e+001)
70	1.006115e+002	7.402884e+002	**1.005813e+002**	1.207612e+002
	(sd:2.413036e+000)	(sd:2.116258e+001)	(sd:4.055406e-001)	(sd:1.756741e+001)
80	**1.005003e+002**	8.486718e+002	1.018142e+002	1.228876e+002
	(sd:9.277055e-001)	(sd:2.060622e+001)	(sd:4.937126e-001)	(sd:2.226988e+001)
90	**1.007153e+002**	9.650093e+002	1.046122e+002	1.371699e+002
	(sd:1.403277e+000)	(sd:2.275197e+001)	(sd:8.757239e-001)	(sd:2.711086e+001)
100	**1.007828e+002**	1.078605e+003	1.095455e+002	1.395137e+002
	(sd:1.103401e+000)	(sd:2.472649e+001)	(sd:1.782662e+000)	(sd:2.787433e+001)
200	**1.125500e+002**	2.232428e+003	4.111508e+002	5.154736e+002
	(sd:4.583578e+000)	(sd:3.347757e+001)	(sd:3.420258e+001)	(sd:1.958782e+002)
400	**2.369664e+002**	4.615841e+003	2.155971e+003	2.726485e+003
	(sd:2.059438e+001)	(sd:4.709237e+001)	(sd:1.213979e+002)	(sd:1.992001e+002)
600	**5.070452e+002**	7.035535e+003	4.588681e+003	5.038908e+003
	(sd:3.568611e+001)	(sd:5.126754e+001)	(sd:1.678681e+002)	(sd:1.737125e+002)
800	**8.780283e+002**	9.473531e+003	7.151913e+003	7.442992e+003
	(sd:4.741353e+001)	(sd:6.236174e+001)	(sd:2.098281e+002)	(sd:1.815672e+002)
1000	**1.376414e+003**	1.191761e+004	9.858184e+003	9.803442e+003
	(sd:7.811933e+001)	(sd:7.084105e+001)	(sd:2.538452e+002)	(sd:1.729752e+002)

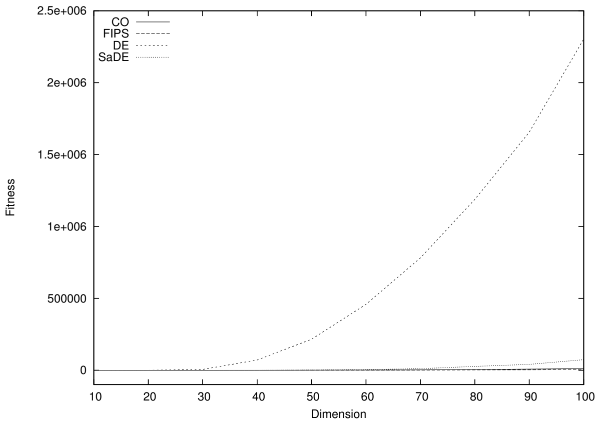


Fig. 27. The quality for the **Shifted Schwefel 1.2** Benchmark over all **low** dimensions

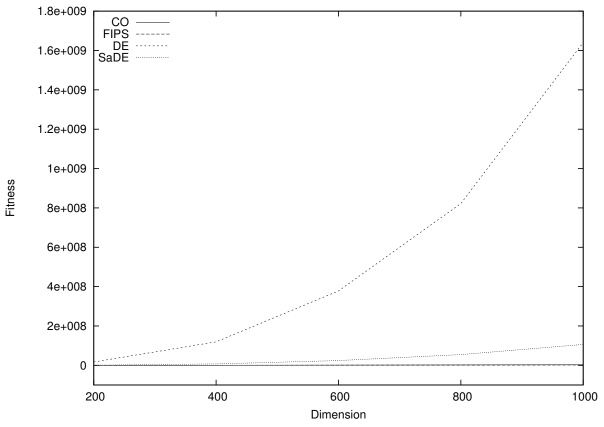


Fig. 28. The quality for the **Shifted Schwefel 1.2** Benchmark over all **high** dimensions

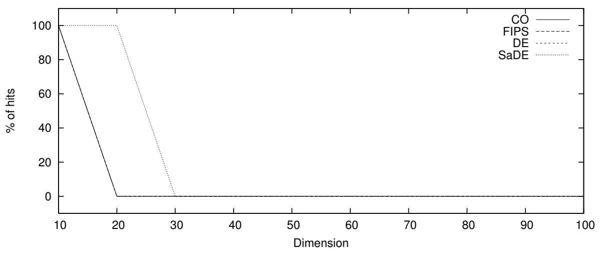


Fig. 29. The perfect-hit rates for the **Shifted Schwefel 1.2** Benchmark over all **low** dimensions

Table 11. The obtained results for the **SHIFTED SCHWEFEL 1.2** benchmark averaged over 100 independent runs. The columns 'Avg. Quality' are the best quality values reached on average and 'sd' are the appropriate standard deviations.

	CO	**FIPS**	**DE**	**SaDE**
D	Avg. Quality	Avg. Quality	Avg. Quality	Avg. Quality
10	1.000000e+002 (sd:1.817990e-013)	1.000000e+002 (sd:3.521311e-014)	**1.000000e+002** (sd:0)	1.000000e+002 (sd:1.503935e-014)
20	1.000004e+002 (sd:5.639871e-004)	1.000000e+002 (sd:4.968346e-005)	1.001896e+002 (sd:2.449396e-001)	**1.000000e+002** (sd:4.168713e-012)
30	1.016924e+002 (sd:1.716799e+000)	1.003658e+002 (sd:2.308946e-001)	5.591121e+003 (sd:2.933719e+003)	**1.000455e+002** (sd:3.601361e-001)
40	1.664022e+002 (sd:3.710328e+001)	**1.164544e+002** (sd:8.747295e+000)	7.137664e+004 (sd:2.479809e+004)	1.491467e+002 (sd:1.162605e+002)
50	4.857013e+002 (sd:2.176081e+002)	**2.147742e+002** (sd:4.432999e+001)	2.152550e+005 (sd:5.158667e+004)	1.063981e+003 (sd:1.130889e+003)
60	1.276157e+003 (sd:4.182165e+002)	**5.140837e+002** (sd:1.569317e+002)	4.580507e+005 (sd:1.003927e+005)	4.279357e+003 (sd:3.084658e+003)
70	2.675152e+003 (sd:7.003092e+002)	**1.013537e+003** (sd:2.152859e+002)	7.813793e+005 (sd:1.565684e+005)	1.114709e+004 (sd:5.399790e+003)
80	4.850910e+003 (sd:1.347817e+003)	**1.921741e+003** (sd:4.788760e+002)	1.188364e+006 (sd:2.491462e+005)	2.672155e+004 (sd:1.230631e+004)
90	7.861116e+003 (sd:1.810027e+003)	**3.075513e+003** (sd:6.263259e+002)	1.655391e+006 (sd:2.918856e+005)	4.064478e+004 (sd:1.390363e+004)
100	1.135344e+004 (sd:2.637208e+003)	**4.716127e+003** (sd:8.648291e+002)	2.303213e+006 (sd:4.288531e+005)	7.320724e+004 (sd:2.860607e+004)
200	8.525638e+004 (sd:1.126252e+004)	**3.489317e+004** (sd:3.934237e+003)	1.699244e+007 (sd:2.839699e+006)	9.004884e+005 (sd:2.574297e+005)
400	4.341353e+005 (sd:4.039150e+004)	**1.944655e+005** (sd:1.791380e+004)	1.193405e+008 (sd:1.879186e+007)	7.264158e+006 (sd:1.815895e+006)
600	1.038128e+006 (sd:1.047097e+005)	**5.088249e+005** (sd:4.861865e+004)	3.778038e+008 (sd:6.527239e+007)	2.437371e+007 (sd:5.998997e+006)
800	1.886925e+006 (sd:1.686144e+005)	**1.003089e+006** (sd:9.943659e+004)	8.222648e+008 (sd:1.420139e+008)	5.461155e+007 (sd:1.360876e+007)
1000	3.030751e+006 (sd:2.625794e+005)	**1.676365e+006** (sd:1.730802e+005)	1.640714e+009 (sd:3.053816e+008)	1.063401e+008 (sd:2.404315e+007)

Concerning the properties of the benchmark functions, it seems that for higher dimensions CO is not the best choice for non-separable unimodal functions that can not be optimized dimension by dimension. FIPS performed better on them, although in one of the two cases CO was at least the second best. But in all other cases CO seems to be a good choice. For lower dimensions CO was the best or second best for non-separable unimodal problems as well as for separable multimodal problems, which can be optimized dimension-wise.

The FIPS approach performed well on non-separable unimodal problems that can not easily be optimized dimension by dimension. Furthermore, FIPS won the case of non-separable multimodal problems that can be optimized dimension-wise. FIPS is also the second best in half of all cases for lower and higher dimensions.

Low dimensional unimodal problems that can be optimized dimension-wise were best solved by the DE algorithm. It also won two of the multimodal problems, whereby the other properties differ too much so nothing can be drawn from this.

6 Conclusions

In this paper the Community Optimization algorithm was introduced, which models and simulates the behavior of members of a collaborative community to emerge their collective intelligence. As presented in section 5.9, CO is the winner in 9 cases and FIPS as well as DE in 4 cases. Furthermore, CO is the winner of 6 of the 8 high dimensional problems. Thus, CO seems to be a suitable approach for high dimensional problems. This could make CO an interesting optimizer for problem domains, which typically suffer from high dimensional representations. The following are a few examples of domains that could possibly benefit from CO:

Neural Networks:
Typically a neural network is trained by learning its weights, which are encoded in a sequence for all neurons and hence easily reach hundreds of dimensions, depending on the topology used.

Data Clustering:
Learning the centroids of a given data set is often realized by encoding them as a sequence, too. For data sets with a higher number of clusters in a high dimensional attribute space, this can lead to a high dimensional optimization problem.

Polynomial Approximation:
In regression problems, which use a polynomial as model, the number of coefficients can easily become high. Since they are typically encoded directly, this leads to a high dimensional optimization problem, too.

Implicit Data Structures:

In applications, where complex data structures are encoded as a vector, a high dimensionality can be easily reached. An example could be to optimize a matrix of weights, which is encoded as a vector containing all rows in a sequence. In case of a quadratic matrix, the dimension of the vector would grow exponentially. Another example could be to encode a static full tree as a vector as done in [21]. Each node of that tree needs at least one dimension of the vector. With an increase of the depth and arity of the tree, the dimensionality would grow exponentially, too.

Interestingly, the CO approach outperformed FIPS, although both can be interpreted as a model of human behavior. FIPS as a PSO-variant uses a direct communication between its swarm mates, whereas in CO stigmergy is employed. This raises the question, whether high-dimensional problems are better solved without direct communication. Of course, an answer can not be given only based on eight benchmark functions as used in this work. Further studies are needed to work out the differences between both approaches and their influences on the performance.

The DE algorithm and its self-adaptive version SaDE did not perform too well for higher dimensions. Both reached only 4 second places and were outperformed by CO and FIPS. But for low dimensional problems DE was slightly better than CO. Thus, one can recommend to use first DE and then CO for low dimensional problems and first CO and then FIPS for high dimensional problems.

Finally, the presented CO approach is not just an optimizer. In fact, it is a model based on the human behavior within a collaborative community (behaviors **B1 - B4**, page 2). CO could be used to verify the effectiveness of mechanisms within web communities, before implementing them into the appropriate collective intelligence system.

Acknowledgement. The author's implementation of CO can be downloaded as *open source* module written in *ANSI-C* under the terms of *GNU GPL 3*:

`http://commopt.sourceforge.net`
`http://sourceforge.net/projects/commopt`

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems. Oxford Press (1999)
2. Clerc, M., Kennedy, J.: The particle swarm Explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
3. Gardner, M., McNabb, A., Seppi, K.: A speculative approach to parallelization in particle swarm optimization. Swarm Intelligence, 1–40 (2011), doi:10.1007/s11721-011-0066-8; Published online first: December 21, 2011

4. Giles, J.: Internet encyclopaedias go head to head. Nature 438, 900–901 (2005), doi:10.1038/438900a
5. Herrera, F., Lozano, M., Molina, D.: Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, `http://sci2s.ugr.es/eamhco/updated-functions1-19.pdf` (retrieved on: February 8, 2012)
6. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, Perth, Australia. IEEE Service Center, Piscataway (1995)
7. Kennedy, J.: The particle swarm: social adaptation of knowledge. In: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, pp. 303–308 (1997), doi:10.1109/ICEC.1997.592326
8. Kennedy, J.: Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In: Proceedings of the 1999 IEEE International Conference on Evolutionary Computation, vol. 3, pp. 1931–1938 (1999)
9. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers (2001) ISBN: 1-55860-595-9
10. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. Proceedings of the 2002 Congress on Evolutionary Computation 2, 1671–1676 (2002)
11. Linux, Linux web site, `http://www.linux.org` (retrieved on June 3, 2013)
12. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: Simpler, maybe better. IEEE Transactions on Evolutionary Computation 8(3), 204–210 (2004)
13. Monson, C.K., Seppi, K.D.: Exposing origin-seeking bias in PSO. In: Proc. of the 2005 Conference on Genetic and Evolutionary Computation (GECCO 2005), pp. 241–248. ACM, New York (2005), doi:10.1145/1068009.1068045
14. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. Proceedings of the 2005 IEEE Congress on Evolutionary Computation 2, 1785–1791 (2005)
15. Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. Computer Graphics 21, 25–33 (1987)
16. Shi, Y.H., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: IEEE International Conference on Evolutionary Computation, Anchorage, Alaska (1998)
17. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, Univ. California, Berkeley, ICSI, Technical Report TR-95-012 (March 1995), Downloadable from `ftp://ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.pdf`
18. Storn, R.: On the Usage of Differential Evolution for Function Optimization. In: 1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS 1996), Berkeley, pp. 519–523. IEEE, USA (1996)
19. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341–359 (1997)
20. Sutton, A.M., Lunacek, M., Whitley, L.D.: Differential Evolution and Non-separability: Using selective pressure to focus search. In: Proc. of the 2007 Conference on Genetic and Evolutionary Computation (GECCO 2007), pp. 1428–1435. ACM, New York (2007) ISBN:1-59593-697-4
21. Veenhuis, C.B.: Tree Based Differential Evolution. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 208–219. Springer, Heidelberg (2009)

22. Veenhuis, C.: Community Optimization: Function Optimization by a Simulated Web Community. In: Proc. of the 12th International Conference on Intelligent Systems Design and Applications (ISDA 2012), Kochi, India, November 27-November 29, pp. 508–514. IEEE (2012); IEEE Catalog Number: CFP12394-CDR, ISBN: 978-1-4673-5118-8
23. Wikipedia, Wikipedia main page, `http://en.wikipedia.org/wiki/Main_Page` (retrieved on June 3, 2013)
24. Wiktionary Edit War, Wiktionary: edit war, `http://en.wiktionary.org/wiki/edit_war` (retrieved on June 3, 2013)

Diffusion of Innovation Simulation Using an Evolutionary Algorithm

Luciano Sampaio[3], João Varajão[1,3], E.J. Solteiro Pires[2,3], and P.B. de Moura Oliveira[2,3]

[1] ALGORITMI
[2] INESC Technology and Science (formerly INESC Porto, UTAD pole)
[3] Escola de Ciências e Tecnologia
Universidade de Trás-os-Montes e Alto Douro
5000–811 Vila Real, Portugal
luciano.sampaio@optimus.clix.pt,
{jvarajao,epires,oliveira}@utad.pt
http://www.utad.pt

Abstract. The diffusion of innovation theory aims to explain how new ideas and practices are disseminated among social system members. A significant number of the existing models is based on the use of parameters which determine the process of innovation adoption, and rely on simple mathematical functions centered in the observation and description of diffusion patterns. These models enable a more explicit diffusion process study, but their use involves the estimation of diffusion coefficients, usually obtained from historical data or chronological series. This raises some application problems in contexts where there is no data or the data is insufficient. This paper proposes the use of evolutionary computation as an alternative approach for the simulation of innovation diffusion within organizations. To overcome some of the problems inherent to existing models an evolutionary algorithm is proposed based on a probabilistic approach. The results of the simulations that were done to validate the algorithm revealed to be very promissing in this context.

Simulation experiment results are presented that reveals a very promising approach of the proposed model.

Keywords: Diffusion, inovation, simulation, evolutionary computation.

1 Introduction

The diffusion of innovation (DOI) theory aims to explain how new ideas disseminate within and among members of a social system [1–10]. Diffusion of innovation, corresponds to the process whereby one innovation is communicated through certain channels to the members of a social system along time [4, 6–10].

In last decades several models have been proposed, each one with different sets of factors determinant to the process of innovation adoption. In general, the focus of these models is the individual and/or the organization within a social system. Some good examples are the models of Rogers [6], Moore and Benbasat

[11], Venkatesh et al. [12] and Davis et al. [13]. Other models were proposed to explain the DOI and to measure artifacts, diseases and the way some behaviors are spread in adjacent areas [3].

Significant research efforts about the diffusion process are based on simple mathematical functions centered in the observation and description of diffusion patterns, and become relevant in the adoption phenomena. The resulting models allow estimating the diffusion velocity, the innovation rates and mimicking rates. Some examples are the logistic macro-models of Gompertz [2] and Bass [1–3, 6, 14]. Recently this topic research has gained more strength with the development of more sophisticated network models and technologies, allowing a more explicit study of the diffusion process [3].

The application of the previously mentioned functions in diffusion models results in 'S' shape or sigmoid curves. However, frequently it is not possible to determine empirically which function better represents the diffusion curve, as any uni-modal distribution function generates a 'S' shape [2]. The use of these models requires the estimation of diffusion coefficients. These are usually estimated from historical data or chronological series. This could be a problem in the absence of data or in case of insufficient data. Alternately, mean values of certain characteristics from related innovations or the expert evaluation are used [2], which does not favor much the estimation process.

Although there is a large number of articles in literature about DOI, only a few were found using techniques based on evolutionary computation. O'Mahoney [15] applies memetics algorithms to explain management innovation diffusion as an evolutionary process. The work draws on qualitative evidence from two case-studies of business process re-engineering implementation to illustrate that the replication, selection and variation of management innovations can form evolutionary algorithms (memes) which support diffusion processes. This clarifies the ways in which innovations contribute to their own replication and explains how the high failure rates associated with business process re-engineering can sometimes improve its chances of reproduction. Alkemade and Castaldi [16] studied the innovations spread on a social network. The network consists of agents that are exposed to the introduction of new products. Consumers decide whether or not to buy a product based on their own preferences and influenced by their neighbors. The aim of the work is to find out a directed-advertising firm strategy from the network structure and consumer characteristics. A genetic algorithm is used in [16] to model the strategy search and learning behavior of the firm.

This paper presents new results of a research work started in [17, 18], that applies evolutionary computation in the context of DOI simulation in organizations, aiming to overcome some of the problems of the current existing models. The remaining of the paper is organized as follows: section 2 presents the background DOI theory principles; section 3 presents an evolutionary algorithm based on a probabilistic approach; section 4 presents simulation results, and finally some conclusions are outlined.

2 Models for Diffusion of Innovation

The DOI concept was first studied by the French sociologist Gabriel Tarde (1843-1904), in its work entitled 'Les lois de limitation'. Since then, mainly from the decade of 1940, several results have been published in this subject. Rogers [6], in its 4th edition of 'Diffusion of Innovations', accounted for 3890 works about this topic, distributed in eleven classical research areas, such as: anthropology; sociology; education sciences; economical sciences. In 2004, about 5000 research works were performed in a wide range of disciplinary areas [8].

Most of the first research projects about DOI, using simple time domain mathematical functions are based both on the observation and description of diffusion patterns in terms of pre-defined tendency or distribution functions. Examples of the former are: the logistic from Gompertz, the accumulated normal distribution and the Bass distribution. These models are generally used with diffusion models as they generate 'S' curves [2], and differ in terms of the number of parameters and in the type of influence conducted in the diffusion process. The first models assume that the diffusion process is totally conducted by pure imitation, with the diffusion performed solely by interpersonal contacts. The Bass model is assumed as a mixed model, taking into account external influence factors as well as imitation factors. The external influence factors represent the decision process intrinsic tendency to adopt, or not, a certain innovation, as well as the external factors effect. Examples of the former are: changing agents influence; mass-media; governmental agencies; marketing agencies; among others [2].

Accordingly to Mahajan and Peterson [2], conceptually it is possible to consider the communication channels effect, which can be of the following type: vertical; centralized; structured or formal. Accordingly to Young [14], innovation is diffused through two channels: from the fonts internal to the group and/or from the fonts external to the group. The intensity of these sources determine the diffusion curve shape. The diffusion patterns of these models can be characterized in function of two mathematical properties [2]: the symmetry of the adoption rate curve and the inflection point location relatively to the adopters accumulation.

Rogers model [6] was selected as the conceptual support of the current work, as: it is one of the most referenced theories in research related to innovation diffusion; and mainly as it is a general model, it can be applied to a wide range of real world problems. This generality results, accordingly to Rogers [8], from the consistent patterns and regular aspects in a large variety of conditions, innovations and cultures in the innovation diffusion process. Moreover, there is a certain mathematical approximation of this model to others models. This approximation is performed by 'S' adoption rate curve (which is the accumulated frequencies graphical representation of the number of individuals which adopt one innovation through time, similar to the Gauss curve or normal accumulated distribution). This represents the influence of a set of internal and external factors, expressed in dependent and independent variables, in the adoption and diffusion process.

Rogers [6] states that innovation is ‘an idea, practice or object that is perceived as new by an individual or other unit of adoption’, and ‘the diffusion of an innovation is defined as ‘the process by which an innovation is communicated through certain channels over time among members of a social system’. The social system is ‘a set of interrelated units that are engaged in joint problem-solving to accomplish a common goal’. The social system members or units can be individuals, informal groups, organizations and/or subsystems. From these definitions, and from the Rogers model, a set of independent variables is identified, which explains the innovation and diffusion process and respective adoption rate: innovation attributes; innovation decision type; communication channel type; social system nature; members of social system (adopters type), extent of change agents promotion efforts; and time. Nevertheless, the experience variable *vs.* knowledge in relation to the innovations is not directly part of this set, it is referred in Rogers work [6] in terms of ‘past experiences’.

In the next subsections a brief description for this set of variables is presented, as they are key elements in the context of this work.

2.1 Innovation Attributes

The innovation attributes are the characteristics of one innovation perceived by individuals, which help to explain the different adoption rates [6]: relative advantage – it is the perception degree regarding one innovation, of how better the innovation is than the associated idea itself; compatibility – it is the perception degree of an innovation as being consistent with the existing values, past experiences, the needs of potential adopters; complexity – it is the innovation perception degree of how difficult it is to understand and use; trialability – the degree of experimentation associated with one innovation in a limited base; observability – it is the degree of visibility of one innovation results for others. Innovations considered as having a high level of relative advantage, compatibility, trialability, perceptibility and low complexity, can be adopted faster than others.

2.2 Innovation-Decision Type

The decision making is a social process by which social behaviors are frequently conducted by the social influences and consequences. The social context also plays a global role in the decision-making processes [19]. Rogers [6] identifies three types of decisions: optional – the decision of accepting or rejecting one innovation is taken independently from other social systems members decision: this is a personal decision; collective – the choices are taken in consensus among social system members; authority – the decision is taken by some individuals within the social system which have authority, status or knowledge in the matter (the other members only implement the decision).

2.3 Communication Channel Type

A communication channel is the medium whereby messages are transmitted from one individual to another [6]. For Mahajan and Peterson [2] channels of communication are mediums by which information is transmitted to or within a social system. Two communication channels type can be identified: *mass-media* and interpersonal communication channels. The first one involves the mass media transmission information which includes radio, television, newspapers, magazines, internet and many others. On the other hand, interpersonal communication channels involve information exchange directly between two or more individuals.

2.4 Social System Nature

A social system, according to Rogers [6], is 'a set of interrelated units that are engaged in joint problem-solving to accomplish a common goal'. Moreover, it states that members or units of the social system may be individuals, informal groups, organizations and/or subsystems. In the same sense, Mahajan and Peterson [2] considers a social system comprised by individuals, organizations, or agencies that share a common 'culture', being potential adopters of an innovation. All members of the social system cooperate between them to solve a common problem in order to achieve a common goal.

2.5 Type of Adopters

Based on adoption time and innovativeness degree (innovation predisposition) of adopters, *i.e.*, the degree in which an individual, or another unit, adopt new ideas earlier than other members of the system, Rogers proposes five adopters types [6]: innovators (risk takers), early adopters (hedgers), early majority (waiters), late majority (skeptics), laggards (slowpokes).

2.6 Extent of Change Agents Promotion Efforts

A change agent is an individual who influences the decision-innovation process towards the desired direction of the individual or organization responsible for driving the innovation. The innovation adoption rate is affected by the promotional efforts extent of change agents. However, this relationship is not direct neither linear, as it depends on certain innovation diffusion stages. Indeed, change agents obtain the greater recognition of the promotion efforts when opinion leaders adopt. An opinion leader is an individual who has the ability, in an informal way, to influence the other individuals attitudes or change their behavior in the sense that he intends to [6].

2.7 Time

The third element underlying Rogers model [6] is the time involved in the innovation-decision process, in innovativeness degree and adoption rate. In the innovation-decision process the individual pursues through several stages. In innovativeness degree certain individuals or adoption units accept an innovation earlier compared to other members of a social system. The adoption rate refers to the relative velocity which an innovation is adopted by the members of a social system.

2.8 Adoption Rate

The adoption rate is the relative adoption speed, per time, of an innovation by the system members or social unit. Usually, it is measured by the system members number that adopted the innovation in a given time period. This rate becomes the indicator of innovation acceptability and success in a system [6, 7].

2.9 Acquired Experience

The individual or collective learning or experience is a factor to take into account in the innovation diffusion. This factor cuts across several models [6, 13]. When an individual or group learn to execute a particular activity, the knowledge of this activity will grow cumulatively. Begins with an initial knowledge and increases, over time, until mastering the activity, reaching therefore the depletion of the natural level of learning.

3 Social and Personal Networks

Social and personal networks concepts [3, 6, 7] are fundamental in this work. For Rogers [6], a social network is the set of individuals that are connected by standardized information flows, whereas a personal area network consists on a set of individuals that are connected by standardized flow communication to an individual. Rogers defines two types of personal networks:

– Interlocking personal network – consists of a set of individuals in which all interact with all;
– Radial personal networks – consist of a set of individuals connected to an central individual, the radial individuals do not interact with each other.

Radial networks are more open to an individual environment and are more important in the DOI [6]. The concept of networking and interpersonal closeness between pairs of individuals was addressed by Rogers [6, 7] as factors that may facilitate or prevent innovation adoption in a social system, depending on its nature, regarding particularly the communication structure. The proximity between pairs of individuals, integration, degree, radiality, inter-connectivity and eigenvector have been used as centrality measures in social network studies based on distance, providing indicators of network structure and how structure influences individual behavior. These measures are important to determine who occupies a relevant position in the network [20, 21].

Recently Valente [22] proposed a bridging measure for the study of network structure, dynamic network and network effects on the individuals behavior. The technique consists in removing links instead of central nodes, and then aggregating changes to the node. Detect connections between individuals and key individuals that increase the cohesion may be an alternative and more efficient to accelerate change or improve network performance.

The structure of social networks, as mentioned before, may influence innovation adoption, as well as behavior change [5, 6, 9, 20, 23–25]. Thus, DOI occurs in complex systems where the networks that connect individuals to a system are overlapping, multiple and complex [9].

The idea that the DOI is supported by a network of social influences is used by Valente [5] in his threshold model of social networks broadcast. This model draws on the four categories of adopters proposed by Ryan and Gross [26], in order to distinguish the collective behavior of adopters in the face of innovation, namely: early adopters; early majority; late majority; and laggards. This categorization, based on innovativeness as a measure of adoption time, is performed taking into account the networking concepts of social, exposure and threshold. Specifically, for each category an adoption threshold is created, following the same apportionment method used by Ryan and Gross [26] as for the adoption time. Thus, individuals with very low and low threshold network have a threshold of personnel network below $\mu - \sigma$ and in the range $[\mu - \sigma, \mu]$, respectively, where σ and μ are the standard deviation and average threshold, respectively. On the other hand, individuals with high threshold network have the threshold of network personnel value in the interval $[\mu, \mu + \sigma]$. Finally, individuals with very high threshold have its value above $\mu + \sigma$. The average value is the threshold mean of the social system adopters [5]. In this sense, adopters with a lower threshold value are individuals who adopt sooner when compared to other elements of their personal network. Furthermore, individuals with high threshold value tend to adopt an innovation much later. According to the four categories of adopters proposed by Ryan and Gross [26], these two types correspond to early and laggards adopters.

Exposure and threshold concepts assume an important role in this work. Rogers proposed a definition of threshold [6, p. 333–334]. In this context important concepts are:

– Exposure in a network: as the neighbors (personal network) adopters of an individual at a given moment. The exposure generally increases as time passes and more individuals adopt the innovation in the social system, raising the possibility of an individual to adopt;
– Adoption threshold: as the exposure at adoption time which varies from individual to individual [3, 5].

Figure 1 illustrates the exposure of a personal network with respect to an innovation. The network consists of five individuals and a weight based relationship between the central individual and its neighbors [5, 27].

In Figure 1, the personnel network (a), taking as reference the central individual as a potential adopter, indicates that when time $t = 1$ exposure is nil; in

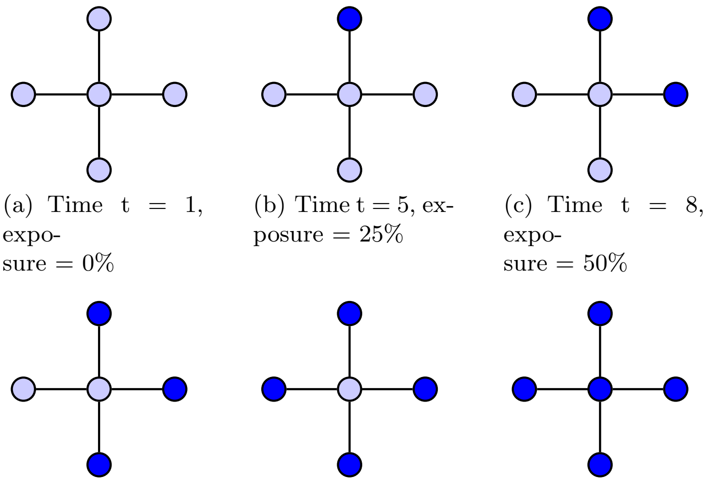

(a) Time t = 1, exposure = 0%
(b) Time t = 5, exposure = 25%
(c) Time t = 8, exposure = 50%
(d) Time t = 9, exposure = 75%
(e) Time t = 11, exposure = 100%
(f) Time t = 15, exposure = 100%

Fig. 1. Exposure degree of a Innovation in a personal network. Adapted from Coleman *et al.* [27] and Valente [5].

plot (b) the individual has an exposure degree of 25%, *i.e.* 1/4 (1 adopter by the total individual number of the personal network consisting of 4 neighbors) at $t = 5$. In network (d) the exposure degree, for the $t = 9$, is 75%. Finally, in plot (f) the individual adopts at time $t = 15$, when all its network partners have already adopted, and the individual threshold exposure at that time is 100%. In this example, the central individual adopted when all partners had already adopted, but since the threshold varies from individual to individual, another potential adopter could adopt, for instance, when t=9 with a 75% exposure. In the example presented in figure 1, the network could be formed by two friends, one discussion partner and one counseling partner.

In this study, the weight or intensity of relations with the central element is random, following a probabilistic uniform distribution $\mathrm{U}(0, 1)$, thus varying the individual exposure degree and threshold. The relationship intensity, of the central individual, is evaluated from the influence of its peers and/or its receptiveness to innovation. The simulated DOI is supported by a social network and, in particular, the social influence process is accomplished by direct contact from individual to individual. In harmony with this idea, Young [28] formalized a relational model based on a graph of particular interest for the present work. According to Young, a social network can be represented by a graph G constituted by a finite set of nodes V and a set of non-oriented arcs. Each node i is an individual of the system. An arc $\{i, j\}$ connects the individuals i and j, if and only if i and j are neighbors. This means that actions are the outcome of mutual behavior. Each link $\{i, j\}$ is supposedly endowed with an intensity $w_{ji} = w_{ij}, w_{ij} > 0$. If $w_{ij} > w_{ik}$ means that individual i gives more importance to individual j actions than to individual k actions [28]. The model also uses the stochastic function proposed by Blume [29], equated from game theory, to represent the individuals decision-making. Individuals may opt for two different types of behavior, A or B, for the perceived usefulness. Thus, the formalization of Young, Valente and Rogers models concept, is an important support to the work presented here.

4 Proposal of Innovation a New Evolutionary Algorithm for the Diffusion of Innovation Simulation

Several diffusion models, including the aforementioned, follows the well known ‘S’–curve [2]. The application of such models involves the estimation of coefficient diffusion parameters. These parameters are usually estimated from historical data or time series. However, in some cases the historical data is not available or the phenomena is not repeatable. To overcome this problem, the historical estimation is often replaced by the average parameter values of other similar innovations or is based on expert evaluation. For instance, in the Bass model, the estimation of these parameters requires a time series having, at least, the number of adopters in three different moments [2]. The Bass model, as a mixed model, is based in three essential parameters [1, 2] and, as it have been described, is not always possible to apply since this process depends on the innovation age, particularly if it is recent or not. However, other techniques, as for example linear or non-linear regressions, can be used if historical data do exists. Because of this, models based on the logistic equation are of restricted applicability. Moreover, these macro-models become inaccurate by admitting a perfect social mixing, where everyone interact with everyone, and neglect similar behaviors where people are linked together [3]. This structural dependence, which is reflected in the parameters estimation, gives rise to several problems related to DOI simulation and has motivated the present work: taking into account the trends in a population adoption for a particular innovation. Evolutionary computation is proposed to simulate the evolutionary process of innovation diffusion here.

The general principle of an evolutionary algorithm (EA) is quite simple. A set of N individuals randomly generated, form the initial population. Each individual has a certain performance, which represents the degree of individual’s adaptation (fitness value) to its environment. The purpose of an EA is progressively improving, generation after generation, the performance of each individual using main evolution mechanisms [30], such as: selection, which favors the reproduction process and thus survival of the most adapted individuals to their environment; reproduction, which allows recombination (or crossover) of parents hereditary characteristics and mutation at the genetic heritage of the individual, to create offspring with new skills.

Each population, $P(t) = \{x_1^t, \ldots, x_n^t\}$, individual x_i^t represents a potential solution, for a certain problem in iteration t, is subjected to genetic operators and evaluated using an objective function, allowing to establish its relative merit in relation to other population individuals. The application of genetic operators results in a new population in iteration $t+1$. The EA is executed for a pre-defined number of generations [31, 32] or until a convergence threshold is achieved.

The contextual relationships between concepts, models and the evolutionary basis of innovation diffusion, and the principles of EA, allowed to create an algorithm structure, presented in algorithm 1 in order to meet the objectives of this work, with the following elements:

1. A finite population consisting of N individuals, each one with three genes, organized in a matrix $\mathbf{P}$ $(N \times 3)$. The genes represent: the degree of knowledge acquired by the individual, the degree of exposure of the individual and the adoption (identifies whether an individual takes or not the innovation);
2. A $\mathbf{R}$ $(n \times m)$ matrix that represents a regular graph (social network), of type torus (or ring), wherein each array element identifies a network node, *i.e.*, an individual i. Each individual i can be influenced by a maximum of four neighbors, which corresponds to a maximum exposure of 100%;
3. The fitness function, that measures the individual's ability, is determined from the influence of the closest peers and of the innovation degree knowledge;
4. The improvement of the individual: the individual is subjected to various types of changes over the generations in order to improve it;
5. A 'selection' mechanism: evaluates, from the fitness function, the individual's propensity for being an adopter.

The algorithm matrix size, the social network, can be configured in terms of the number of rows and columns. In this work a 4×100, 20×20 and 40×10 matrix sizes are used.

For the proposed model experimentation, a social network of interactions was defined with a constant neighborhood (4 individuals) represented as a ring grid. This network is represented by an matrix $\mathbf{R}(n \times m)$ with every element identifying a grid node, and this node represents an individual. Population individuals are organized in another matrix $\mathbf{P}(n \times 3)$. The network matrix is initialized sequentially, with the line index (i) (individual position) of the population matrix (see Figure 2), *i.e.*, the index of the first individual $(i = 1)$ from $\mathbf{P}$ is placed in the first element of $\mathbf{R}(i \to r_{1,1})$, the second individual $(i = 2)$ is placed in the second element $(i \to r_{2,1})$, and so on, column by column, until the last individual $(i = n)$, occupies the last element of $\mathbf{R}(i \to r_{n,m})$. As in the ring typology all individual are interconnected, alternatively to another process, two columns and two lines are added to the matrix $\mathbf{R}$: one column to left of the first column and the other to the right of the last column; one line above the first line and another bellow the last line. In this way the dimension of matrix $\mathbf{R}$ was changed to $\mathbf{R}(n + 2 \times m + 2)$. The individuals from the first column are duplicated in the new right column and vice-versa, *i.e.*, the last column elements are duplicated

```
Data: Logit coeficients
/* Individuals initialization of matriz $\mathbf{P}(n \times 3)$ */
$p_{i,1} \leftarrow 0; p_{i,2} \leftarrow 0; p_{i,3} \leftarrow 0;$
/* Inicialization of network $\mathbf{R}(n \times m)$ from $\mathbf{P}(n \times 3)$ (insert
   individuals into the network) */
/* Random inicialization of early initial adopters $(x)$ */
$p_{i,1} \leftarrow 1; p_{i,2} \leftarrow 1; p_{i,3} \leftarrow 1;$
while number of adopters $(a) < N$ do
    initialize random sequence of $N$ individuals to be changed;
    foreach Individual $i$ do
        if $U(0,1) < 0.001$ then
            Mutate adoption gene
        end
        change knowledge degree gene;
        change exposure degree gene;
        evaluate fitness if $p_m > 0.5$ then
            $p_{i,3} \leftarrow 1$
        end
    end
    $a \leftarrow$ number of neighbors adopters;
    $t \leftarrow t + 1;$
    save the number of adoptants;
end
```

Algorithm 1. Proposed algorithm

in the new left column. The same procedure is applied to the new lines, closing the ring network.

The innovators (early adopters) are randomly generated, following a uniform distribution ($X \sim U(0,1)$), at the beginning of the algorithm. All individuals genes are initialized with the value 1, indicated in this way that have already adopted the innovation.

During the evolution, in each generation, the population individual genes (known as *exposure* and *knowledge* degree) are updated following a random uniform probability distribution. Therefore, the *exposure degree* gene is modified according to both the adopters number in the neighborhood and the connection weight between the individual and its neighbors. This update is cumulative, *i.e.*, the *exposure* degree acquired by the individual in a previous generation, is added with the *exposure* degree evaluated in the current generation. The *knowledge degree* gene across generations memorizes the level of knowledge acquired by the individual (is a real number that varies between 0 and 1). This gene is updated from the logistic function of Verhulst [33, 34], often used to evaluate the knowledge evolution of a person.

When the probability exceeds 0.5, the individual i has a strong adoption likelihood. Therefore, the *adoption* gene takes the value 1.

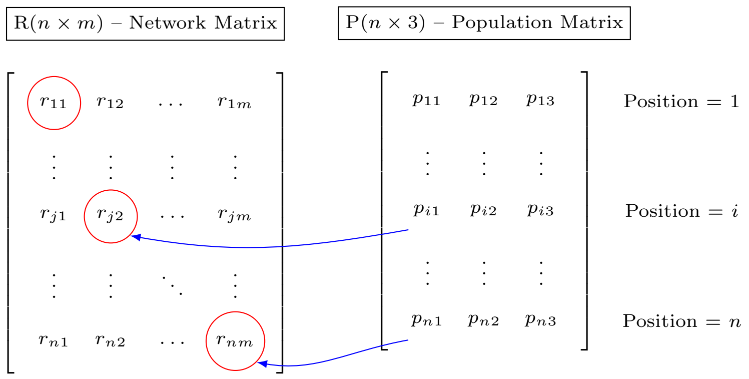

R($n \times m$) – Network Matrix
P($n \times 3$) – Population Matrix
r_{11} r_{12} … r_{1m}
r_{j1} r_{j2} … r_{jm}
r_{n1} r_{n2} … r_{nm}
p_{11} p_{12} p_{13}
p_{i1} p_{i2} p_{i3}
p_{n1} p_{n2} p_{n3}
Position = 1
Position = i
Position = n

Fig. 2. Relation between the network matrix and population matrix

The individuals evaluation is based on the binary regression model. If the adjusted *logit* proves significant and high quality as well as useful for the classification of new individuals, it is used in the respective simulation.

5 Simulation Results

Twenty independent simulations were performed for each used network sizes of 4×100, 20×20 and 40×10. For each case 5% and 10% initial adopters was considered. Each subnetwork, or individual network, is formed by 5 individuals in a radial topology connection, where all the participants have the same centrality degree. This means that, each individual can be influenced, at most, by the 4 neighbors. When this influence occurs the exposure is maximum.

The network was constructed in order to model the contact and influence flow according to the physical space available or according to organizational methods. However, other network size could be adopted in order to represent other social situations.

According to Bass [1], Mahajan and Peterson [2], Rogers [6], Valente [3] and Young [14] the diffusion curves has a ‘S’ shape. The diffusion obtained curve seems to follow the logistic, Gompertz and Bass models as can be seen by figures 3, 4 and 5. Therefore, the non-linear regression was used to estimate the curves using the R-statistical software. The regression curves of the three models reveals adjustements with high quality in the simulations performed. The R^2 coefficients varies in the range $[0.8099, 0.9972]$, $[0.8916, 0.9963]$ and $[0.9643, 0.9994]$ for logistic, Gompertz and Bass models, respectively. Figures 6, 7 and 8 show the adjustment of the curve for the experiments.

All the experiments performed, for each model, were considered since they present high adjustment quality. Although, in most cases, the assumption of non-linear regression for independence and random error was not verified. A fact which is attributed to the high regularity inherent in the process of computer

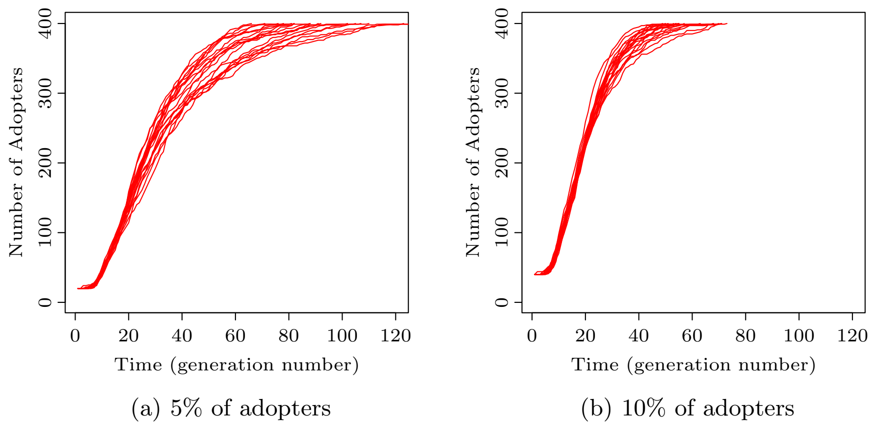


Fig. 3. Adoption evolution curves of 20 independent simulations in a 4×100 network size, for 5% and 10% of initial adopters

simulation. However, these models have created a more realistic picture of the innovation diffusion, regardless of the innovation itself.

6 Conclusions

The currently existing DOI models require data that in some cases is unavailable or does not exist. Therefore, it is pertinent to propose new models that allow, in dataless situations, to perform innovation diffusion simulations with rigor.

In this context, a new evolutionary algorithm it was proposed based on a probabilistic approach and conceptually based on the social system adoption trends, when the system is exposed to innovation phenomena. The proposed model describes the system in macro terms, at organizational and social levels. Diffusion is supported, on one hand, by social networks of influences where the influence process is carried out by direct contact between individuals in personal networks. On the other hand, it is supported in the social system trends at micro level (individual), based on the exposure level, the individual characteristics evolution and on the factors that influence decision making.

The main advantage of the proposed model is not requiring, unlike the aforementioned macro-models, to estimate the diffusion coefficients that materialize the innovation transmission rate, the internal and external influences to describe the innovation diffusion, since this is based on evolutionary trends of social system adoption. This model is particularly interesting since it shows, from the estimation of regression coefficients, the first generations adoption trend in each new simulation. These early trends (coefficients set) will determine the adoption rate or pace of a social system.

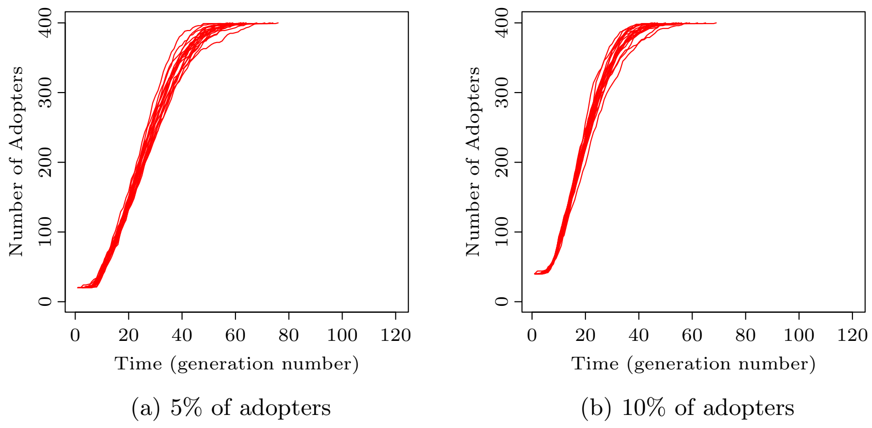


Fig. 4. Adoption evolution curves of 20 independent simulations in a 20 × 20 network size, for 5% and 10% of initial adopters

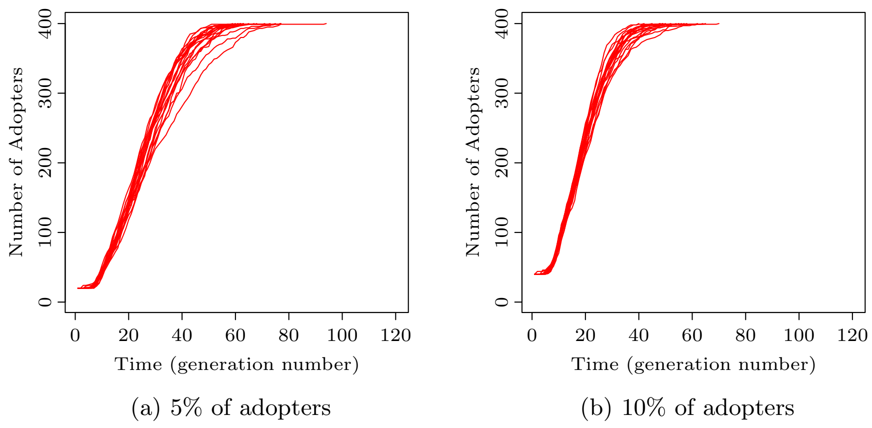
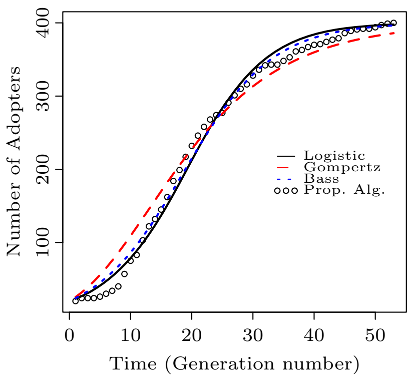
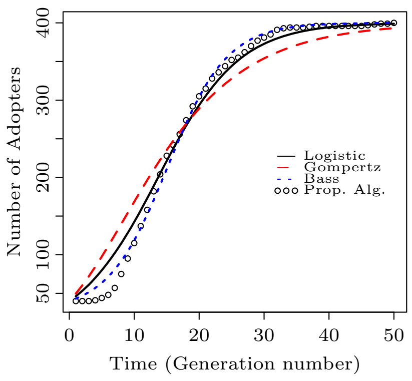


Fig. 5. Adoption evolution curves of 20 independent simulations in a 40 × 10 network size, for 5% and 10% of initial adopters

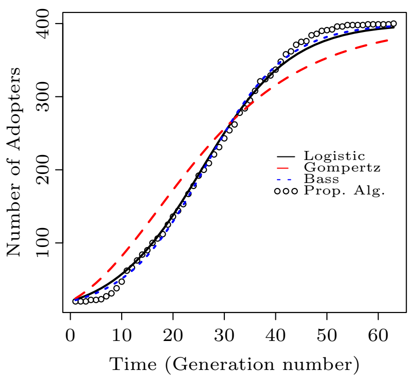


(a) 5% of adopters

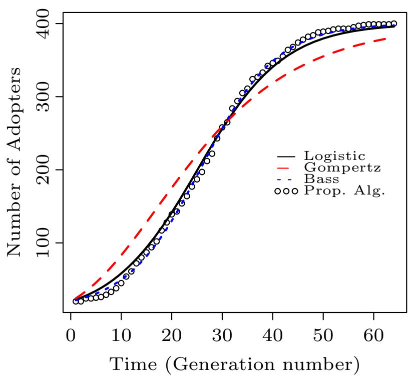


(b) 10% of adopters

Fig. 6. Adjustment quality of models: logistic, Gompertz and Bass in the model for a simulation with 5% and 10% of initial adopters in the network 4×100

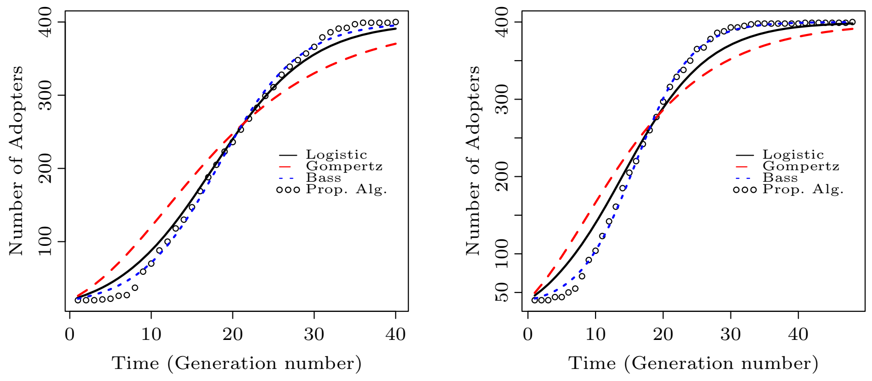


(a) 5% of adopters



(b) 10% of adopters

Fig. 7. Adjustment quality of models: logistic, Gompertz and Bass in the model for a simulation with 5% and 10% of initial adopters in the network 20×20



(a) 5% of adopters (b) 10% of adopters

Fig. 8. Adjustment quality of models: logistic, Gompertz and Bass in the model for a simulation with 5% and 10% of initial adopters in the network 40 × 10

7 Future Work

In future, considering the applicability of the theoretical probabilistic model above mentioned and obtained: the logit and the respective explanatory variables that contribute to explain the adoption; the mathematical models inherent in the adoption trends of each variable (in this case the knowledge follows a logistic function); and the network type and the networks for each individual and its interpersonal influence weights (the relationship between the individual and their partners) a real DOI simulation could be performed.

The algorithm, based on these foundations, could simulate the DOI evolution in a social system, be it a set of businesses, schools, municipalities, etc.

Acknowledgment. This work is part financed by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project "FCOMP – 01–0124–FEDER–022701".

References

1. Bass, F.M.: A new product growth for model consumer durables. Management Science 15, 215–227 (1969)
2. Mahajan, V., Peterson, R.A.: Models for innovation difusion, vol. 48. Sage publications, Newbury Park (1985)
3. Valente, T.W.: Network Models and Methods for Studying the Diffusion of Innovations, pp. 98–116. Cambridge University Press, New York (2005)

4. Rogers, E.M.: Diffusion of Innovations, 3rd edn. Free Press, New York (1983)
5. Valente, T.W.: Social network thresholds in the diffusion of innovations. Social Networks, 69–89 (1996)
6. Rogers, E.M.: Diffusion of Innovations, 4th edn. Free Press, New York (1995)
7. Rogers, E.M.: Diffusion of Innovations, 5th edn. Free Press, New York (2003)
8. Rogers, E.M.: A prospective and retrospective look at the diffusion model. Journal of Health Communication (9), 13–19 (2004)
9. Rogers, E.M., Medina, U.E., Rivera, M.A., Wiley, C.J., Rogers, E.M., Medina, U.E., Rivera, M.A., Wiley, C.J.: Complex adaptive systems and the diffusion of innovation. The Innovation Journal: The Public Sector Innovation Journal 10(3), 2–26 (2005)
10. Rogers, E.M., Peterson, J.C.: Diffusion of clean indoor air ordinances in the southwestern united states. Health Education & Behavior (35), 683–697 (2008)
11. Moore, G., Benbasat, I.: Development of an instrument to measure to perceptions of adopting. Information Systems Research 2(3), 192–222 (1991)
12. Venkatesh, V., Morris, M.G., Davis, G.B., Davis, F.D.: User acceptance of information technology: Toward an unified view. MIS Quarterly 27(3), 425–478 (2003)
13. Davis, F.D., Bagozzi, R.P., Warshaw, P.R.: User acceptance of computer technology: A comparison of two theoretical models. Management Science 35(8), 982–1003 (1989)
14. Young, P.H.: Innovation diffusion in heterogeneous populations: Contagion, social influence, and social learning. American Economic Review 99(5), 1899–1924 (2009)
15. O'Mahoney, J.: The Diffusion of Management Innovations: The Possibilities and Limitations of Memetics. Journal of Management Studies 44(8), 1324–1348 (2007)
16. Alkemade, F., Castaldi, C.: Strategies for the diffusion of innovations on social networks. Comput. Econ. 25(1-2), 3–23 (2005)
17. Sampaio, L., Varajão, J., Solteiro Pires, E.J., de Moura Oliveira, P.B.: Innovation diffusion in organizations – an evolutionary approach. In: CENTERIS 2011 – Book of Abstracts, CENTERIS – Conference on Enterprise Information Systems, SciKA, p. 5 (2011)
18. Sampaio, L., Varajão, J., Solteiro Pires, E.J., de Moura Oliveira, P.B.: Diffusion of innovation in organizations: Simulation using evolutionary computation. In: 2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC), Mexico City, Mexico (2012)
19. Coronges, K., Stacy, A.W., Valente, T.W.: Social network influences of alcohol and marijuana cognitive associations. Addictive Behaviors (36), 1305–1308 (2011)
20. Valente, T.W., Coronges, K., Lakon, C., Costenbader, E.: How Correlated Are Network Centrality Measures? Connections (Toronto, Ont.) 28(1), 16–26 (2008)
21. Valente, T.W., Foreman, R.K.: Integration and radiality: Measuring the extent of an individuals connectedness and reachability in a network. Social Networks 20, 89–105 (1998)
22. Valente, T.W., Fujimoto, K.: Bridging: Locating critical connectors in a network. Social Networks 32, 212–220
23. Valente, T.W., Rogers, E.M.: The origins and development of the diffusion of innovations paradigm as an example of scientific growth. Science Communications 16(3), 242–273 (1995)
24. Valente, T.W., Davis, R.I.: Accellerating the diffusion of innovations using opinion leaders, the annals of the american. Academy of the Political and Social Sciences 566, 55–67 (1999)

25. Valente, T.W., Fujimoto, K., Palmer, P., Tanjasiri, S.P.: A network assessment of community-based participatory research: Linking communities and universities to reduce cancer disparities. American Journal of Public Health 100(7), 1319–1325 (2010)
26. Ryan, B., Gross, N.: The diffusion of hybrid seed corn in two iowa communities. Rural Sociology 8(1), 15–24 (1943)
27. Coleman, J.S., Katz, E., Menzel, H.: Medical innovation: A diffusion study. Bobbs-Merrill, New York (1966)
28. Young, P.H.: Diffusion in social networks. In: Working Paper (2), Brookings Institution (1999)
29. Blume, L.E.: The statistical mechanics of strategic interaction. Game and Economic Behavior 5, 387–424 (1993)
30. Darwin, C.: The Origin of Species. J.M. Dent and Dons Ltd. (1975)
31. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison – Wesley (1989)
32. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer (1996)
33. Verhulst, P.F.: Recherches mathématiques sur la loi d'accroissement de la population (mathematical researches into the law of population growth increase). Nouveaux Mmoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles 18, 1–42 (1845)
34. Verhulst, P.F.: Deuxième mémoire sur la loi d'accroissement de la population. Mémoires de l'Académie Royale des Sciences, des Lettres et des Beaux-Arts de Belgique 20, 1–32 (1847)

Driving Robots Using Emotions

Shashi Shekhar Jha, Shrinivasa Naika C.L., and Shivashankar B. Nair

Indian Institute of Technology Guwahati,
Department of Computer Science and Engineering
Guwahati, Assam, India
{j.shashi,shrinivasa,sbnair}@iitg.ernet.in

Abstract. Researchers have tried to embed synthetic emotions into robot much like their biological counterparts. While many have shown the effect of emotion on decision-making for the robot, the scenarios that portray when to use emotions for robots are rare. In this paper, we evaluate the performance of a robot by empowering it with a decision-making capability which uses synthetic emotions. Since from the robot's perspective the environment is stochastic, it needs to make the right decision for survival. A *Comfort* level is defined as a metric which determines the quality of life of the robot. The robot possesses various *needs* and *urges* all of which influence its decisions. The main objective was to make the robot perform high profile tasks rather than menial ones so as to increase its utility. Results obtained from experiments conducted using a real situated robot with and without emotion indicate that emotion aids more significantly when the environment has abundant resources.

Keywords: Artificial Life, Synthetic emotions, Behavior selection, Robotics, Software agents, Softbots, Decision-making.

1 Introduction

Researchers are still yet to unravel the secrets of the complex phenomenon of human emotions and its use. Theories suggest that emotions are caused due to possible bio-chemical changes within the living body which in turn can change the behavior and actions performed by the being [1]. Emotions possibly contribute to a survival strategy of the being. Both decision-making and learning have been known to be influenced by emotions [2]. Several attempts have been made to embed emotions into robots and machines [3]. In [4], authors have compared an emotion based decision-making methodology with the traditional approaches and shown that the use of emotions incur a lower computational cost for problem solving. Domenica *et al.* [5] have embedded emotion in the form of a neural network to aid robots in decision-making. Lee *et al.* [6] have used an emotion model to arbitrate predefined behaviors depending upon external stimuli. Liang *et al.* [7] have proposed a fuzzy logic based artificial emotion generation model (FLAEEM) for behavior selection by an agent to survive in dynamically changing environment. Kim *et al.* [8] have proposed a Cognitive appraisal theory based emotion generation model which uses a Partially Observable

Markov Decision Process (POMDP) to model actions of the robot. The model uses six predefined criterion which are pre-derived using POMDP from tasks that a robot is expected to execute. They have concluded that if a robot has apriori probabilistic information of state transitions and sensory observations then the deliberative emotion generation process can be applied to enhance natural human interactions. Park *et al.* [9] have put forward an emotion generation model which takes into account personality, mood and history of the emotion felt, using predefined rules for personality and mood. Emotion is generated and expressed using a simulated facial expression model based on FACS (Facial Action Coding System) [10]. Emotion also seems to play a significant role in speeding up the natural selection process [11]. Lv *et al.* [12] have used an emotion model derived from psychological model of the emotional content of music. The model is used to analyze the rhythm of music in dance performance. While many have worked towards rendering emotions using robotic faces [13–15] few have proposed practical and viable models for their actual generation [16, 17].

Researchers have tried modeling emotion generation using agent-based concepts. Zhenhua Cai *et al.* [18] have proposed a dynamic computational emotion model for emotion generation. The model consists of simulated agents (*avatars*) which interacts with environment and executes different tasks to satisfy demands. The model generates emotions like *Happy*, *Sad* and *Fear* as a result of interactions among the *avatars* and the environment. Becker *et al.* [19] show how boredom can be used to augment emotions and hence the mood of an agent. Raïevsky *et al.* [20] show the manner in which decision-making capabilities of an agent are affected by emotions. Ricardo *et al.* [21] have used an agent based emotion model to study how emotion supports functional adaptation to achieve system autonomy. Nair *et al.* [22] have proposed a set of complex dynamics to generate deep emotions using a multi-agent system. They have shown how emotions can be churned by a set of agents running concurrently in a more human-like manner.

Evolving multiple agents or autonomous robots using discrete event simulations [23] falls in the domain of Artificial Life (AL). They involve biological mechanisms, which in turn provide insights into the complex problems faced in distributed networks, parallel computations and the realization of intelligent systems. Though AL has been used in several applications including computer viruses, autocatalytic nets, cellular automata, artificial nucleotides, immune systems and market simulations, it has not yielded significantly leaving several open problems that are still to be tackled [24]. AL is realized using software agents and autonomous robots to emulate or simulate the intricacies of a biological organism and its ecosystem. The underlying techniques used to realize AL rely mostly on bio-inspired mechanisms which include Evolutionary Algorithms, Artificial Neural Networks and Cellular Automata [23, 25–27]. Scheutz *et al.* [28] lists 12 significant roles that emotion could play in AL and propose an agent architecture that uses action selection, adaptation and social regulation. Nair *et al.* [22, 29] have portrayed a dynamic model for emotion generation and shown how it can be effectively used to control the speed of a robot on a terrain. In [30], the authors have used emotions for the survival of a robot in an artificial environment. They have shown that emotions can possibly aid a robot to perform better in terms of its utility in the environment. Their robot, which is situated in an artificial-life-like environment, uses a set of linear equations for calculating the internal urges coupled with a simple probabilistic model to motivate and execute a relevant task.

This paper extends the work reported in [30] by improving on the manner of computing the *Urges* and the decision-making process. By using the dynamics for emotion generation, as proposed by Nair *et al.* [22] in a robot's task selection process, we also try and portray the utility of emotions in improving the performance and survival of a robot in an artificial-life-like environment. We examine the performance with a more robust and deterministic model for the robot as opposed to the probabilistic model used in [30]. The calculation of *Urges* too involves a one-to-one correspondence with the emotion model outputs rather than a cumulatively derived quantity viz. *Comfort* level proposed in [30]. In the work described herein, this *Comfort* level has been used to merely quantify the state the robot in the environment and strictly for high level task selection. As suggested in [31], we have evaluated our model by running experiments with and without emotions and comparing the results. Further the paper presents a detailed analysis of the utility of emotion along with the results obtained from real experiments. Subsequent sections describe the emotion model, the environment the robot is situated in and the related dynamics together with the experiments performed and the results obtained.

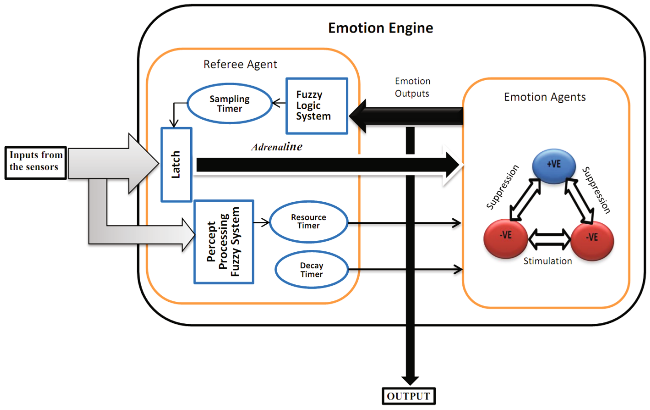


Fig. 1. The Emotion model and its dynamics based on [22]

2 The Emotion Generating Model

In order to aid the robot situated in an environment, in the task selection process, we have used the multi-agent emotion generation model proposed by Nair et al. [22]. This emotion model, unlike others is deep in nature as it features concurrent emotion generation by dedicated agents, stimulations and suppressions for both positive and negative emotions which have their own respective lifetimes, emotion decay based on a look-back into the past and an adrenalin based sampling of the environment. The latter sampling method makes the model to sample the environment at a faster rate when negative emotions predominate. The emotion agents behave like biological glands wherein an emotion resource regulates for the actual generation of emotion. This resource is charged intermittently based on the conditions within and those perceived from the environment. The agents generating the same type of emotion (positive or negative) stimulate their own kind (e.g. positive stimulates positive) while suppressing their opposites (e.g. positive suppresses negative) eventually making the model to churn out the resultant emotions.

Figure 1 shows the flow of information within the emotion model described in [22]. It consists of a Referee agent that acts as the band-master and controls the flow of signals, received internally as well as from the external environment, to the actual emotion generating agents. The external inputs are derived from sensors which the emoting robot uses to perceive its environment. The inputs are routed to the emotion generating agents based on an internally generated mood which triggers a sampling timer. When this internal Mood, which is the fuzzified version of all the emotions generated, is low the sampling timer triggers more frequently enabling the Latch accordingly consequently making the emotion agents to churn out fresh emotions at a faster rate. This makes the emotion engine emulate an adrenaline-like effect. The inputs perceived from the environment are also fuzzified to sense the external ambience which charges the emotion resource together with the rewards and penalties received from the environment. The Referee agent also controls the rate of decay of emotion using a timer whose value changes based on the state of the past emotions generated.

Equations that govern the dynamics of emotion generation as described in [22] are given below.

The Intensity of an emotion e at time $(t+1)$, I_e, is given by:

$$I_e(t+1) = I_e(t) + F_e(t) \tag{1}$$

$$F_e(t) = \begin{cases} \left(P_e(t) - I_e(t) + w_1 \sum_{k=1}^{E} S_{ke}(t) - w_2 \sum_{k=1}^{E} S'_{ke}(t)\right)\left(R_e(t)/R_{eMax}\right), & P_e(t) > I_e(t) \\ \left(w_3 P_e(t) + w_4 \sum_{k=1}^{E} S_{ke}(t) - w_5 \sum_{k=1}^{E} S'_{ke}(t)\right)\left(R_e(t)/R_{eMax}\right), & P_e(t) = I_e(t) \\ -\left|\left(I_e(t) - P_e(t) + w_6 \sum_{k=1}^{E} S_{ke}(t) - w_7 \sum_{k=1}^{E} S'_{ke}(t)\right)\left(R_e(t)/R_{eMax}\right)\right|, & P_e(t) < I_e(t), \\ & \text{and } k \neq e \end{cases} \tag{2}$$

where $F_e(t)$ is the fraction of resource extracted at time t which is calculated using equation 2, $P_e(t)$ is the input intensity, S_{ke} and S'_{ke} is the stimulation and suppression respectively, $R_e(t)$ is the current resource of the emotion e and R_{eMax} is its maximum value.

Stimulation (or Suppression) of an emotion k received by an agent generating emotion e:

$$S_{ke}(t+1) = \sum_x^M S_{ke}^x(t) + s_{\mathrm{ke}}(t),\ k \neq e \tag{3}$$

Stimulation and Suppressions generated by emotion e to be sent to other emotions:

$$\begin{aligned} S_e(t) &= s'_e(t+1) \\ &= \begin{cases} \left[\{P_e(t) - I_e(t)\}/I_e(t)\right]\{R_e(t)/R_{eMax}\}, & \\ & \text{if } P_e(t) > I_e(t) \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

Decay of emotion Resource $R_e(t)$ of an emotion e at time $t+1$:

$$R_e(t+1) = R_e(t) - |F_e(t)| \tag{5}$$

The decay in emotion intensity (I_e) of an emotion e at time $t+1$:

$$I_e(t+1) \ = \begin{cases} \left(I_e(t) - c_1 e^{-\Delta_e(t)}\right) for\ \Delta_e(t) > 0 \\ \left(I_e(t) - c_2 e^{\frac{-1}{\Delta_e(t)}}\right) for\ \Delta_e(t) < 0 \\ (I_e(t) - c_3)\, for\ \Delta_e(t) = 0 \end{cases} \tag{6}$$

Δ_e is given by:

$$\Delta_e(t+1) = c_4 \left[\sum_{i=t-(B-1)}^{t} I_e(i) - I_e(i-1) \right] /B - 1 \tag{7}$$

where c_1, c_2, c_3 and c_4 are positive non-zero constants.

The resultant emotions are used to drive the corresponding actuators of the robot. The emotion agents calculate their respective emotion intensities (I_e) based on the inputs received via the latch using equation 1. The fraction of emotion resource extracted $F_e(t)$ is calculated using equation 2 where w_i, $i = 1...7$, are positive non-zero constants. The stimulations and suppressions are modeled based on the equation 3. $R_e(t)$ is the resource of each emotion at time t while R_{eMax} is its maximum value. Stimulations and suppressions have a lifetime component. As long as the lifetime is non zero, these stimulations and suppressions continue to live within an emotion agent and affect one another. The overall stimulation (or suppression) within an emotion agent is calculated using equation 3. The stimulations and suppressions generated due to interactions across the agents are calculated using equation 4. The Resource, which forms the basis of emotion generation, is depleted as and when the corresponding emotion is generated and is governed by equation 5. It is charged as and when its associated timers triggers. The amount by which resource of an emotion is charged depends on its current value and the intensity of related input. Emotion decay is affected by the Decay timer using equation 6 depending on the manner in which the emotions change in a window of time in the past. The information on immediate past of an emotion within a window of time is aggregated using equation 7. Emotion resource, stimulations and suppressions and their life-times, charging of resources, decay of emotions along with the adrenaline effect all contribute to making the emotion model a deep one.

3 The Robot and Its Environment

Similar to an *animat* which inhabits in an artificial environment [32], the robot herein is also situated in an artificial life-like environment. Just as its biological counterpart, this robot has both physical *needs* and *urges.* Figure 2 shows the various functional blocks and decision-making process within the robot. *Needs* include the requirement of water and food, among which have thirst and hunger as their correspondingly *urges.* The physical needs of the robot include:

Food	This need creates a requirement within the robot to consume resources from the environment.
Water	This is similar to the need of Food except robot is requirement to consume a different resource viz. water from the environment.
Energy	Robot loses energy while performing different activities in the environment. A need for *Energy* makes the robot to convert the resources it consumes from the environment into useful energy.
Recce	The prime objective of the robot in the environment is to perform useful tasks. A need to reconnoiter (*Recce*) is aggravated if the robot is busy fulfilling its other needs viz. *Food*, *Water* or *Energy* instead of carrying out any useful activity.

These *needs* are computed based on, which task the robot performs, its urges and the condition of the environment. The corresponding meters (Figure 2) reflect their respective amounts. The *needs* cause changes in the corresponding urges which in turn effect the decision-making and hence the behavior of the robot. A behavior is exhibited by executing a task. *Urges* form a set of entities that are internally generated within the robot based on how comfortably it is when it is situated within its environment. These include:

Hunger	This urge motivates the robot to perform a task which involves consumption of resources from the environment.
Thirst	Robot is motivated to drink when its *Thirst* become high.
Explore	This urge propels the robot to explore the environment in order to perform useful tasks.
Relax	This urge is associated with energy. The robot relaxes when this urge becomes high and in turn generates energy.

The environment supports the robot by providing *Water*, *Grass* and *Shrubs* which when consumed generate the required energy for the robot to survive. As the robot consumes these resources, they are once again topped up by three simulated natural phenomena viz. *Wind*, *Rain* and *Heat*. While the former two increase all the resources, the latter tends to decrease them. In our experiments we have used the former two to simulate a tropical environment and the heat from the sun to deplete the resources portraying a drought-like situation.

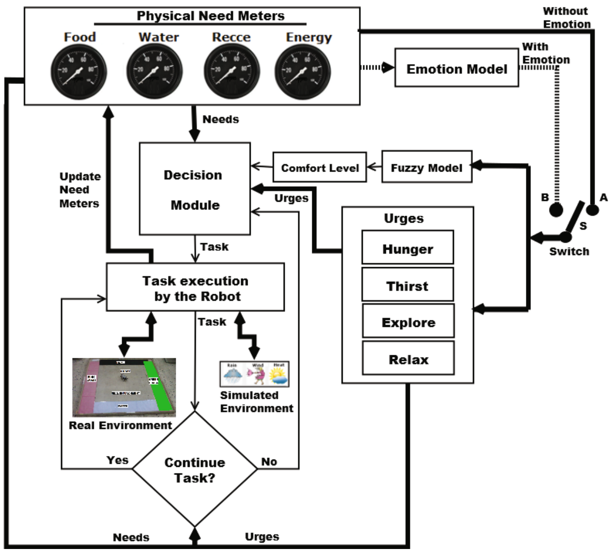


Fig. 2. Various functional blocks and decision-making process

Using on-board sensors the robot in the real world detects the following resources in the environment:

1. Water : It is assumed that *Water* is blue in colour and is available as a blue coloured patch in the environment. Robot detects *Water* using its colour sensor.
2. Grass : A green coloured patch in the environment forms an area designated as *Grass*. *Grass* is required by the robot to fulfil its need of *Food*.
3. Shrubs : *Shrubs* are designated in the environment as a yellow coloured patch.

The robot is capable of performing two categories of tasks viz.

– *Low profile* tasks which include Eating and Drinking which in turn satisfy its primary needs of generating energy.
– *High profile* tasks that include Exploring, Relaxing and Learning. These tasks constitutes the performance of the robot in the environment. More

the robot performs these high level tasks, more will be its productivity or usefulness in the environment.

Exploration facilitates it to move around its environment to re-assess and estimate the amount of current resources and also to discover the terrain. The robot relaxes for two reasons - either to convert the food or water taken in, into energy or simply because all its other needs have been satisfied. The task of Learning is triggered when the *Comfort* level of the robot is high and its needs and the corresponding *urges* (shown in Table 1) are low. Learning can act as a feedback mechanism and influence the decision-making process thereby making the robot perform better. The *Comfort* level of the robot is found by a fuzzy logic based system that takes in either the physical need meter values or the emotions generated by the emotion model and portrays how good the robot feels within its environment.

In a true robotic environment a low profile tasks could stand for charging the on-board batteries or cleaning the dust of its sensors, while high profile tasks could refer to transmitting of the data gathered by the robot to the base station or processing the same to find some meaningful rules which could in turn aid in learning and adapting to the environment. Thus the main objective herein is to force the robot to perform more of high level tasks than the mundane low level ones during its life-time.

The overall success with which the robot lives and survives in the stochastic environment depends greatly on the decisions it makes. As shown in Figure 2, the decision-making process takes as input the current values of *needs*, *urges* and the *Comfort* level and outputs the task to be performed. The *Comfort* level (α) is a qualitative measure of the quality of life of the robot. A high *Comfort* value signifies that all its physical needs are low and thus the robot has no real requirements. Such a situation motivates the robot to perform high profile tasks rather than the mere lower ones of Eating and Drinking. This *Comfort* level is determined using a fuzzy system. Table 2 lists the fuzzy rules used to obtain the value of *Comfort* level. While the *needs* signify the actual requirements of the robot, *urges* act as dynamic thresholds to check on these requirements. When the Switch S is connected to point A (Figure 2), the robot does not use the emotion model for its decisions. While a connection at point B embeds the emotion model within the decision-making loop of the robot. The switch thus aids in conducting experiments with and without the emotion model to aid in the decision-making process. The emotion model used herein is similar to the described by Nair *et al.* [22] and uses three emotion agents to produce three emotion viz. Happy, Sad and Surprise. The Physical Need Meters of the robot are provided as inputs to the emotion model. While the *Energy* and *Recce* need meter values are fed to the Happy and Surprise emotion agents respectively to produce the corresponding outputs, *Food* and *Water* need meter values are added up and provided as input to the Sad emotion agent to churn out the Sad emotion. After a task is selected, the robot executes the tasks based on certain conditions derived from its *needs*, internal *urges* and the availability of the concerned resource (*Grass*, *Shrubs* or *Water*). Until these conditions remain true, the robot keep on executing the current task. For instance if the robot decided to eat then before performing this task it would require to verify whether its need for *Water*, *Recce* and *Energy* are below the *thirst*, *explore* and *relax* urges respectively. If it cannot execute the task then it goes back to previous level to select another task. Performance of a task causes the concerned resource in the environment to decrease and also affects the corresponding *needs*. After the task is executed the cycle is repeated.

Table 1. Related Needs, Urges and Tasks

Needs (Υ)	**Urges (Ω)**	**Tasks (Γ)**
Food (f)	Hunger (H)	Eating
Water (w)	Thirst (T)	Drinking
Recce (r)	Explore (L)	Exploring or Learning
Energy (g)	Relax (X)	Relaxing

Table 2. Fuzzy rules for deriving *Comfort* level

Recce	**Food/Water**	**Energy**	**Comfort Level**
low	low	low	super
low	low	medium	super
low	low	high	good
low	medium	low	super
low	medium	medium	good
low	medium	high	okay
low	high	low	good
low	high	medium	good
low	high	high	okay
medium	low	low	good
medium	low	medium	good
medium	low	high	okay
medium	medium	low	good
medium	medium	medium	okay
medium	medium	high	good
medium	high	low	okay
medium	high	medium	okay
medium	high	high	bad
high	low	low	okay
high	low	medium	okay
high	low	high	bad
high	medium	low	okay
high	medium	medium	bad
high	medium	high	worst
high	high	low	bad
high	high	medium	worst
high	high	high	worst

4 The Decision-Making Process

Table 1 shows the *needs* and their corresponding *urges* and the associated tasks. The robot has to perform the associated task Sin order to satisfy the respective *needs* and *urges*.

The robot makes a decision to perform a task based on the following equations:

$$\tau = \arg\min_t \{W_t\} \tag{8}$$

$$W_t = \frac{(Q^i - \mu^j) \times \max \eta^i}{\eta^i} \tag{9}$$

$$Q^i = \max \eta^i - \eta^i \tag{10}$$

$$\mu^H = \frac{\{H_{max} - \tilde{Q}^f\} \times k_3}{\tilde{Q}^g} \tag{11}$$

$$\mu^T = \frac{\{T_{max} - \tilde{Q}^w\} \times k_4}{\tilde{Q}^g} \tag{12}$$

$$\mu^L = \frac{\tilde{Q}^g \times k_1}{\tilde{Q}^r} \tag{13}$$

$$\mu^X = \frac{\{E_{max} - \tilde{Q}^g\} \times k_2}{E_{max} - \frac{\mu^H + \mu^T}{2}} \tag{14}$$

$$\tilde{Q}^i = \max \eta^i - E\{\eta^i\} \tag{15}$$

$i \in \Upsilon,\ j \in \Omega,\ t \in \Gamma$
where,

- τ is the selected task.
- t is the task associated with need (η^i) and urge (μ^j).
- η^i is the value indicated by the need meters associated to the need i.
- μ^j is value of urge associated to the task t as shown in Table 1.
- $\max \eta^i$ is the maximum possible value of η^i.
- Q^i signifies the actual amount of *Food*, *Water*, *Recce* and *Energy* available within the robot.
- $\tilde{Q}^i = Q^i$ when the emotion model in not used i.e. the switch S is at point A in Figure 2.
- $\max \mu^H$, $\max \mu^T$ and $\max \mu^X$ are the predefined maximum possible values of μ^H, μ^T and μ^X respectively.
- k_1, k_2, k_3 and k_4 are positive non-zero constants.
- $E\{\eta^i\}$ is the output of the emotion model for the input η^i.

The robot makes a decision to perform a task (τ) based on equation 8 which selects the task having the minimum weight (W_t). It is evident from equations 8 and 9 that when the need for a task is high then its chances of selection are also high. In equation 9 the numerator on the right hand side indicates the motivation of the robot to perform a specific task which is scaled by its respective physical need. If multiple Q^i values are greater than their corresponding *urges* (μ^j) then the task t associated with the $\max \eta^i$ is selected. Further, if the task selected

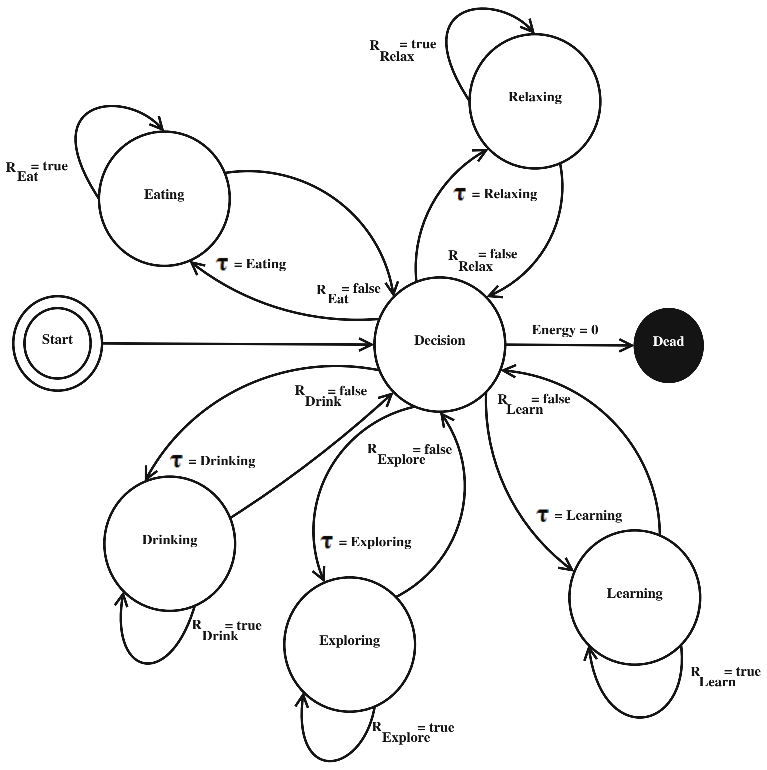


Fig. 3. The State Diagram of Decision Process

is that of Exploration and the value of α is above an upper limit (α_{th}) then the robot is assumed to be in a very comfortable situation. In such a condition, instead of continuing with the task of Exploration it exhibits its high comfort by selecting the task of Learning.

The *urges* are calculated based on the equations 13 to 15. The urges always try to pull down the needs if they are high and also try to lift them when they are low. Thus the urges act as dynamic thresholds and maintain the internal milieu of the robot in a dynamic equilibrium. Emotions play a vital role in determining the urge to select a particular task which is why emotions have been used to calculate the *urges*. The values of all the *urges* oscillate below an upper limit. As can be seen from equations 11 and 12, the *urges* associated to *Hunger* and *Thirst* are proportional to the corresponding requirements needed to reach their maximum values and inversely proportional to the current energy. The *urge* to *Explore* is directly proportional to the energy of the robot and inversely proportional to the current *Recce* value. The *urge* to *Relax* depends on the need to gain energy provided that the robot has enough *Food* and *Water* (i.e. needs for *Food* and *Water* is low) to generate same.

Once a decision to execute a task is made then the robot continues to execute the same till any of the *urges* (μ^j) other than that of the current task exceeds Q^i. Figure 3 shows the state diagram for task selection by the robot. The robot selects a task (τ) when it is in the Decision state and finally exits when the energy becomes zero (*Dead*). As seen in Figure 3, the robot continues to perform the same task as long as the associated condition to that state is true and goes back to the Decision state when it is false. To continue execution of a particular task in a state respective conditions of that state must be true. These conditions for various task's states are listed below:

R_{Eat} : $\underset{i,j}{\forall} Q^i \geq \mu^j, i \in \Upsilon - \{f\}, j \in \Omega - \{H\} \bigwedge Q^f \leq (\max \eta^f \bigwedge (Q^f_{initial} + k_E)) \bigwedge (Grass \bigwedge Shrubs > 0)$

R_{Drink} : $\underset{i,j}{\forall} Q^i \geq \mu^j, i \in \Upsilon - \{w\}, j \in \Omega - \{T\} \bigwedge Q^w \leq (\max \eta^w \bigwedge (Q^w_{initial} + k_D)) \bigwedge (Water > 0)$

R_{Relax} : $\underset{i,j}{\forall} Q^i \geq \mu^j, i \in \Upsilon - \{g\}, j \in \Omega - \{X\} \bigwedge Q^g \leq (\max \eta^g \bigwedge (Q^g_{initial} + k_R)) \bigwedge Q^f > 0$

$R_{Explore}$: $\underset{i,j}{\forall} Q^i \geq \mu^j, i \in \Upsilon - \{r\}, j \in \Omega - \{L\} \bigwedge Q^r \leq (\max \eta^r \bigwedge (Q^r_{initial} + k_X))$

R_{Learn} : $\underset{i,j}{\forall} Q^i \geq \mu^j$, $i \in \Upsilon$, $j \in \Omega \bigwedge Number\ of\ Iteration \leq k_L$

where k_E, k_D, k_R, k_X and k_L are positive non-zero constants.

Thus to continue the task of Eating, all *urges* other than *Hunger* must be satisfied (i.e. $Q^i \geq \mu^j$), the *need* for *Food* must be non-zero and the robot should have eaten less than k_E units of *Grass* or *Shrubs* more than the initial value of *Food* ($Q^f_{initial}$) when it entered this state and the resources viz. *Grass* and *Shrubs* in the environment must be non-zero. The rules for other tasks can be explained similarly except that of Learning. The robot will continue to be in Learning state provided all urges are low and the number of Learning iterations (looping within the Learning state) has not exceeded k_L.

5 Environment Dynamics

The environment of the robot is influenced by *Heat*, *Rain* and *Wind* all of which effect the resources viz. *Water*, *Shrubs* and *Grass*. *Heat* depletes *Water*, *Shrubs* and *Grass* present in the environment. On the contrary *Rain* catalyzes the growth of *Shrubs* and *Grass* and restores *Water* within specific regions of the environment. We characterize *Wind* as a phenomenon which only aids in the growth of *Shrubs* and *Grass*. The total amount of resources added or depleted to or from the environment is given by the equation:

$$R_{\pm} = \xi \times \theta \times d \tag{16}$$

where $R_{\pm}$ is the total amount of resources added or depleted to or from the environment, θ is the increment or decrement rate at which the resources are restored or depleted and can be positive or negative depending on the active natural phenomenon (*Heat*, *Rain* or *Wind*). Thus when it rains θ is positive while when heat is turned on it is negative. ξ is the intensity of environmental phenomenon and d is the duration of the current phenomenon.

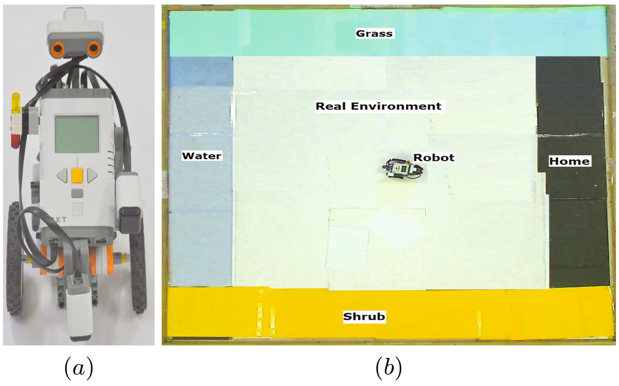


Fig. 4. (a) The Lego NXT® robot (b) The robot in its physical environment

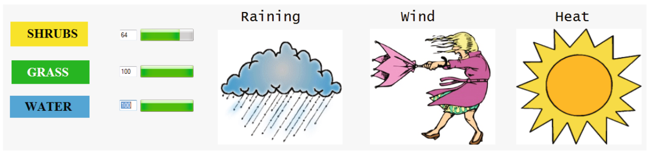


Fig. 5. The simulated environment used to vary the resources

6 Experimental Setup

The basic objective of the experiments conducted was to find whether the new set of dynamics for decision-making (non-probabilistic model) can in anyway improve the performance of the robot described in [30]. Figure 4(a) shows the LEGO® MINDSTORM® NXT robot used in the implementation. The robot was armed with two sensors viz. a colour and a compass sensor. Communication to the main computer hosting the decision-making process, the emotion model and the simulated environment was achieved using Bluetooth®. Java 1.7 in conjunction with Lejos NXJ® API was used for programming the robot. Figure 4(b) depicts the physical environment sensed by the robot. The green and the yellow strips constitute the *Grass* patches and *Shrubs* respectively while the blue portions form the *Water* body. The black strip constitutes the home base where the robot performs the tasks of Relaxing and Learning. The map of this physical environment is embedded within the robot. Thus the robot is capable of navigating towards the *Water* body, *Grass* and *Shrubs* within its environment using its on-board sensors (colour sensor to detect the colour of the strips and compass sensor for navigation). The natural phenomena (*Wind*, *Rain* and *Heat*) and the restoration and depletion of the actual quantum of resources (*Water*, *Grass* and *Shrubs*) were simulated by a program running on the computer hosting the decision-making process. A screen shot of the simulated environment is depicted in Figure 5. The robot was made to perform tasks by physically moving and sensing the environment.

The robot was programmed to be capable of performing the following tasks. Each of these tasks are performed until the constraint given in Section 4 becomes false, which is when a fresh decision-making process is invoked.

- Eating: This is a low profile task. Here the robot moves towards the *Grass* or *Shrubs* patches (designated by green and yellow strips in the environment as shown in Figure 4(b)) and stays on the patch to enact the process of eating. By doing so, the robot augments its food reserve and thus fulfills its hunger. Hence the need for *Food* is decreased while those for *Water*, *Recce* and *Energy* is increased by constant non-zero values.
- Drinking: This low profile task makes the robot move to the *Water* (designated blue in the environment) and stays there to augments its water reserve. By doing so, its *need* for *Water* decreases while those for *Recce* and *Energy* are increased by constant non-zero values.
- Relaxing: This high profile task makes the robot to rest by moving towards the black patch in the environment. Having rested in this region the robot is made to inherently feel a gain in energy. This is done by decreasing its need for *Energy* and increasing the needs for *Food*, *Water* and *Recce* by constant non-zero values.
- Exploring: While exploring, the robot visits one of the resources in its physical environment. In the process it updates itself with the actual value of that resource in the environment and decreases its need to *Recce* by a constant non-zero value. By doing so, it loses its energy and the needs for *Food*, *Water* and *Energy* are increased by constant non-zero values. Exploring provides the robot with the actual amount of resource within its environment. This value is used by the robot in its internal computations as a base till the execution of the next exploration task. It may be noted that the asynchronous natural phenomena (Wind, Rain and Heat) may alter the actual amount of resource, but since the robot can sense these changes only during an explorative task it depends on the values of the resources it sensed during its last exploration. These inconsistent values are used by it during all decision-making processes. Thus if the robot is forced to explore more frequently, its internal model of the actual environment is bound to be more consistent. This makes exploration a high level task.

– Learning: This is a very high profile task which is performed when the *Comfort* level of the robot is high and its *needs* and *urges* for other tasks are low. This task makes the robot to move towards the black patch and stays there to enact the process of learning. Learning increases the need for *Food*, *Water* and *Energy* and decreases the need to *Recce* by constant non-zero values.

It may be noted that the robot loses energy even while moving from one location to another in its environment which in turn increase its need for the same. When its energy becomes zero we assume the robot to be dead thus ending the experiment.

Experiments were carried out in the absence (Switch at position A in Figure 2) and in the presence (Switch at position B in Figure 2) of the emotion model as shown Figure 2. The physical need meter values were fed directly as inputs to the urge generating block when the emotion model was not used. While using the emotion model, the current Energy of the robot is fed to the positive emotion agent while the need to recce and needs for *Food* and *Water* (combined) were fed to the negative emotion agents respectively.

The parameters used in the experiments carried out are listed below:

Shrubs { θ_i = 0.3, R_{max}=100, θ_d =-0.7, θ_r =-0.8},
Grass { θ_i =0.7 , R_{max} =80, θ_d =-0.3, θ_r =-0.3},
Water { θ_i =0.7 , R_{max} =100, θ_d =-0.3, θ_r =-0.2 },
Rain $\{10 < \xi \leq 20, 5 < d \leq 15\}$,
Wind $\{5 < \xi \leq 20, 1 < d \leq 8\}$,
Heat $\{5 < \xi \leq 15, 1 < d \leq 8\}$,
$\max \eta^f = \max \eta^w = \max \eta^r = \max \eta^g = \max \mu^H = \max \mu^T = \max \mu^X = 100$, $k_1 = 30$, $k_2 = 30$, $k_3 = 30$, $k_4 = 25.5$, $k_E = k_X = 40$, $k_D = k_R = 60$, $k_L = 10$, $1 \leq \alpha \leq 10$ and $\alpha_{th} = 8$

where, θ_i and θ_d are increment and decrement rates of the resources respectively caused due to the environmental phenomenon (*Rain*, *Water* and *Heat*) while θ_r is the reduction rate of the resources in the environment caused due to their consumption (Eating or Drinking) by the robot.

7 Results and Discussions

The performance of the robot was tested in two scenarios of the environment:

1. Tropical: Tropical conditions were induced by varying *Heat*, *Rain* and *Wind* in the manner shown in Figure 6. As mentioned earlier, Rain and Wind catalyze the growth of *Grass* and *Shrubs* and thereby increase the resources in the environment. In Figure 6, Heat is also introduced in the former and later parts of the graph to curtail any excessive growth of resources. The corresponding resources generated within this tropical environment is shown in Figure 7. It can be seen here that these resources constantly remain on the higher side indicating an abundant source of *Water*, *Grass* and *Shrubs* for the robot.

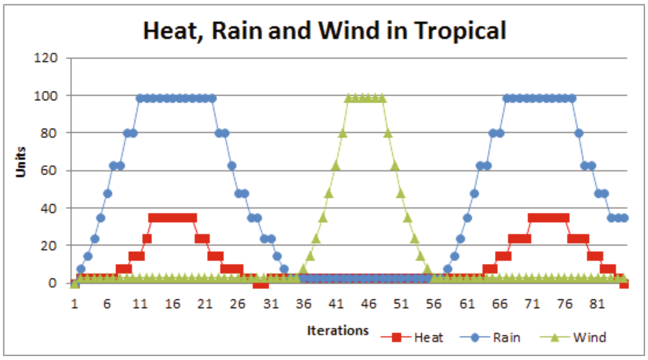


Fig. 6. Varying Natural phenomena in Tropical environment

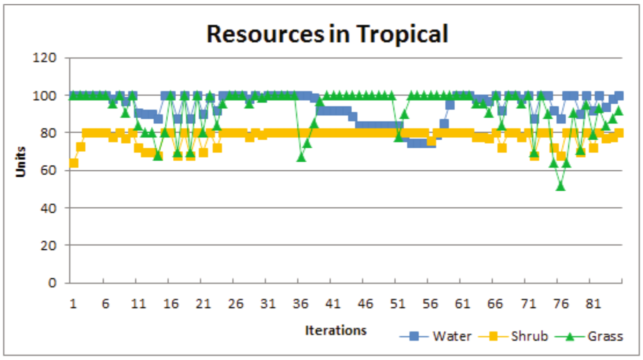


Fig. 7. Variation of resources in Tropical environment

2. Drought: Drought-like conditions were created by initially allowing the resources to grow to a high value by increasing both Rain and Wind for a certain period. Heat was constantly increased from a low value to its maximum and made to extend beyond this period. The variations of Heat, Rain and Wind induced to create a drought condition is shown in Figure 8. The corresponding variations of the resources under the drought condition is shown in Figure 9. As can be seen beyond iteration 11 in Figure 8 Heat increase in the absence of Rain and Wind causing rapid depletion in resources beyond iteration 11 in Figure 9 eventually becoming zero beyond 15 and thus forcing a drought-like condition.

The following experiments were conducted by placing the robot within the Tropical and Drought environments:

– Condition 1: Tropical, with switch S at position A (emotion model absent).
– Condition 2: Tropical, with switch S at position B (emotion model present).
– Condition 3: Drought, with switch S at position A (emotion model absent).
– Condition 4: Drought, with switch S at position B (emotion model present).

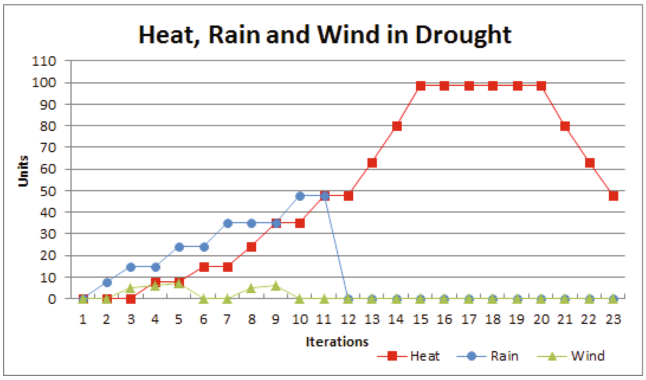


Fig. 8. Varying Natural phenomena in Drought environment

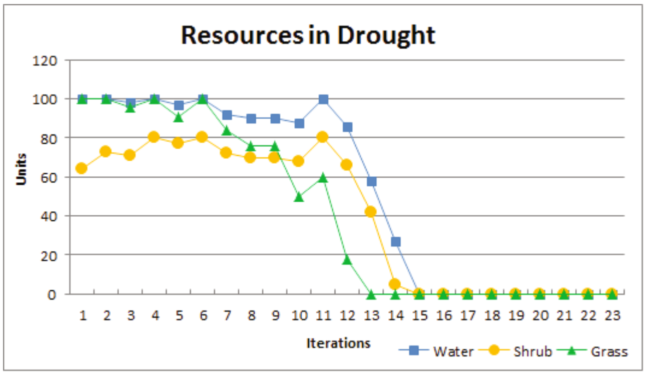


Fig. 9. Variation of resources in Drought environment

Since there is always some stochastic element in the form of noise induced while physically sensing the environment, each of these experiments was repeated five times and the corresponding values were averaged and graphs plotted accordingly.

A discussion on the comparisons of the Comfort Levels, Lifetimes, Tasks executed and their distribution over the robot's lifetimes follows.

7.1 Comfort Levels, Lifetimes and Tasks

Since the *Comfort* level changes over the lifetime of the robot, an average value of the *Comfort* level was found over the lifetime in an experiment for a specific Condition. Five such values obtained in the set of experiments were then averaged to obtain the average mean of the *Comfort* level for that Condition.

The graph in Figure 10 shows the average mean of *Comfort* (α) levels of the robot while that in Figure 11 shows the corresponding average lifetimes for each of the Conditions. Figure 12 depicts the high and low profile tasks executed

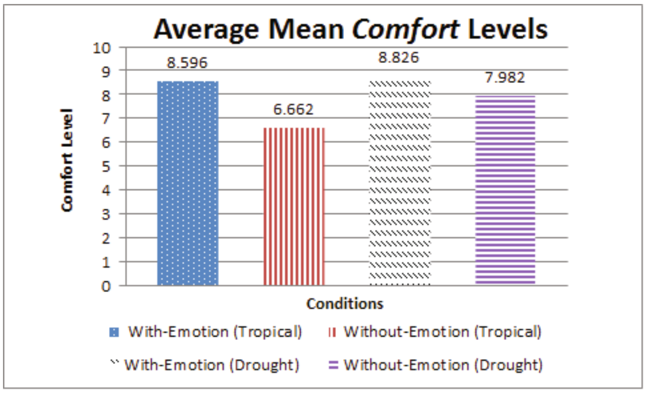


Fig. 10. Average Mean *Comfort* Levels of the robot for various environment Conditions

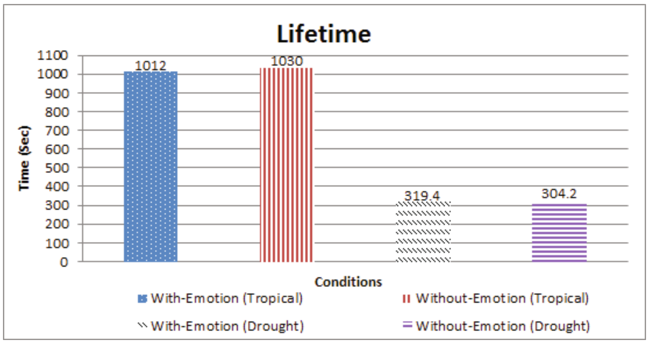


Fig. 11. Average lifetime of the robot in seconds in different Conditions

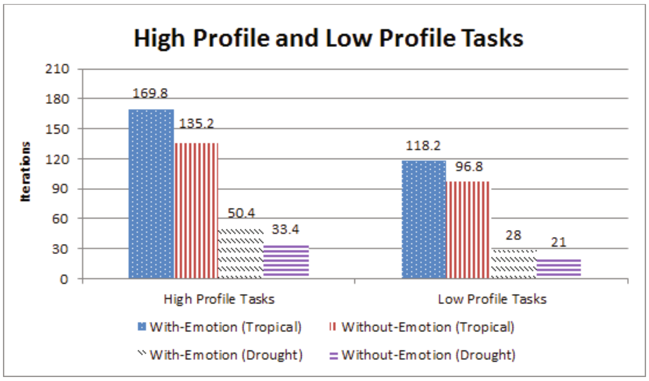


Fig. 12. Average duration (δ_t) for which High and Low profile tasks were performed by the Robot

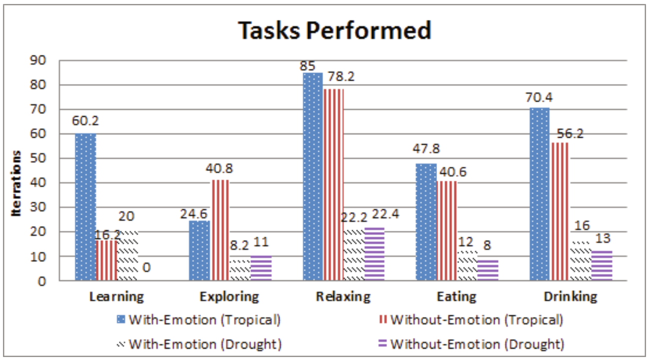


Fig. 13. Average duration (δ_t) for which various tasks were performed by the robot in different Conditions

and their averaged durations ($\sum \delta_t$) during the lifetime of the robot under the different Conditions.

It can be clearly seen from the graph in Figure 10 that the *Comfort* level of the robot with emotion is higher than those obtained without emotion in both tropical and drought environments. It may be noted that the *Comfort* levels for both with and without emotion in case of the drought environment (8.826 and 7.982) is higher than their respective counterparts in the tropical environment (8.596 and 6.662). This anomaly is mainly because the lifetimes of the robot in case of the drought environment is far less than those in case of the tropical environment (as can be inferred from the graph in Figure 11) and the *Comfort* values plotted herein are averaged over these lifetimes.

Though the lifetimes in tropical (with and without emotion) and likewise in drought environments (with and without emotion) are almost same, the amount of time the robot spends in performing the high and low level tasks is consistently higher when the emotion model is used as shown in the graph in Figure 12. This

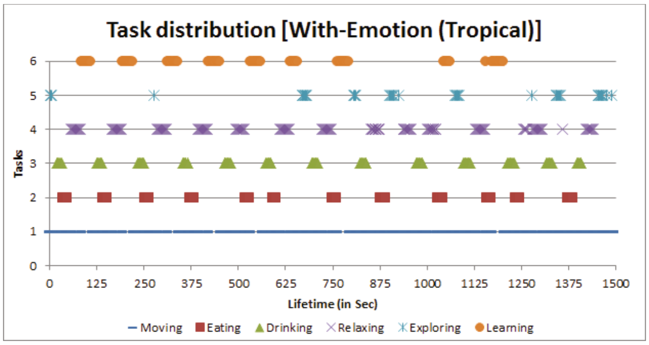


Fig. 14. Tasks performed by the robot having emotions during its lifetime in Tropical environment

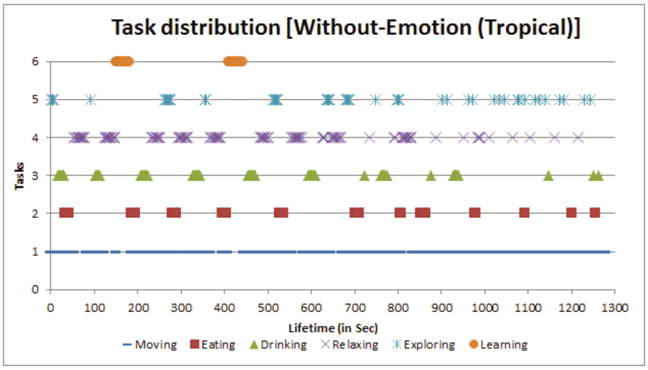


Fig. 15. Tasks performed by the robot without emotions during its lifetime in Tropical environment

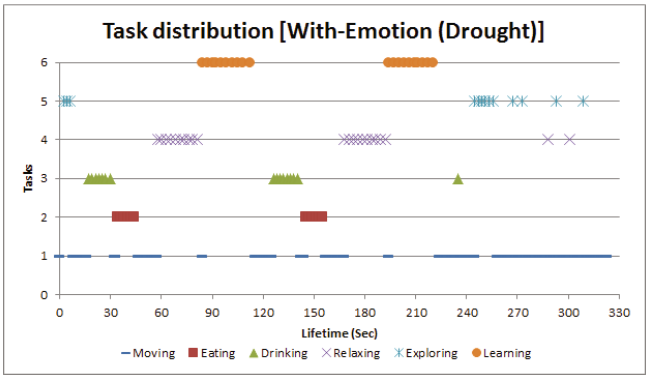


Fig. 16. Tasks performed by the robot having emotions during its lifetime in Drought environment

shows that in a given lifetime the robot using the emotion model out-performs the one not using this model by performing more number of tasks (high and low) irrespective of the environments (tropical and drought). We have assumed the duration (δ_t) of a particular task t in a single decision step as the number of iterations of that task performed until the robot switches back to the Decision state (Figure 19).

Figure 13 shows the duration ($\sum \delta_t$) for which each task was performed for all the Conditions. Except for the task of exploration the durations for which the tasks are performed for both tropical and drought environments are consistently higher when the emotional model is used. The high level task of Learning is performed only when the task of Exploration is triggered and the *Comfort* level is beyond an upper limit (α_{th}) as mentioned in Section 4. As can be seen in Figure 13, since this upper limit is not achieved the task of Exploration is done for a higher period of time when emotion is not used. Accordingly the amount

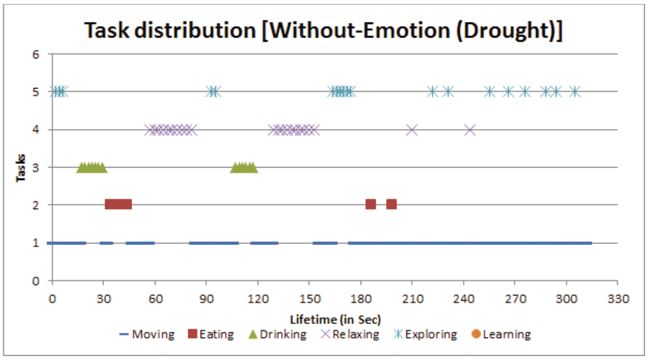


Fig. 17. Tasks performed by the robot without emotions during its lifetime in Drought environment

of time for which Learning is performed is proportionately less. The converse is true when the emotion model is used indicating clearly that α_{th} is crossed more often. Further Learning was never attempted in all experiments under Condition 4 (Drought, without emotion model) while the same for Condition 3 (Drought, with emotion model) was performed for a substantial interval (20 iterations on an average) considering the short life spans for which the robot lived in each of the Conditions (319.4 and 304.2). It may thus be inferred that a higher value of *Comfort* level makes the robot perform more number of tasks. More importantly, emotion seems to increase the *Comfort* level thereby motivating the robot to perform high level tasks even under extreme conditions.

Figures 14 - 17 show the times at which the various tasks were triggered for each of the Conditions 1-4 in a sample experiment. Each task is plotted in the graph along the Y-axis. Eating, Drinking, Relaxing, Exploring and Learning are plotted as tasks 2, 3, 4, 5 and 6 respectively. At any point of time only one task is triggered. Since the triggered tasks are to be executed at different locations in the environment the robot needs to move from its current position to the concerned position. For instance in order to eat, it has to move towards the green patch from its current location. Some time is lost in such movements and none of tasks from 2-6 are executed during these periods. The times consumed in such movements is indicated as task numbered 1 in these graphs.

It can be seen that while the lower level tasks were performed quite frequently and almost uniformly for the tropical cases with and without emotion (Figures 14 and 15), the higher level task of Learning was triggered more often when the emotion model was used and is spread across the lifetime. The durations for which the robot performs the task of Learning is approximately 30 seconds in both the Conditions (Tropical, with and without emotions).

In case of the Drought Conditions (Figures 16 and 17) in the absence of emotion since the α_{th} value is not reached Learning is never triggered though-out its life-span where with emotion Learning is triggered twice for a considerable amount of contiguous time (28 seconds). It may also be observed that the sequence of tasks selected by the robot in both with and without emotion cases is almost the same except that Learning is triggered in place of Explorations when emotion is used. This sequence, Exploring-Drinking-Eating-Relaxing-Exploring/Learning-Drinking-.., seems to be a logical one wherein Drinking and Eating is followed by Relaxing to gain energy. Once the energy is high, it is motivated to Explore/Learn. Emotions thus seem to play a significant role in forcing the robot to trigger the task of Learning during its lifetime.

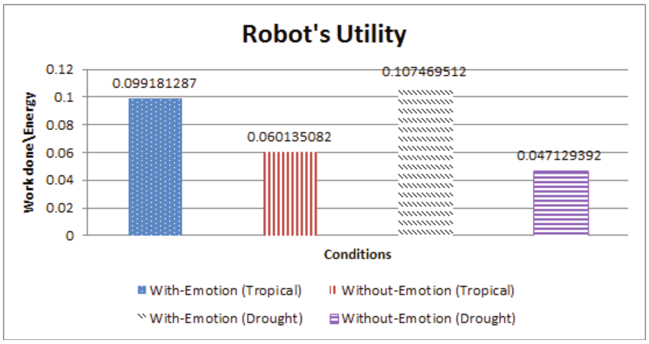


Fig. 18. Utility of the robot in various Conditions

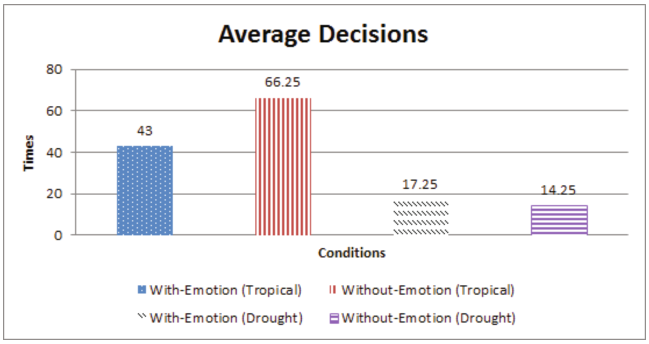


Fig. 19. Average number of decisions in different Conditions

7.2 Robot's Utility and Decisions

The utility of the robot is calculated as the amount of work done by the robot per unit of energy consumed. In the present case we have considered only the more important tasks of Exploring and Learning to signify the real work done by the robot in its environment. These two tasks have been equally weighted and their respective durations have been taken into consideration in the calculation of the this utility. As can be seen from Figure 18, the utility of the robot is always high (0.09 in Tropical and 0.1 in Drought) when emotions are embedded within the robot. While the cases when the emotions are not used show a low utility (0.06 in Tropical and 0.04 in Drought). Thus one may infer that emotion motivates the robot to perform better in the environment.

The number of times the decision-making process is invoked in a lifetime of a robot is a indirect indicator to the stability of the system. Too many and too frequent decisions would necessarily mean that the robot is almost constantly moving from one place to the other to execute a task in the environment. Such movements result in a waste of energy and the robot does not perform any meaningful tasks. Figure 19 shows the average number of times the decision-making process was triggered by the robot in all the four Conditions. It is clear from this graph that in the tropical case the number of times the robot with emotion goes into the decision-making mode is far less than when it does not use emotion. On the contrary in the drought environment the robot with emotion seems to going to this mode more often than when it does not use emotions. From this graph one may conclude that emotion may best be used for decision-making under conditions of abundance as in the tropical case where resources are in plenty. For cases when the resources are low as in case of the drought situation a emotionless or logical decision-making procedure may be the best suited. This is analogous to the advice given to a person, in an adverse or dangerous situation, to suppress his/her emotions and act in a logical manner.

8 Real World Metaphors

The *needs* and *tasks* of the robot described in this paper can be mapped to their real world equivalents. The need to go to areas where *Grass*, *Shrubs*, *Water* and *Energy* are available could be looked upon as the need to search for battery charging sources by a robot within its environments. As the robot inhabits its environment it also collects data and preserve within. With limited amount of memory available on-board the robot may feel the need to send and thus off-load the data to its base or controlling station so as to make room for fresh data.

The task of exploration could mean mapping an unknown terrain and building its map. Learning could be associated with high level optimization tasks or mining of the data acquired.

9 Conclusions

The basic objective of the work described herein was to verify whether emotions can play a significant role in enhancing the performance and utility of a situated robot. In [30], the authors have presented some preliminary results that indicate emotion to play a role in decision-making. In the present work too, it was found that an increase in *Comfort* level motivates the robot to choose tasks having higher profiles thereby increasing its utility. Based on results portrayed herein, one may infer that emotion plays a vital role in reducing the number of decisions made when the environment offers an abundance of resources. This is clear from the Tropical case with emotion. Since the number of times the decisions taken is lower than that without emotion in Tropical case the robot seems to be performing more useful higher level tasks rather than merely switching tasks. However the converse is true in the case when there is a scarcity of resources in the environment as in the Drought Condition. The amount of tasks both high and low profile performed in abundant Tropical environment using emotion is also for more than when emotion is not used. This difference in the number of tasks performed in the Drought Conditions (with and without emotion) is however not significant. Further emotion does not seem to play any significant role in increasing the lifetime of the robot in either of the environments (Tropical or Drought). One may thus conclude that under conditions of abundance in the given environment it is preferable for the robot to opt for emotion based decision-making.

We envisage to develop a learning model which could further aid the robot to optimize the task selection process. The realization of a hybrid model which causes the switch to move from A to B (in Figure 2) adaptively is also being looked into so as to make the best use of emotion based model.

Acknowledgements. The work reported herein was carried out at the Robotics Lab.[1], Department of Computer Science and Engineering, Indian Institute of Technology Guwahati. The authors wish to acknowledge the funding provided by Department of Science and Technology, Government of India under the FIST scheme to setup this Lab. The first author would like to acknowledge MHRD, Govt. of India for providing the financial assistantship during the research.

References

1. Harl, K.M., Chang, L.J., van't Wout, M., Sanfey, A.G.: The neural mechanisms of affect infusion in social economic decision-making: A mediating role of the anterior insula. NeuroImage 61(1), 32–40 (2012)
2. Hudlicka, E.: Modeling the mechanisms of emotion effects on cognition. In: Proceedings of the 2008 AAAI Fall Symposium. The AAAI Press, Menlo Park (2008)
3. Fong, T., Nourbakhsh, I., Dautenhahn, K.: A survey of socially interactive robots. Robotics and Autonomous Systems 42(3-4), 143–166 (2003)
4. Antos, D., Pfeffer, A.: Using emotions to enhance decision-making. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, vol. 1, pp. 24–30. AAAI Press (2011)
5. Parisi, D., Petrosino, G.: Robots that have emotions. Adaptive Behavior 18, 453–469 (2010)
6. Lee, W.P., Kuo, J.W., Lai, P.C.: Building adaptive emotion-based pet robots. In: Proceedings of the World Congress on Engineering, WCE 2008 (2008)

[1] `www.iitg.ernet.in/cse/robotics`

7. Liang, J., Dong, D., Wang, X.: An artificial endocrine-emotion model based on fuzzy logic. In: Zhu, R., Ma, Y. (eds.) Information Engineering and Applications. LNEE, vol. 154, pp. 307–314. Springer, London (2012)
8. Kim, H.R., Kwon, D.S.: Computational model of emotion generation for human-robot interaction based on the cognitive appraisal theory. Journal of Intelligent and Robotic Systems 60(2), 263–283 (2010)
9. Park, J.W., Kim, W.H., Lee, W.H., Chung, M.J.: Artificial emotion generation based on personality, mood, and emotion for life-like facial expressions of robots. In: Forbrig, P., Paternó, F., Mark Pejtersen, A. (eds.) HCIS 2010. IFIP AICT, vol. 332, pp. 223–233. Springer, Heidelberg (2010)
10. Ekman, P., Friesen, W.V.: Unmasking the face: A guide to recognizing emotions from facial clues. Prentice-Hall, Oxford (1975)
11. Zhang, G., Li, Z.: The emotion mechanism of artificial life fight behavior. In: Proceedings of the International Conference on Computational Intelligence and Natural Computing, CINC 2009, vol. 1, pp. 15–18 (June 2009)
12. Lv, D., Wang, Y., Guo, C., Pan, Z., Shi, H.: Music-driven emotion model applied in digitalized dance performance of sacrificial ceremony for confucius. In: Pan, Z., Cheok, A.D., Müller, W., Liarokapis, F. (eds.) Transactions on Edutainment IX. LNCS, vol. 7544, pp. 205–212. Springer, Heidelberg (2013)
13. Park, C., Ryu, J., Kang, S., Kim, J., Sohn, J., Cho, H.: The emotion expression robot through the affective interaction: Kobie. In: Proceedings of the 1st International Conference on Robot Communication and Coordination, RoboComm 2007, pp. 53:1–53:4. IEEE Press, Piscataway (2007)
14. Zecca, M., Roccella, S., Carrozza, M.C., Miwa, H., Itoh, K., Cappiello, G., Cabibihan, J.J., Matsumoto, M., Takanobu, H., Dario, P., Takanishi, A.: On the development of the emotion expression humanoid robot WE-4RII with RCH-1, vol. 1, pp. 235–252 (June 2005)
15. Breazeal, C.: Emotion and sociable humanoid robots. International Journal of Human-Computer Studies 59(1-2), 119–155 (2003)
16. Salichs, M.A., Malfaz, M.: A new approach to modeling emotions and their use on a decision-making system for artificial agents. IEEE Transactions on Affective Computing 3(1), 56–68 (2012)
17. Maria, K.A., Zitar, R.A.: Emotional agents: A modeling and an application. Information and Software Technology 49(7), 695–716 (2007)
18. Cai, Z., Goertzel, B., Zhou, C., Zhang, Y., Jiang, M., Yu, G.: Dynamics of a computational affective model inspired by drners {PSI} theory. Cognitive Systems Research 1718, 63–80 (2012)
19. Becker, C., Kopp, S., Wachsmuth, I.: Simulating the emotion dynamics of a multimodal conversational agent. In: André, E., Dybkjær, L., Minker, W., Heisterkamp, P. (eds.) ADS 2004. LNCS (LNAI), vol. 3068, pp. 154–165. Springer, Heidelberg (2004)
20. Raïevsky, C., Michaud, F.: Improving situated agents adaptability using interruption theory of emotions. In: Asada, M., Hallam, J.C.T., Meyer, J.-A., Tani, J. (eds.) SAB 2008. LNCS (LNAI), vol. 5040, pp. 301–310. Springer, Heidelberg (2008)
21. Sanz, R., Snchez-Escribano, M.G., Herrera, C.: A model of emotion as patterned metacontrol. Biologically Inspired Cognitive Architectures 4, 79–97 (2013)
22. Nair, S.B., Godfrey, W.W., Kim, D.H.: On realizing a multi-agent emotion engine. International Journal of Synthetic Emotions (IJSE), 1–27 (2011)
23. Martin, W.A.: Introduction and overview of artificial life evolving intelligent agents for modeling and simulation. In: Proceedings of the 28th Conference on Winter Simulation, pp. 161–168. IEEE Computer Society (1996)

24. Bedau, M.A.: Open problems in artificial life. Artif. Life 6(4), 363–376 (2000)
25. Olivier, M.: An artificial life approach for the synthesis of autonomous agents, pp. 220–231. Springer, London (1996)
26. Sung-Bae, C.: Applications of artificial life to developing robot and softbot. Artificial Life and Robotics 5, 142–147 (2001)
27. Kenji, I., Ikuo, S., Masahito, Y., Masashi, F.: Behavior emergence of a virtual creature living in complex environments. Artificial Life and Robotics 16, 185–189 (2011)
28. Matthias, S.: Useful roles of emotions in artificial agents: a case study from artificial life. In: Proceedings of the 19th National Conference on Artifical Intelligence, AAAI 2004, pp. 42–47. AAAI Press (2004)
29. Nair, S.B., Godfrey, W.W., Kim, D.H.: Towards a dynamic emotional model. In: Proceeding of the IEEE International Symposium on Industrial Electronics, pp. 1932–1936 (2009)
30. Jha, S.S., Naika C.L., S., Nikhra, V., Nair, S.B.: On using emotions in decision-making by situated robots. In: Proceedings of 12th International Conference on Hybrid Intelligent Systems (HIS), pp. 372–377 (December 2012)
31. Cañamero, L.: Emotion understanding from the perspective of autonomous robots research. Neural Networks 18(4), 445–455 (2005)
32. Wilson, S.W.: Knowledge growth in an artificial animal. In: Proceedings of the 1st International Conference on Genetic Algorithms, Hillsdale, NJ, USA, pp. 16–23. L. Erlbaum Associates Inc. (1985)

Land Cover Feature Extraction of Multi-spectral Satellite Images Based on Extended Species Abundance Model of Biogeography

Lavika Goel[1], Daya Gupta[1], and V.K. Panchal[2]

[1] Department of Computer Engineering, Delhi Technological University (DTU), Delhi, India
goel_lavika@yahoo.co.in, dgupta@dce.ac.in
[2] Defense Terrain & Research Lab, Defense & Research Development Organization (DRDO), Metcalfe House, Delhi, India
vkpans@ieee.org

Abstract. This paper presents a land cover feature extraction technique based on the extended species abundance model of biogeography [15, 18] where we consider the HSI as a function of different combinations of SIVs depending upon the characteristics of the habitat under consideration as an extension to the classical BBO [33, 39]. Making use of the proposed hypotheses, we calculate the *HSI* of each of the habitats representing the image pixels using two different functions namely entropy and standard deviation and hence maximize the classification efficiency achieved by adapting to dynamic changes in the HSI function definition. The proposed algorithm has been successfully tested on two different multi-spectral satellite image datasets. We also incorporate the above extended model in our previously designed hybrid bio-inspired intelligent classifier [16] and compare its performance with the original hybrid classifier and twelve other classifiers on the 7-Band Alwar Image.

Keywords: Biogeography, extinction, evolution, remote sensing, feature extraction, kappa coefficient.

1 Introduction

In recent years, many soft computing techniques like fuzzy sets [34], artificial neural network (ANN) [1], rough set theory [37, 38] and swarm intelligence techniques [10, 32] of ant colony optimization (ACO) [4, 8, 9, 36], particle swarm optimization (PSO) [2, 5, 6], genetic algorithms (GA) [19], membrane computing [20] and biogeography based optimization (BBO) [33, 39, 40] etc. have been used for feature extraction or image classification in satellite remote sensing. The swarm intelligence techniques of ACO [4, 8, 9, 36], PSO [2, 5, 6] and BBO [33, 39] and membrane computing [20] are based on the concepts of image clustering and heuristic method implementation and are more accurate when working with low spatial resolution images. Enhanced versions of these techniques have also been used for solving various engineering problems such as the improved PSO called as the immunity enhanced particle swarm optimization (IEPSO) for damage detection [24] wherein a damage detection method based on combined data of static and modal tests using particle swarm optimization (PSO) is discussed. To improve the performance of PSO, some immune properties such as selection, receptor editing and vaccination are introduced into the basic PSO and an improved PSO algorithm is formed. The GA has been modified in the form of the real coded genetic algorithm for target sensing [35] wherein an evolutionary soft-computing technique of real coded genetic algorithm is applied to solve the system of linear equations. The BBO technique has been extended as blended BBO [29] and has been adapted and modified for economic load dispatch analysis [3] to solve both convex and non-convex economic load dispatch (ELD) problems of thermal plants involving constraints such as transmission losses, ramp rate limits, valve point loading, multi-fuel options and prohibited operating zones.

Our proposed work is based on extending the species abundance model of biogeography since the extensions of BBO which have been proposed till date do not consider the factors of evolution and extinction of species in determining the no. of species in a habitat at a time instant and only consider migration as a measure of species count. Hence, we introduce two additional factors, population evolution rate and population extinction rate, besides immigration and emigration in the calculation of the species count, for the characterization of a habitat with a high or a low HSI value. The mix of species and their evolution pattern has significant effect on the physical characteristics of habitat and their contribution to HSI [7]. For example, if a species for which a particular SIV was important is extinct now due to natural evolution or human activity or epidemic, then that particular SIV is no longer relevant to HSI calculation. We extend the original BBO by modeling these factors as several definitions of HSI of a habitat, each definition based on a different set of SIVs which simulates the effect of these other factors. In each iteration of the optimization process, one of these definitions of HSI is chosen for the purpose of HSI calculation, the decision of choice of HSI function being based on these other factors. Making use of the proposed hypotheses, we calculate the *HSI* of each of the habitat representing the image pixels using two different functions namely entropy and standard deviation and assume the. This means that the HSI function which decides the classified feature for each habitat changes based on the classification efficiency that is provided by it and hence, the proposed algorithm adapts to dynamic changes in the HSI (its definition / function) during the optimization process. Hence, we model the factors of evolution and extinction as the phenomenon which make the definition of HSI dynamic [15, 18] and propose the extended model of the biogeography based feature extraction technique as an extension to the original biogeography based feature extraction technique which modeled the HSI function as static [33].

The organization of the paper is as follows: Section 2 presents a brief review of the biogeography based land cover feature extraction technique, the definitions, assumptions, the algorithm and the demonstration of the working of the feature extractor with the HSI function as ‘Standard deviation’ as well as ‘Entropy’ both. Section 3 presents the proposed extended species abundance model of biogeography based feature extraction, the proposed software architecture and the algorithm. Section 4 presents the classification results of the extended species abundance model of biogeography based feature extraction technique on two different datasets and compares its efficiency with the original model by analyzing their kappa coefficients. Also, this section presents a comparison of the proposed classifier with the traditional probabilistic classifiers and other recent soft computing techniques. Section 5 presents the conclusion and future scope of the proposed work.

2 Biogeography Based Land Cover Feature Extraction

BBO is a population based evolutionary algorithm motivated by the migration mechanisms of ecosystems [7, 28, 30]. In BBO each individual solution or a habitat is characterized by a habitat suitability index [39] which is determined by factors called as the suitability index variables. Habitats with a high HSI are characterized by high species count, high species immigration rate, low species emigration rate and dynamic species distribution and vice-versa for habitats with a low HSI value. A good solution is analogous to an island with a high HSI and a poor solution indicates an island with a low HSI [28, 39]. High HSI solutions tend to share their features with low HSI solutions. Low HSI solutions accept a lot of new features from high HSI solutions [28, 39]. Thus, BBO is a swarm intelligence technique based on the concept of information sharing as discussed in our paper [14]. From our paper, we know that the concepts of sharing information in BBO can be adapted to suit to the problem of land cover feature extraction where it has been mentioned that BBO is well suited for natural terrain feature elicitation application and hence we modify the original BBO and adapt it to the problem of feature extraction. This section describes the details of the biogeography based land cover feature extraction process –the parameters settings and the algorithm used. Fig. 1 presents the algorithm for biogeography based land cover feature extraction [33].

```
Input –Multi-spectral satellite image
Output – Classified image
Classify image in elementary classes using fuzzy c- means and consider them
as species of the universal habitat. Consider each feature as one habitat.
Ecosystem = Total No. of Habitats = Universal Habitat + Feature Habitat
Define HSI, $S_{max}$, $S_{min}$, immigration rate and emigration rate.
Calculate HSI for each feature habitat.
Select species from universal habitat and migrate it to one of the other habitat
and recalculate HSI.
If recalculated HSI is within threshold then absorb the species to that habitat
else.
Check for the other habitats and recalculate the HSI.
If all species in universal habitat are checked then stop else go to step 4.
```

Fig. 1. Algorithm for biogeography based land cover feature extraction

The parameters which define the pre-processing steps of the biogeography based land cover feature extraction algorithm [33] are summarized below:

1. It is assumed that the species S_i refer to the image pixels of the Image I.

$$I = \{S_i \mid \forall\, i \in [1...(\text{Size of image I})]\} \tag{1}$$

2. An ecosystem is considered to be comprised of N habitats, H_i [33], the Universal Habitat H_u which hosts all the image species such that

$$H_u = \cup_{i=1}^{N} H_i \tag{2}$$

and the feature habitats (H_f) which are the labeled dataset/training dataset generated by the expert.

3. Each of the multispectral band of image represents one suitability index variable (SIV) of the habitat. Further, image in each band is a gray image; therefore, SIV$\in$ C is an integer and C $\subseteq$ [0,255].

4. A habitat H $\in$ SIV_m where, **m** is the total image bands.

5. The image *I* is subjected to 'n' simple partitions P_i. The image may be partitioned into any number as per the individual preference.

$$I = \{P_i \ldots P_n\} \tag{3}$$

Each P_i is subjected to unsupervised clustering so that based on some criteria derived from the P_i statistics, new smaller clusters are generated. The rough sets based clustering [37, 38] is performed to P_i, and new, unsupervised, clusters are generated. These are now known as the new habitats H_i.

6. For application of BBO in a classification problem, it was found appropriate to choose either entropy or the standard Deviation as the HSI functions. For the calculation of entropy, we follow the procedure described next [12]. We copy the pixels of the species habitat (rough sets generated equivalence classes) for which the entropy has to be calculated, in an excel sheet and then divide the digital number (DN) values in the considered band into 'm' intervals. Next, we calculate the no. of pixels of the entire species habitat H_i falling in each of the 'm' ranges obtained above and hence, calculate the entropy of the particular habitat by using the equation below.

$$\mathrm{Entropy}(E_k) = -\sum_{k=1}^{m} P_k \log (P_k) \tag{4}$$

where, $P_k = \frac{m_k}{N}$ where m_k is the no. of pixels in the k^th interval and *N* is the total no. of pixels in the species habitat.

For the HSI calculation using standard deviation or entropy functions, we proceed as explained next [21]. Let the average of the standard deviation / entropy of the species habitat C_i (or rough sets generated equivalence classes) from H_i of the Universal Habitat **Π** be represented by:

$$HSI\left((Standard\ Deviation/Entropy)_{H_i}\right) = \sigma_{c_i} = \frac{\left(\sigma_{1_i} + \sigma_{2_i} + \sigma_{3_i} + \sigma_{4_i} + \ldots\ldots + \sigma_{m_i}\right)}{m} \tag{5}$$

where, $\sigma_{1_i}, \sigma_{2_i}, \sigma_{3_i}, \sigma_{4_i}, \ldots\ldots, \sigma_{m_i}$ are the standard deviations / entropy of the DN values of each of the m bands respectively of the i^th species habitat in the Universal habitat H_u. The standard deviation / entropy calculations for each habitat H_j in H_f can be performed similarly. Let σ_{F_j}, be the average of the standard deviations / entropies of any of the feature islands.

$$HSI\left((Standard\ Deviation/Entropy)_{H_f}\right) = \sigma_{F_j} = \frac{\left(\sigma_{1_j} + \sigma_{2_j} + \sigma_{3_j} + \sigma_{4_j} + \ldots\ldots + \sigma_{m_j}\right)}{m} \tag{6}$$

$$HSI\left((Standard\ Deviation/Entropy)_{H_i}\right) - HSI\left((Standard\ Deviation/Entropy)_{H_f}\right) = |\sigma_{c_i} - \sigma_{F_j}| \tag{7}$$

3 Land Cover Feature Extraction Based on Extended Species Abundance Model of Biogeography

This section presents the extended model of BBO by considering the fact that if we consider a standard definition of HSI, then some SIVs in it may be added or removed analogous to the species evolution and extinction on a habitat. This can be achieved by making the coefficient of the extinct SIV in the standard HSI function definition (containing all the SIVs) as zero and the evolved SIV as one. For the purpose, we calculate the factor of relevance of a particular SIV in the HSI calculation on different habitats and eliminate the SIVs for which the relevance factor value is below 50%. We extend the original BBO by modeling these factors as several definitions of HSI of a habitat and in each iteration of the optimization process, one of these definitions of HSI is chosen for the purpose of HSI calculation [15]. We therefore propose the following extended model of biogeography based algorithm in Fig. 2. in order to extract land cover features from the satellite image as an extension to the original biogeography based feature extraction algorithm [33]. The input to the proposed algorithm is the multi-spectral satellite image and the output is the extracted features from the image. The proposed architectural framework for our extended species abundance model of biogeography based land cover feature extractor is presented in Fig. 3 below and explained layer by layer next. The flowchart for the extended biogeography based feature extraction algorithm is presented in Fig. 4. The above layers of the software architecture of our proposed feature extractor (or classifier) are explained step by step below:

Input Layer: A high resolution multi-spectral satellite image is taken as input. For our illustration, we used two different datasets that of the 7-band cartoset satellite image of size 472 X 546 of the Alwar Region in Rajasthan and the 4-band cartoset satellite image of size 472 X 576 of the Patalganga region in Himachal Pradesh.
Ecosystem Initialization Layer: This layer is responsible for the initialization of the ecosystem for generating the input equivalence classes initialize our proposed extended biogeography based feature extractor. The sub-layers are described next.
(a) Image Grid Division Layer: In this layer, the satellite image is divided into n clusters.
(b) Rough Sets Theory Applier: For the purpose, we use the rough sets part of the biogeography based land cover feature extraction algorithm [31, 33] (described in section 2) which uses multiple bands of the image (which represent the texture of each of the land cover features in the image clearly, hence, clustering is based on texture analysis) during the discretization and partitioning step [37, 38] to generate the unsupervised clusters called the equivalence classes. In the initial state of the ecosystem required to initialize our extended biogeography based feature extractor, we consider each of these equivalence classes as a member of the Universal habitat, 'H_u', where 'j' ranges from 1 to 'n' where 'n' is the total no. of members in 'H_u'.
(c) Band Selector: Depending on our application, in other words depending on which feature we want to extract from the image most efficiently, we can choose the band for partitioning [11]. We have chosen the NIR and MIR bands of the 7-band image

since these bands contain a good amount of geo-spatial information and these are the bands in which the features are particularly more highlighted and best viewed [11].

(d) Discretizor and Partitioner: Therefore, we use the NIR and the MIR bands for discretization and partitioning step in the semi –naïve algorithm used for creating rough set equivalence classes for each of the clusters. Each of these resultant classes are put in the universal habitat.

Surface Entropy / Standard Deviation Representation Layer: In the extended model of original BBO for land cover feature extraction, we consider the fact that the HSI function can vary dynamically during optimization [15]. Making use of the proposed hypotheses, we calculate the HSI of each of the habitat H_i in the Universal habitat H_u using two different functions namely entropy and standard deviation and hence define the HSI function definition as {Entropy, Standard deviation} such that the feature index decided by the function chosen achieves a smaller percentage difference value. The sub-layers are described below:

(a) HSI Based on Average Surface Entropy/ Average Standard Deviation Calculator for Species Habitat: The HSI calculation procedure using standard deviation and entropy functions for the species habitat is the same as described in section 2 [33].

(b) Difference HSI with the Feature Habitat ($\mathbf{H_f}$) Calculator: The standard deviation and entropy calculations for each habitat H_j in H_f can be performed similarly as explained in section 2. The difference in the HSI of the species habitat (σ_{c_i}) with the feature habitat (σ_{F_j}) is given by equation (7).

Dynamic HSI Function Selection Layer: This layer is responsible for the selection of the suitable HSI function dynamically with each iteration. The HSI function which decides the classified feature for each species habitat H_i changes (evolves or becomes extinct) based on the classification efficiency that is provided by it. The detailed procedure for the dynamic HSI function modeler layer is described below.

(a) Feature Habitat with Minimum Difference HSI Estimator: After calculating the difference between the HSI of the selected species H_i and each of the feature habitats H_j based on entropy and standard deviation separately, this layer is responsible for finding out the feature habitat with which H_i has the minimum difference i.e. H_j for which $|\sigma_{c_i} - \sigma_{F_j}|$ is minimum for both entropy and standard deviation.

(b) Percentage Difference Matrix Calculator Based on Entropy and Standard Deviation: This layer is responsible for calculating the percentage of this minimum difference value with the HSI of the species habitat H_i based on entropy and standard deviation separately and is given by the following expression.

$$\frac{min\left(HSI_{entropy\,/\,stddev}\,(H_i) - HSI_{entropy\,/\,stddev}\,(H_j)\right)}{HSI_{entropy}\,/\,stddev\,(H_i))} \times 100 \tag{8}$$

(c) Normalized Percentage Difference Matrices Comparator: This layer compares the percentage difference matrices based on entropy and standard deviation. The output percentage difference values are dynamically decided based on the HSI function chosen for the equivalence class under consideration. For comparing the percentage difference matrices, the values are normalized so that both the HSI function based on entropy return the percentage difference values that are scaled in the interval of the percentage difference values returned by the HSI function based on standard deviation. Scaling factor for each of the table entries i.e. for the percentage difference values for each equivalence class used is defined as below:

$$Scaling\ factor_i = \frac{Max(Percentage\ diff(Stddev)) - Min(Percentage\ diff(Stddev))}{Max(Percentage\ diff(Entropy)) - Min(Percentage\ diff(Entropy))} \times Percentage\ diff_i\ (Entropy) \tag{9}$$

where i represents the ith equivalence class.

Feature Extractor Layer: This layer is responsible for feature extraction based on dynamic HSI function modeling.

(a) Resultant Percentage Difference Matrix Calculator: This layer constructs the matrix of the output percentage difference values which will be mapped to the corresponding feature indices which will form the classified image. The function (entropy or standard deviation) which yields smaller percentage difference value is chosen as the HSI function.

(b) Threshold Condition Satisfier: This layer checks if the HSI of the species habitat H_i for which the percentage difference is minimum is within the threshold 'δ' in order for the input equivalence class to be classified into the corresponding feature index. If the HSI calculated is within the threshold, then the input class is classified into the corresponding feature and removed from 'H_u' otherwise, we further split the input class which remained in 'H_u' (since it could not be classified) through rough sets and repeat the above procedure for the input equivalence class left in the Universal habitat. Initialize a new ecosystem for the classification of the remaining classes in 'H_u'.

(c) Feature Index Assignor: This layer is responsible for assigning the feature index of the HSI function which has the smaller percentage difference value. These feature index values are the values into which the equivalence class is classified into finally based on the dynamically decided HSI function.

(d) Color Codes Assignment Layer: Finally color codes are assigned for each pixel of the image corresponding to the classified feature of each pixel.

Output Layer: Final classified image is obtained in .jpeg, .tiff or any other image format.

4 Results and Discussion

Next we discuss the applications of the proposed algorithm for land cover feature extraction using two different datasets of Alwar region in Rajasthan and Patalganga region in Shivalik ranges. The results and analyses for each of the case studies are described in the sections below.

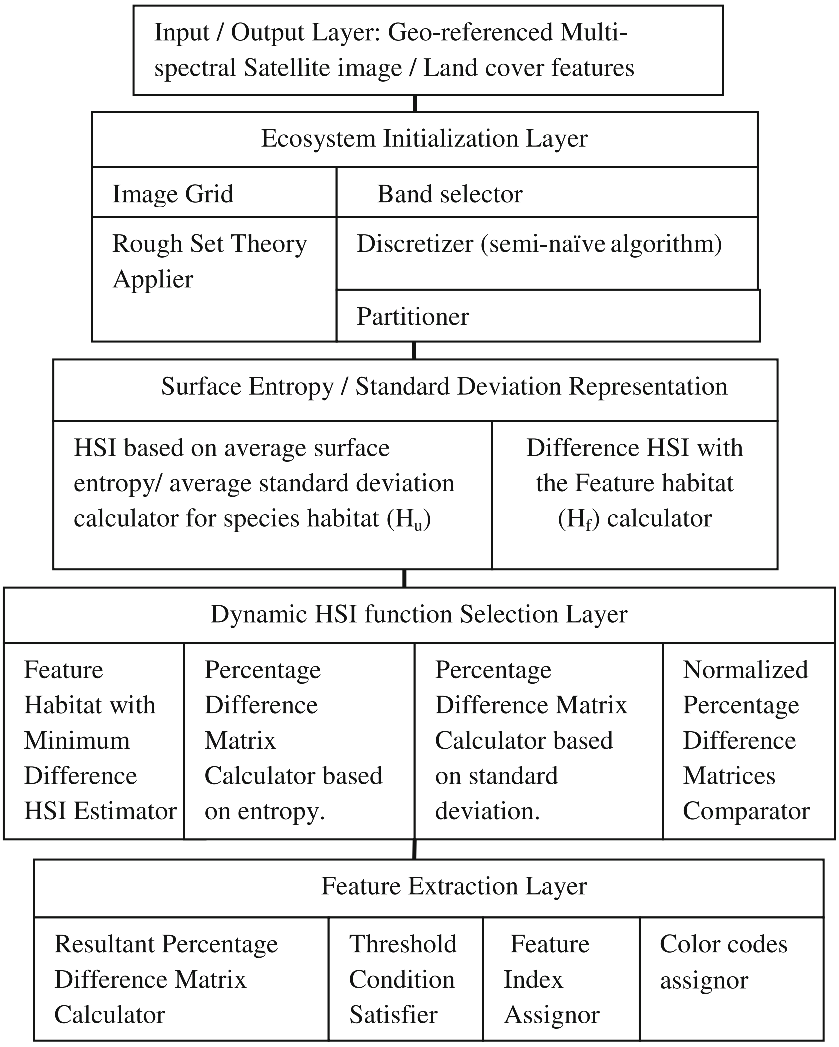

Input / Output Layer: Geo-referenced Multi-spectral Satellite image / Land cover features
Ecosystem Initialization Layer
Image Grid
Band selector
Rough Set Theory Applier
Discretizer (semi-naïve algorithm)
Partitioner
Surface Entropy / Standard Deviation Representation
HSI based on average surface entropy/ average standard deviation calculator for species habitat (H_u)
Difference HSI with the Feature habitat (H_f) calculator
Dynamic HSI function Selection Layer
Feature Habitat with Minimum Difference HSI Estimator
Percentage Difference Matrix Calculator based on entropy.
Percentage Difference Matrix Calculator based on standard deviation.
Normalized Percentage Difference Matrices Comparator
Feature Extraction Layer
Resultant Percentage Difference Matrix Calculator
Threshold Condition Satisfier
Feature Index Assignor
Color codes assignor

Fig. 2. Software architecture of the proposed land cover feature extractor based on the extended species abundance model of biogeography

4.1 Case Study of Alwar Region in Rajasthan

We have used a multi-spectral, multi resolution and multi sensor image of Alwar area in Rajasthan with dimensions 472 X 576. The satellite image for seven different bands is taken. The Bands are Red, Green, Near Infra-Red (NIR), Middle Infra-Red (MIR), Radarsat-1 (RS1), Radarsat-2 (RS2) and Digital Elevation Model (DEM). The 7-Band satellite image of Alwar area in Rajasthan is shown in Fig. 5 is taken [25].

```
Begin
/* BBO ecosystem initialization */
Create a Universal habitat consisting of the rough set generated equivalence classes also
called as the species (population) $H_1, H_2, \ldots, H_n$ which comprises $H_u$;
Create a Feature habitat consisting of the expert generated training set for each of the land
cover features to be extracted $H_f$;
/*Compute the corresponding $HSI$ values */
While $(i \leq n)$
  Compute the $HSI$ for each species $H_i$ in $H_u$ based on the corresponding function $f_i(HSI)$
where
   $\mathrm{f_i(HSI) = \{Entropy, Standard\ Deviation\}}$;
  Also, calculate the HSI for each of the feature habitat $\mathrm{H_j}$.
 End while
/* End of BBO parameter initialization */
While not $\mathrm{T\ (H_u\ != NULL)}$ do /* T is the termination criterion */
For each species $\mathrm{H_i}$ in the Universal habitat $\mathrm{H_u}$
  Select a species from $\mathrm{H_u}$
  For each of the feature habitat $\mathrm{H_j}$.
    Migrate the specie $\mathrm{H_i}$ to $\mathrm{H_j}$.
    Recalculate the HSI of the feature habitat $\mathrm{H_j}$ after the migration of the species $\mathrm{H_i}$ to it
based on entropy and standard deviation separately.
End for
 Calculate the percentage of the minimum difference obtained between the species $\mathrm{H_i}$ and
the feature habitat $\mathrm{H_j}$ based on entropy and standard deviation both.
 If  $\mathrm{\%\ Difference_{entropy}\ \ \%\ Difference_{std\ deviation}}$
          $\mathrm{f_i(HSI) = \{Entropy\}}$;  $\mathrm{Feature_{Index}(f_i(HSI)) = Feature_{Index}(Entropy)}$;
 Else
          $\mathrm{f_i(HSI) = \{Standard\ Deviation\}}$;
$\mathrm{Feature_{Index}(f_i(HSI)) = Feature_{Index}(Standard\ Deviation)}$;
End
Absorb the species $H_i$ in the feature habitat $H_j$.
End for.
End while.
End
```

Fig. 3. Algorithm for land cover feature extraction based on the extended species abundance model of biogeography

We calculate the percentage difference values for the average entropies and the average standard deviations in the multi-spectral bands for each rough sets generated equivalence class with the equivalence class with which it has minimum difference with in each of the

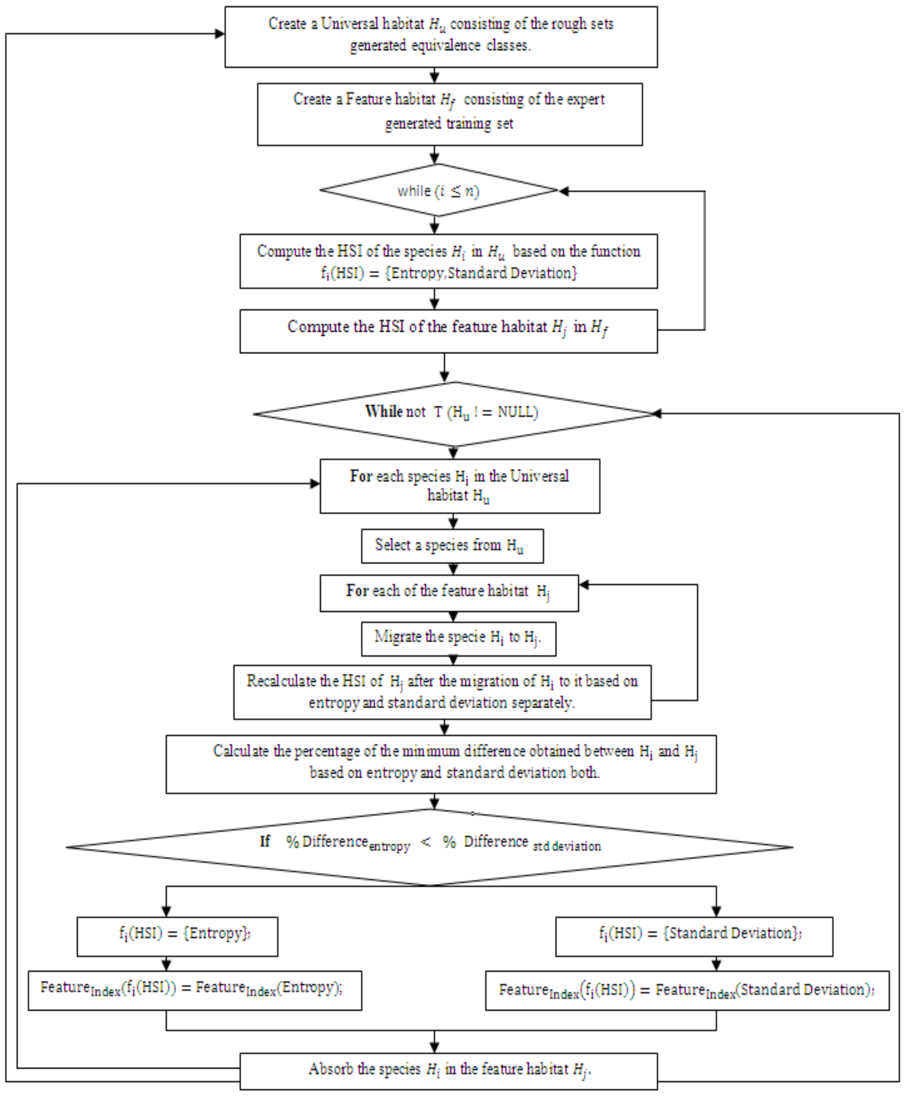

Create a Universal habitat H_u consisting of the rough sets generated equivalence classes.
Create a Feature habitat H_f consisting of the expert generated training set
while ($i \le n$)
Compute the HSI of the species H_i in H_u based on the function f_i(HSI) = {Entropy,Standard Deviation}
Compute the HSI of the feature habitat H_j in H_f
While not T (H_u != NULL)
For each species H_i in the Universal habitat H_u
Select a species from H_u
For each of the feature habitat H_j
Migrate the specie H_i to H_j.
Recalculate the HSI of H_j after the migration of H_i to it based on entropy and standard deviation separately.
Calculate the percentage of the minimum difference obtained between H_i and H_j based on entropy and standard deviation both.
If % Difference$_{entropy}$ < % Difference$_{std\ deviation}$
f_i(HSI) = {Entropy};
f_i(HSI) = {Standard Deviation};
Feature$_{Index}$(f_i(HSI)) = Feature$_{Index}$(Entropy);
Feature$_{Index}$(f_i(HSI)) = Feature$_{Index}$(Standard Deviation);
Absorb the species H_i in the feature habitat H_j.

Fig. 4. Flowchart representing the mechanism of biogeography based feature extraction based on the extended species abundance model

'n' (n = 20) partitions of the image after the values have been normalized and scaled respectively (using equation (9)).

4.2 Case study: Alwar Region in Rajasthan

We calculate hence the output percentage difference values which will be mapped to the corresponding feature indices to form the classified image based on the

Table 1. Resultant Percentage Difference matrix based on the dynamically decided HSI function for the Alwar image

Result % difference matrix (based on dynamic HSI function) for equivalence class (i) / Partition (z)	1	2	3	4	5	6	7	8	9
1	3.1387	0.3156	2.6645	2.3874	1.0133	0.8855	1.4097	0.3414	0.3197
2	2.5095	0.5967	3.1925	2.1780	1.2364	1.4634	1.3080	0.7829	0.6457
3	3.3650	1.0817	2.4894	2.2944	2.1934	1.1082	0.0010	0.4121	0.5788
4	3.4060	2.7764	1.4383	2.7176	2.4378	3.0953	1.4464	0.8451	1.1230
5	2.9662	0.6659	1.2697	0.4937	3.1466	2.3419	1.7159	1.0119	0.8046
6	2.7015	1.7846	0.7089	1.7502	2.5295	0.4444	0.0433	2.4822	2.2536
7	0.9142	0.4377	0.9412	0.2463	1.5018	3.2462	0.0780	3.1369	1.0562
8	0.3304	0.9013	3.0350	1.4279	1.9940	3.0107	0.0231	0.7091	1.8127
9	0.4167	0.7449	4.0645	0.8191	1.1809	2.0885	1.3138	1.1681	1.0605
10	2.8680	1.4106	1.2324	1.0254	2.4472	2.1280	1.3375	1.0379	1.1226
11	1.2879	0.3028	2.9969	1.9558	2.8283	2.8712	3.5600	2.0904	0.0000
12	0.0480	0.8665	0.0384	1.4400	2.3192	0.3045	0.0149	3.4109	0.9663
13	4.7319	1.7193	4.0712	2.6879	2.1738	2.7199	0.2169	2.6836	1.0931
14	0.3967	1.4640	3.9184	0.5054	1.2255	1.0397	0.4773	0.6835	1.0674
15	2.7057	2.5381	1.7159	1.3473	0.1594	0.1044	0.7475	1.2777	1.8372
16	1.8155	1.4522	1.5695	1.0300	3.2664	3.8690	3.1855	2.1780	0.0000
17	1.5321	0.2648	2.5766	2.3653	2.3758	3.1161	0.0019	0.2038	3.3854
18	3.6384	3.4229	0.4210	3.0027	2.7517	2.8416	3.8026	1.0664	0.9535
19	1.9808	0.4728	1.1273	3.5561	2.3165	1.5299	1.3880	3.1042	1.6254
20	4.2054	3.6353	1.4102	0.4450	2.0559	0.1914	0.5930	2.3053	1.2168

dynamically decided HSI function. Fig. 6 presents the bar graph representing the percentage difference matrix based on HSI function ‘Entropy’ and ‘Standard Deviation’ for the Alwar Image. The color codes are red for entropy and blue for standard deviation. From the graphs, it can be observed that there is a significant difference between the percentage difference matrix plot of entropy and that of standard deviation for most of the equivalence classes and hence, the accuracy provided by these functions may vary significantly on these equivalence classes. This means that the classification accuracy of the HSI function entropy might be significantly less than the classification accuracy provided by the HSI function standard deviation and vice versa. Hence, our dynamic functioning in BBO is guaranteed to improve the classification accuracy provided by either of the two HSI functions used independently which is what is reflected and also verified from the classification results of Alwar too. Table 1 presents the table of the output percentage difference values which will be mapped to the corresponding feature indices which will form the classified image. Table 2 represents the final feature index values for each of the equivalence classes based on the dynamically decided HSI function. The feature index codes are 1-Barren, 2-Rocky, 3-Urban, 4-Vegetation and 5-Waterbody. Table 3 carries the information about the HSI function that was chosen for each of the equivalence classes based on the criteria of maximization of classification accuracy at runtime. The HSI function codes are 1 is for entropy and 2 is for standard deviation.

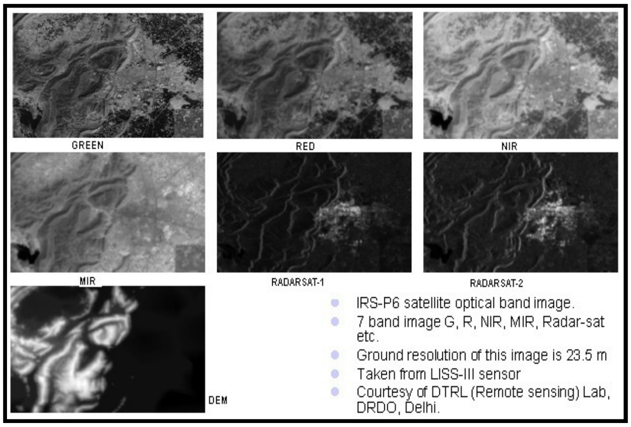

GREEN
RED
NIR
MIR
RADARSAT-1
RADARSAT-2
DEM
IRS-P6 satellite optical band image.
7 band image G, R, NIR, MIR, Radar-sat etc.
Ground resolution of this image is 23.5 m
Taken from LISS-III sensor
Courtesy of DTRL (Remote sensing) Lab, DRDO, Delhi.

Fig. 5. 7-band satellite image of Alwar region in Rajasthan

The classified image is obtained in Fig. 9 (Kappa Coefficient = 0.8271) which clearly shows the extraction of land cover features of water, urban, rocky, vegetation and barren. The yellow color represents rocky area, green color represents vegetation area, black color represents barren area and red color represents the urban area. The classified image through BBO with the HSI function as ‘Standard Deviation’ [33] is shown in Fig. 7 which has a kappa coefficient of 0.6751 and that with the HSI function as ‘Entropy’ [13] is shown in Fig. 8 which has a kappa coefficient of 0.7440 which reflects that the proposed algorithm performs better than the original BBO used for land cover feature extraction [33]. From the Figs. 7, 8 and 9, it can also be seen that the encircled urban region in grey which was wrongly classified as mixture of barren with urban by the original BBO with HSI function ‘Standard deviation’ as well as the original BBO with the HSI as entropy, has been classified correctly by our proposed classifier. Also, the encircled barren region in pink which was wrongly classified as urban by the original BBO with HSI function ‘Standard deviation’ and as mixture of urban with barren by the original BBO with the HSI as entropy, has been classified correctly by our proposed classifier. Also, the encircled urban region in blue which was wrongly classified by the original BBO (with HSI function 'Standard deviation') as barren has been classified correctly by original BBO with entropy as the HSI function as well as by our proposed extended biogeography based classifier. Tables 4 and 5 present the error matrices for the original BBO classifier with the HSI function 'Standard deviation' and the HSI function 'Entropy' respectively [12, 13, 33]. Table 6 presents the error matrix for the proposed biogeography based classifier.

Table 2. Classified feature index matrix based on dynamically decided HSI function for the Alwar image. The feature index codes are 1-Barren, 2-Rocky, 3-Urban, 4-Vegetation and 5-Waterbody.

Classified Feature Index based on dynamic HSI function for equivalence class (i) / Partition (z)	1	2	3	4	5	6	7	8	9
1	2	1	1	1	1	1	1	1	1
2	2	1	4	1	3	1	1	1	1
3	2	2	4	2	1	1	1	1	1
4	4	4	4	3	3	4	1	1	1
5	4	4	4	1	4	4	3	3	1
6	2	2	2	2	2	2	2	2	1
7	2	2	2	2	2	3	2	3	1
8	2	2	4	2	2	3	2	2	3
9	1	1	4	3	3	3	1	1	1
10	4	4	4	3	1	4	1	1	1
11	2	2	4	2	2	1	2	1	0
12	2	2	2	2	2	2	2	2	2
13	2	3	3	2	3	3	2	3	1
14	1	1	4	3	3	1	3	3	3
15	1	4	4	1	3	1	1	1	1
16	5	2	4	2	2	3	3	3	0
17	2	2	2	2	2	2	2	2	3
18	2	4	4	3	1	1	3	1	1
19	4	4	4	4	4	4	3	3	1
20	3	3	4	3	3	1	1	3	1

Table 3. HSI functions chosen dynamically for each equivalence class in the Alwar image. The HSI function codes are 1-Entropy and 2-Standard Deviation.

Dynamic HSI function for equivalence class (i) / Partition (z)	1	2	3	4	5	6	7	8	9
1	2	2	1	1	1	1	1	2	2
2	2	2	1	1	2	1	1	1	1
3	1	2	1	2	2	1	2	2	1
4	1	1	1	1	1	1	1	1	1
5	1	2	1	2	1	1	1	2	2
6	2	2	2	2	1	2	2	2	1
7	1	2	2	2	2	1	2	1	2
8	2	2	1	2	1	1	2	2	1
9	2	2	1	1	1	2	1	1	1
10	1	2	1	2	2	1	1	1	2
11	1	2	1	1	1	1	1	1	2
12	2	2	2	2	1	2	2	1	2
13	1	1	1	1	1	2	2	1	1
14	2	2	1	1	1	2	1	1	1
15	2	1	1	1	2	2	1	1	1
16	1	2	1	2	2	1	1	1	2
17	1	2	2	2	1	2	2	2	1
18	1	1	2	1	2	2	1	1	1
19	1	2	1	1	1	1	1	1	2
20	1	1	1	2	2	2	2	1	1

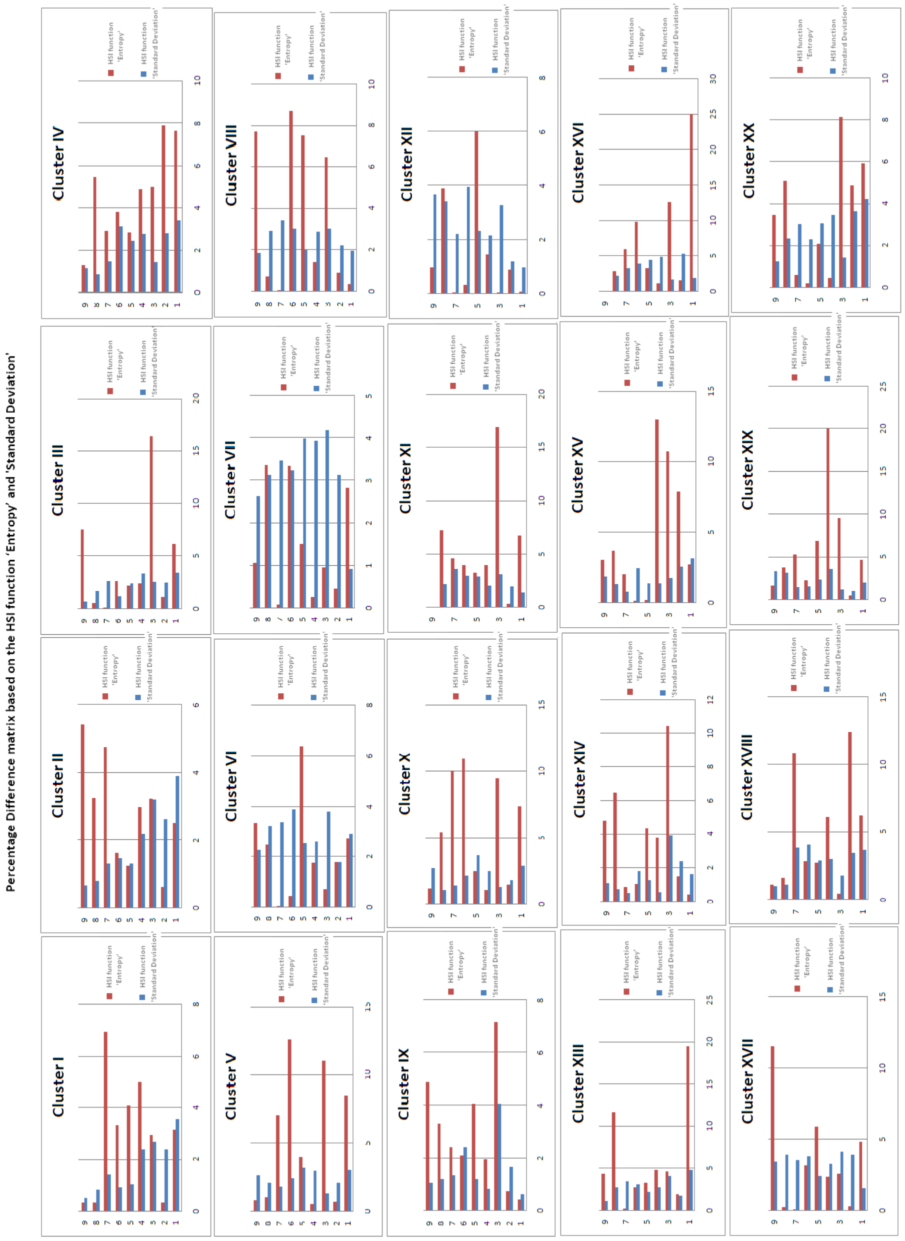
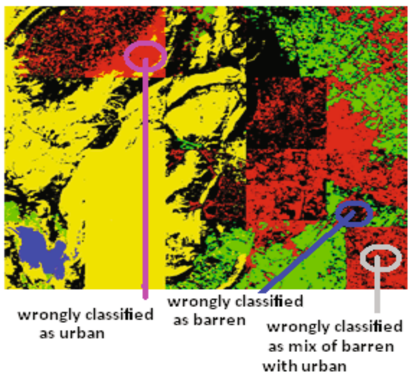


Fig. 6. Bar graph representing the percentage Difference Matrix based on HSI function 'Entropy' and 'Standard Deviation' for the Alwar Image. Color codes are: Red- Entropy and Blue-Standard Deviation.

Table 7 represents the producer's accuracy of classification from which it is reflected that water pixels have been extracted most efficiently followed by barren and vegetation pixels, respectively. Table 8 represents the user's accuracy of classification from which it is observed that water shows 100% accuracy of classification followed by vegetation pixels. From the error matrix, the KHAT statistics of the extended biogeography based land cover feature extractor is calculated as 0.8271. This value is a substantial agreement between the ideal classifier and the proposed algorithm. The kappa (K) coefficient of the Alwar image is 0.8271 which indicates that an observed classification is 82.71% better than one resulting from chance.


wrongly classified as urban
wrongly classified as barren
wrongly classified as mix of barren with urban

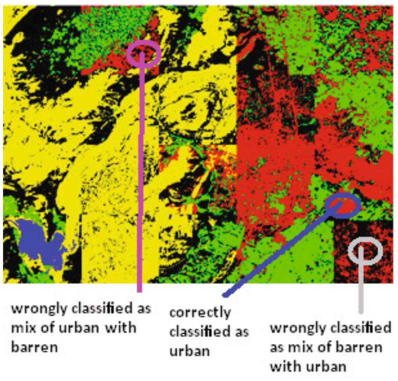

wrongly classified as mix of urban with barren
correctly classified as urban
wrongly classified as mix of barren with urban

Fig. 7. Final Classified image of Alwar after applying BBO for land cover feature extraction with the HSI function 'Standard Deviation'. (Kappa Coefficient = 0.6751) [33]

Fig. 8. Final Classified image of Alwar after applying original BBO for land cover feature extraction with HSI function 'Entropy'. (Kappa Coefficient = 0.7440) [13]

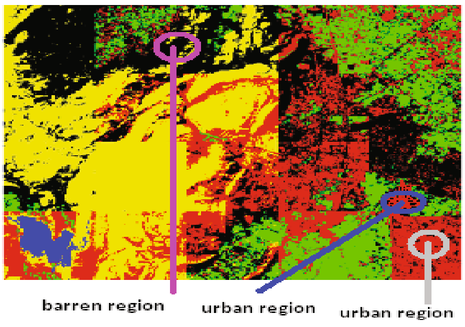


Fig. 9. Final Classified image of Alwar after applying extended species abundance model of biogeography based land cover feature extraction (Kappa Coefficient = 0.8271)

The classified image is obtained in Fig. 9 (Kappa Coefficient = 0.8271) which clearly shows the extraction of land cover features of water, urban, rocky, vegetation and barren. The yellow color represents rocky area, green color represents vegetation area, black color represents barren area and red color represents the urban area. The classified image through BBO with the HSI function as 'Standard Deviation' [33] is shown in Fig. 7 which has a kappa coefficient of 0.6751 and that with the HSI function as 'Entropy' [13] is shown in Fig. 8 which has a kappa coefficient of 0.7440 which reflects that the proposed algorithm performs better than the original BBO used for land cover feature extraction [33]. From the Figs. 7, 8 and 9, it can also be seen that the encircled urban region in grey which was wrongly classified as mixture of barren with urban by the original BBO with HSI function 'Standard deviation' as well as the original BBO with the HSI as entropy, has been classified correctly by our proposed classifier. Also, the encircled barren region in pink which was wrongly classified as urban by the original BBO with HSI function 'Standard deviation' and as mixture of urban with barren by the original BBO with the HSI as entropy, has been classified correctly by our proposed classifier. Also, the encircled urban region in blue which was wrongly classified by the original BBO (with HSI function 'Standard deviation') as barren has been classified correctly by original BBO with entropy as the HSI function as well as by our proposed extended biogeography based classifier.
Tables 4 and 5 present the error matrices for the original BBO classifier with the HSI function 'Standard deviation' and the HSI function 'Entropy' respectively [12, 13, 33]. Table 6 presents the error matrix for the proposed biogeography based classifier. Table 7 represents the producer's accuracy of classification from which it is reflected that water pixels have been extracted most efficiently followed by barren and vegetation pixels, respectively. Table 8 represents the user's accuracy of classification from which it is observed that water shows 100% accuracy of classification followed by vegetation pixels. From the error matrix, the KHAT statistics of the extended biogeography based land cover feature extractor is calculated as 0.8271. This value is a substantial agreement between the ideal classifier and the proposed algorithm. The kappa (K) coefficient of the Alwar image is 0.8271 which indicates that an observed classification is 82.71% better than one resulting from chance.

Table 4. Error matrix of BBO with the HSI function 'Standard deviation' on Alwar region. (Kappa Coefficient = 0.6715) [33]

	Vegetation	Urban	Rocky	Water	Barren	Total
Vegetation	127	9	0	0	2	138
Urban	0	88	1	0	32	121
Rocky	6	2	176	1	17	202
Water	0	0	3	69	0	72
Barren	17	91	20	0	119	247
Total	150	190	200	70	170	780

Table 5. Error matrix of original BBO with the HSI function 'Entropy' on Alwar region. (Kappa Coefficient = 0.7440) [13]

	Vegetation	Urban	Rocky	Water	Barren	Total
Vegetation	142	0	0	0	1	143
Urban	3	66	8	0	4	81
Rocky	0	14	185	1	4	204
Water	0	0	0	69	0	69
Barren	5	111	6	0	161	283
Total	150	190	200	70	170	780

Table 6. Error matrix of the extended model of biogeography based feature extractor on Alwar region. (Kappa Coefficient = 0.8271)

	Vegetation	Urban	Rocky	Water	Barren	Total
Vegetation	145	2	0	0	37	184
Urban	0	151	7	0	6	164
Rocky	0	10	182	1	2	194
Water	0	0	0	69	0	70
Barren	4	27	10	0	121	162
Total	149	190	199	70	166	774

Table 7. Producer's Accuracy

Feature	Accuracy Calculation	Producer's Accuracy
Vegetation	145/149	97.3%
Urban	151/190	79.5%
Rocky	182/199	91.5%
Water	69/70	98.7%
Barren	121/166	72.9%

Table 8. User's Accuracy

Feature	Accuracy Calculation	User's Accuracy
Vegetation	145/184	78.8%
Urban	151/164	92.1%
Rocky	182/194	93.8%
Water	69/70	98.7%
Barren	121/162	74.7%

4.3 Case Study: Patalganga Region in Shivalik Ranges

For Patalganga, area in Shivalik mountainous ranges is undertaken. We have used 4 band images of Patalganga taken from Landsat-I satellite with dimensions 508 X 744. The bands are Red, Green, Near Infra-Red (NIR) and Middle Infra-Red (MIR) [25]. The 4-Band satellite image of Patalganga area in Shivalik ranges is shown in Fig. 10 is taken.

The image is subjected to 'n' (here, we take $n = 15$) simple partitions, for the sake of simplicity. Table 9 presents the table of the output percentage difference values which will be mapped to the corresponding feature indices to form the classified image [12, 13, 33]. Table 10 represents the final feature index values for each of the equivalence classes. The feature index codes are 1-Rocky, 2-Snow and 3-Vegetation. Table 11 carries the information about the HSI function that was chosen for each of the equivalence classes based on the criteria of maximization of classification accuracy at runtime. The HSI function codes are 1 is for entropy and 2 is for standard deviation. Fig. 11 presents the bar graph representing the percentage difference matrix based on HSI function 'Entropy' and 'Standard Deviation' for the Patalganga Image. The color codes are red for entropy and blue for standard deviation.

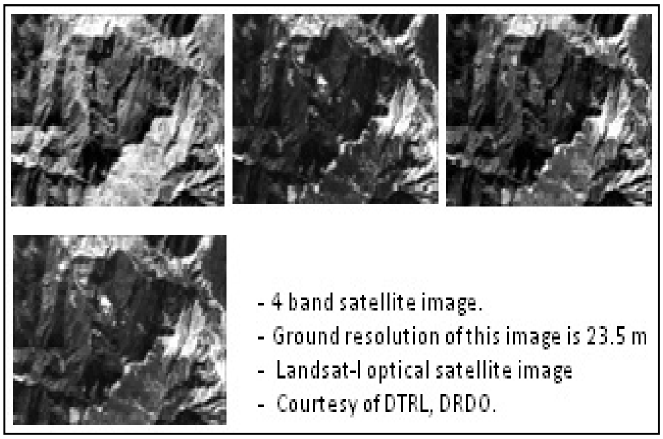

- 4 band satellite image.
- Ground resolution of this image is 23.5 m
- Landsat-I optical satellite image
- Courtesy of DTRL, DRDO.

Fig. 10. 4 band image of Patalganga

The classified image is obtained in Fig. 14 (Kappa Coefficient = 0.8014) which clearly shows the extraction of land cover features of snow, vegetation and rocky area. The yellow color represents rocky area, green color represents vegetation area, and red color represents the snowy area. The classified image through original BBO with the HSI function 'Standard deviation' [33] is shown in Fig. 12 which has a kappa coefficient of 0.7248 and that with the HSI function as 'Entropy' [12, 13] is shown in Fig. 13 which has a kappa coefficient of 0. 7870 which reflects that the proposed algorithm performs better than the original BBO used for land cover feature extraction [33]. From the Figs. 12, 13 and 14, it can also be seen that the encircled snowy region in black which was wrongly classified as rocky by the original BBO with HSI function 'Standard deviation' and as the mixture of rocky with snowy by the original BBO with the HSI as entropy, has been classified correctly by our proposed classifier. Also, the encircled vegetation region in green which was wrongly classified as mixture of rocky with vegetation by the original BBO with HSI function ‘Standard deviation’ and as rocky region by the original BBO with the HSI as entropy, has been classified correctly by our proposed classifier. However, the encircled rocky region in blue which was wrongly classified by the original BBO (with HSI function ‘Standard deviation’) as snowy has been classified correctly by original BBO with entropy as the HSI function as well as by our classifier.

Table 9. Resultant Percentage Difference matrix based on the dynamically decided HSI function for the Patalganga image

Result % difference matrix (based on dynamic HSI function) for equivalence class (i) / Partition (z)	1	2	3	4	5	6	7	8	9	10
1	31.4595	32.6077	36.5877	2.6325	3.6958	13.1228	24.6765	17.4335	37.5536	46.3884
2	37.2776	46.7695	14.0495	37.9567	1.4395	46.1176	18.3926	9.5926	10.4002	46.8037
3	42.8281	54.0777	41.7646	1.2983	1.6602	39.7041	48.7101	58.2453	8.1338	18.5350
4	5.7710	49.0691	38.2160	2.9689	22.6297	19.6380	34.2123	50.8345	7.3310	55.3811
5	0.9113	19.5651	35.4748	41.1007	7.0122	45.2776	11.6049	53.9228	4.1491	46.6975
6	25.1551	30.6992	20.6391	33.9123	35.5987	36.8342	56.0569	38.8241	10.7753	46.9550
7	51.4970	33.1910	16.2469	46.1758	32.7724	29.1094	32.1576	0.8653	36.1015	49.0937
8	4.6016	42.5368	39.3962	4.3798	0.3924	32.6203	27.8994	3.9418	16.9248	3.9046
9	17.3630	26.9176	47.0444	18.6882	18.6694	35.8005	4.0049	22.6204	49.2508	30.5126
10	43.8309	2.0041	37.3043	8.5308	45.3732	42.2212	3.0492	25.4183	6.4234	45.0970
11	41.7998	20.9356	17.4706	17.3992	24.0679	32.6477	42.4163	14.5413	6.8747	1.4292
12	22.2286	35.1091	12.8016	9.3076	0.2770	9.6848	40.1519	35.5811	54.4791	37.7329
13	45.3393	18.7474	44.7532	38.8009	20.0577	18.6259	59.1142	47.8397	51.7726	45.5289
14	41.7281	2.5363	13.5216	36.5922	15.6229	33.5431	21.8654	35.8845	29.6682	32.8326
15	42.8897	43.5736	4.1505	45.3139	11.4804	13.5281	48.6938	22.5328	46.0053	46.1073

Tables 12 and 13 present the error matrices for the original BBO classifier with the HSI function ‘Standard deviation’ and the HSI function ‘Entropy’ respectively. Table 14 presents the error matrix for the proposed extended species abundance model of biogeography based classifier. Table 15 represents the producer’s accuracy of classification from which it is reflected that vegetation pixels have been extracted perfectly followed by rocky pixels. Table 16 represents the user’s accuracy of classification. From the error matrix, the KHAT statistics of the proposed land cover feature extractor is calculated as 0.8014.

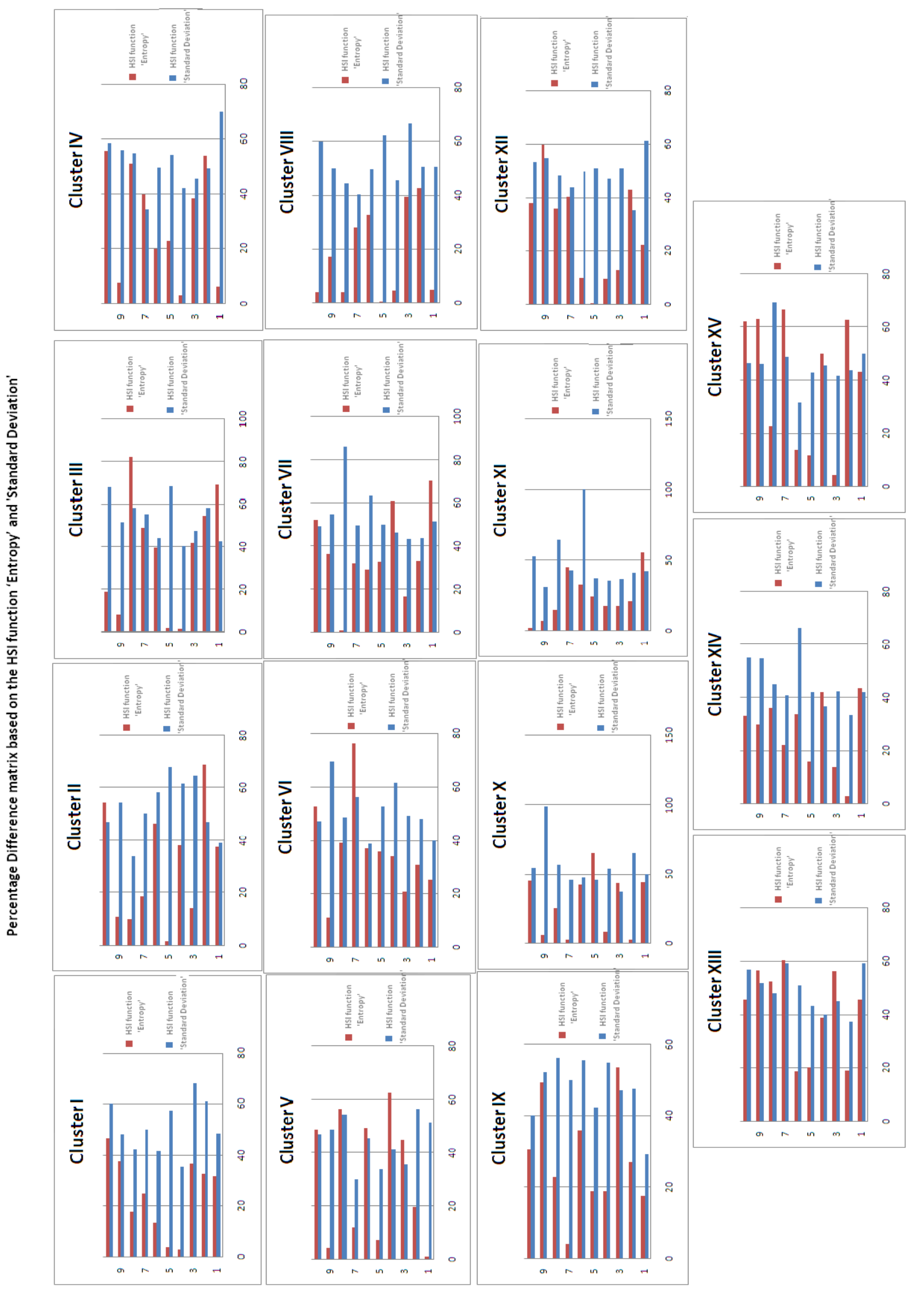


Fig. 11. Bar graph representing the percentage Difference Matrix based on HSI function 'Entropy' and 'Standard Deviation' for the Patalganga Image. Color codes are: Red- Entropy and Blue-Standard Deviation.

Table 10. Classified feature index matrix based on dynamically decided HSI function for the Patalganga image. The feature index codes are 1-Rocky, 2-Snow and 3-Vegetation.

Classified Feature Index based on dynamic HSI function for equivalence class (i) / Partition (z)	1	2	3	4	5	6	7	8	9	10
1	2	2	2	2	2	1	1	1	2	2
2	2	2	2	2	2	2	2	1	1	2
3	2	2	2	2	2	2	2	2	1	2
4	2	2	1	2	2	1	3	2	2	2
5	3	2	3	2	1	2	1	2	2	2
6	2	1	2	2	2	1	2	1	2	2
7	2	2	2	2	2	2	2	3	1	2
8	2	2	2	2	3	2	2	2	2	2
9	1	2	3	2	2	2	2	1	2	1
10	2	2	3	2	2	2	2	2	3	2
11	2	2	2	1	1	1	3	2	1	2
12	1	3	2	2	2	2	2	2	2	2
13	2	2	2	1	2	2	2	2	2	2
14	3	2	2	3	2	2	2	2	2	2
15	2	2	2	2	2	1	2	2	2	2

Table 11. HSI functions chosen dynamically for each equivalence class in the Alwar image. The HSI function codes are 1-Entropy and 2-Standard Deviation.

Dynamic HSI function for equivalence class (i) / Partition (z)	1	2	3	4	5	6	7	8	9	10
1	2	2	2	2	2	2	2	2	2	2
2	2	1	2	2	2	2	2	2	2	1
3	1	2	2	2	2	2	2	1	2	2
4	2	1	2	2	2	2	1	2	2	2
5	2	2	1	1	2	1	2	1	2	1
6	2	2	2	2	2	2	1	2	2	1
7	1	2	2	1	2	2	2	2	2	1
8	2	2	2	2	2	2	2	2	2	2
9	2	2	1	2	2	2	2	2	2	2
10	2	2	1	2	1	2	2	2	2	2
11	1	2	2	2	2	2	1	2	2	2
12	2	1	2	2	2	2	2	2	1	2
13	2	2	1	2	2	2	1	1	1	2
14	1	2	2	1	2	2	2	2	2	2
15	2	1	2	1	2	2	1	2	1	1

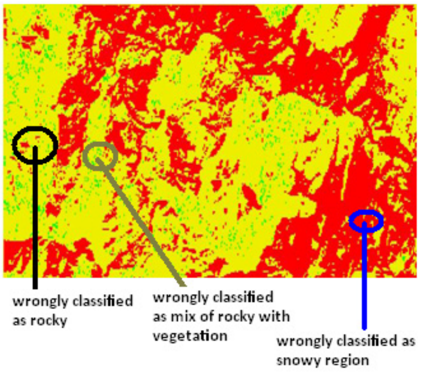

wrongly classified as rocky
wrongly classified as mix of rocky with vegetation
wrongly classified as snowy region

Fig. 12. Final Classified image of Patalganga after applying BBO with the HSI function 'Standard Deviation' for land cover feature extraction. (Kappa Coefficient = 0.7248) [33]

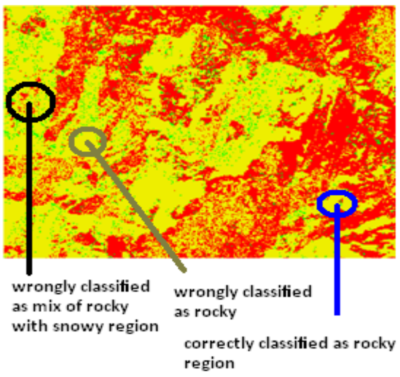

wrongly classified as mix of rocky with snowy region
wrongly classified as rocky
correctly classified as rocky region

Fig. 13. Final Classified image of Patalganga after applying BBO with the HSI function 'Entropy' for land cover feature extraction. (Kappa Coefficient = 0.7870) [13]

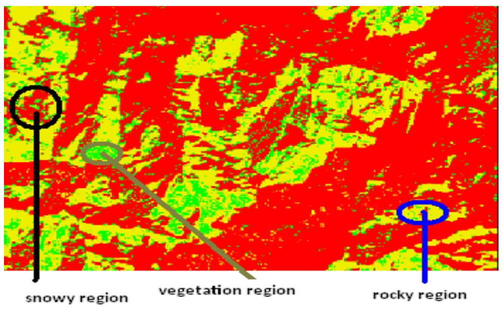


Fig. 14. Final Classified image of Patalganga region after applying BBO with the HSI function 'Entropy' for land cover feature extraction (Kappa Coefficient=0.7870) [13]

Table 12. Error matrix of BBO with the HSI function 'Standard deviation' on Patalganga region. (Kappa Coefficient = 0.7248). [14, 33]

	Vegetation	Snow	Rocky	Total
Vegetation	89	7	7	103
Snow	48	181	4	224
Rocky	24	5	189	227
Total	161	200	200	561

Table 13. Error matrix of BBO with the HSI function 'Entropy' on Patalganga region. (Kappa Coefficient = 0.7870) [12, 13]

	Rocky	Snow	Vegetation	Total
Rocky	197,630	22,572	0	220,202
Snow	18,330	121,008	0	139,338
Vegetation	0	0	16,412	16,412
Total	215,960	143,580	16,412	375,952

Table 14. Error matrix of extended model of biogeography based feature extractor (Kappa Coefficient = 0.8014) on Patalganga region

	Rocky	Snow	Vegetation	Total
Rocky	197,730	22,472	0	220,202
Snow	18,130	121,208	0	139,338
Vegetation	0	0	16,412	16,412
Total	215,860	143,680	16,412	375,952

Table 15. Table 15. Producer's Accuracy

Feature	Accuracy Calculation	Producer's Accuracy
Rocky	197,730/215,860	91.6%
Snow	121,208/143,680	84.3%
Vegetation	16,412/16,412	100%

Table 16. User's Accuracy

Feature	Accuracy Calculation	User's Accuracy
Rocky	197,730/220,202	89.8%
Snow	121,208/139,338	86.9%
Vegetation	16,412/16,412	100%

4.4 Classification Comparison of Hybrid Classifier Based on Extended Species Abundance Model of Biogeography with the other Recent Soft and Hard Computing Classifiers

In this section, we present the results of classification of the extended biogeography based hybrid classifier since the hybrid ACO2/PSO/BBO classifier is the best known classifier developed till date [9] and hence we study the result of substituting the original BBO with our proposed extended BBO in the design of this hybrid bio-inspired intelligent classifier. We compare our results with all the recent soft computing and the traditional hard computing classifiers developed, on the 7-band cartoset satellite image of Alwar region in Rajasthan, India. The 7-band Alwar image is a benchmark image for testing the performance of a bio-inspired classifier on multi-spectral satellite images since this image is a complete image in the sense that it contains all the land cover features that we need to extract and hence land cover feature extraction results are demonstrated and compared using this image as the standard image [25].

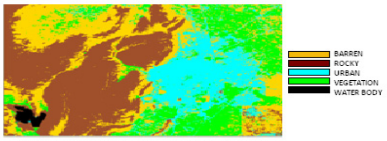
From the discussion above, it is evident that the concept of dynamic HSI function in biogeography based optimization for the development of the proposed hybrid classifier produces better results than the other recent soft computing as well as the hard computing classifiers (traditional probabilistic classifiers such as the Minimum Distance to Mean Classification (MDMC) and the Maximum Likelihood Classifier (MLC) [27].) Also, the proposed classifier is an addition to the taxonomy of soft computing techniques that we proposed in our paper [17]. The satellite image classification results of the other recent soft and hard computing classifiers when applied on the Alwar region are shown below in Fig. 15. Figs. 15 (a) and (b) show the traditional MDMC and MLC classification of Alwar region which achieve a classification accuracy of 0.7364 and 0.7525 only [22, 27]. Figs. 15 (c) and (d) show the result of applying the fuzzy and the integrated rough-fuzzy classification of Alwar region which achieve the Kappa –Coefficients of 0.9134 and 0.9700 respectively [34, 37, 38]. Fig. 15(e) applies the cAntMiner (ACO2) Algorithm on the Alwar Region which has a Kappa Coefficient of 0.964 [8, 9, 36]. Figs. 15(f) and (g) show the result of applying the hybrid ACO-BBO Technique and the hybrid ACO2/PSO technique on the Alwar Image which achieve kappa-coefficients of 0.96699 and 0.975 respectively [2, 10, 32]. Fig. 15 (h) presents the results of the semantic web based classifier on the image with a Kappa Coefficient of 0.9881 [26]. Fig. 15(i) presents the results of the hybrid ACO2/PSO/BBO classifier on the image with a kappa coefficient of 0.9818 [11, 16]. Fig. 15(j) presents the results of the biogeography based feature extractor with non-linear immigration and emigration rates on the image with a kappa coefficient of 0.6912 [21]. Figs. 15 (k) and (l) are the result of applying the membrane computing and the bacterial foraging algorithms (hybrid FPAB/BBO) and achieve accuracy of 0.6818 and 0.6970 respectively [20, 23]. Fig. 15(m) shows the classification results after applying the proposed hybrid ACO2/PSO/(ext-BBO) classifier which achieves the classification efficiency of 0.9916 and hence enhances the accuracy provided by the original hybrid ACO2/PSO/BBO classifier (which had a kappa coefficient of 0.9818). *In fact, the hybrid ACO2/PSO/(ext-BBO) based on the proposed concept of dynamic HSI function in BBO achieves the highest classification efficiency of 99.16% and hence outperforms all the recent soft and the hard computing classifiers developed till date.* Table 21 summarizes and compares the kappa coefficients of the proposed hybrid ACO2/PSO/(ext-BBO) classifier with the other recent classifiers. Table 17 presents the error matrix of the original hybrid ACO2/PSO/BBO classifier and Table 18 presents the error matrix of the proposed ACO2/PSO/ext-BBO classifier. Tables 19 and 20 present the producer's and the user's accuracies of this ACO2/PSO/ext-BBO classifier.

Table 17. Error matrix of Hybrid ACO2/PSO/BBO on Alwar region. (Kappa Coefficient = 0.9818) [16]

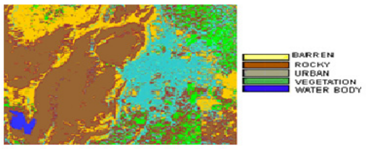
	Vegetation	Urban	Rocky	Water	Barren	Total
Vegetation	142	0	0	0	0	142
Urban	5	190	0	0	0	195
Rocky	0	0	198	0	3	201
Water	0	0	0	70	0	70
Barren	2	0	1	0	163	166
Total	149	190	199	70	166	774

Table 18. Error matrix of Hybrid ACO2/PSO/(ext-BBO) on Alwar region. (Kappa Coefficient = 0.9917)

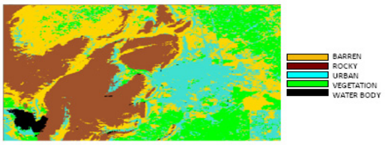
	Vegetation	Urban	Rocky	Water	Barren	Total
Vegetation	142	0	0	0	0	142
Urban	1	190	0	1	0	192
Rocky	0	0	198	0	2	200
Water	0	0	0	69	0	69
Barren	0	0	1	0	163	164
Total	143	190	199	70	165	767



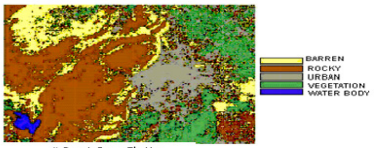
a) Minimum Distance to Mean Classification (MDMC)
(Kappa Coefficient = 0.7364)



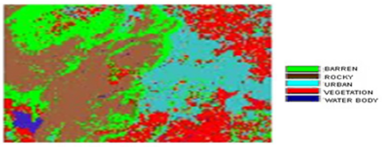
b) Maximum Likelihood Classifier (MLC)
(Kappa Coefficient = 0.7525)



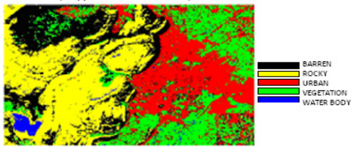
c) Fuzzy Classification of Alwar region
(Kappa Coefficient=0.9134)



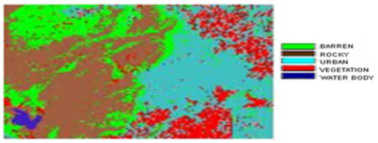
d) Rough-Fuzzy Tie Up
(Kappa Coefficient=0.9700)



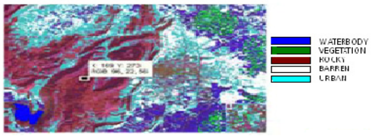
e) cAntMiner Algorithm
(Kappa Coefficient=0.964)



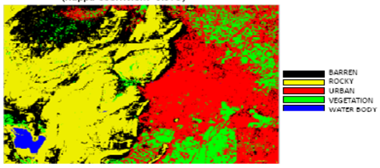
f) Hybrid ACO-BBO Approach
(Kappa Coefficient=0.96699)



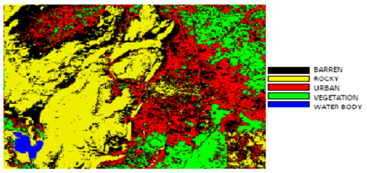
g) Hybrid ACO2/PSO Algorithm
(Kappa Coefficient=0.975)



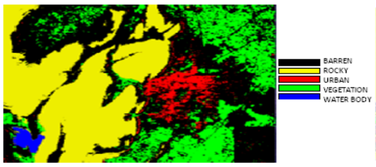
(h) Particle Swarm Optimization (PSO)
(Kappa Coefficient= 0.970



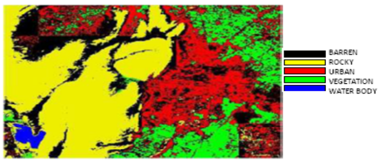
(i) Hybrid ACO2/PSO/BBO Classifier
(Kappa Coefficient = 0.9818)



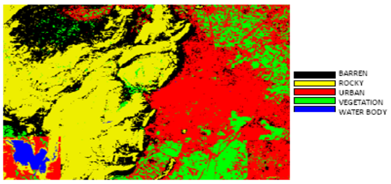
(j) Non-linear biogeography based Feature Extractor
(Kappa Coefficient = 0.6912)



(k) Membrane Computing Classifier
(Kappa Coefficient =0.6818)



(l) Hybrid FPAB Classifier
(Kappa Coefficient=0.6790)



(m) Hybrid ACO2/PSO/ ext-BBO Classifier
(Kappa Coefficient = 0.9917)

Fig. 15. Classified Images of Alwar Region after applying various Soft Computing Techniques

Table 19. User's Accuracy

Feature	Accuracy Calculation	User's Accuracy
Vegetation	142/142	100.0%
Urban	190/192	99.0%
Rocky	198/200	99.0%
Water	69/69	100%
Barren	163/164	99.4%

Table 20. Producer's Accuracy

Feature	Accuracy Calculation	Producer's Accuracy
Vegetation	142/143	99.3%
Urban	190/190	100.0%
Rocky	198/199	99.5%
Water	69/70	98.6%
Barren	163/165	98.8%

Table 21. Kappa coefficients (k) of soft computing classifiers v/s probabilistic classifiers

MDMC	MLC	Fuzzy set	R-F Tie up	ACO	ACO/ PSO	Semantic Web Based	BBO	Hybrid ACO-BBO	Hybrid ACO /PSO/ BBO	MC	Non-linear BBO	FPAB	Hybrid ACO/PSO/ (ext—BBO) based
0.74	0.75	0.91	0.97	0.96	0.975	0.988	0.67	0.967	0.982	0.682	0.69	0.68	0.9917

5 Conclusion and Future Scope

By taking the HSI function to be different and dynamic for each habitat, a lot of flexibility is brought in the choice of best solution. We demonstrate the performance of our extended species abundance model of biogeography based feature extraction technique by running it on two different satellite images. The results indicate that our proposed optimizer is highly effective in extracting land cover features and when compared with the original BBO with the HSI function 'Standard deviation' and with the HSI function 'Entropy' that was applied for land cover feature extraction [12, 33], it proves itself to be a much better classifier as shown in Table 21. We also present a comparative study of the results of substituting our extended-BBO classifier in the hybrid ACO2/PSO/BBO classifier with the other recent soft computing classifiers wherein it is observed that the hybrid ACO2/PSO/(ext-BBO) classifier based on the concept of dynamic HSI function in BBO achieves the highest classification efficiency of 99.16% and hence outperforms the recent soft and the hard computing classifiers developed till date. The future scope of the research includes proposing certain modification to the algorithm so that the Kappa coefficient can be improved further. The current system is implemented using the simple heuristic techniques of standard deviation and entropy; the system performance can be increased by using other heuristic functions for building a dynamic model.

References

[1] Alpaydin, E.: Introduction to Machine Learning. MIT Press, United States of America (2004)

[2] Bansal, S., Gupta, D., Panchal, V.K., Kumar, S.: Swarm Intelligence Inspired Classifiers in Comparison with Fuzzy and Rough Classifiers: A Remote Sensing Approach. In: Ranka, S., et al. (eds.) IC3 2009. CCIS, vol. 40, pp. 284–294. Springer, Heidelberg (2009)

[3] Bhattacharya, A., Chattopadhyay, P.K.: Application of biogeography-based optimization for solving multi-objective economic emission load dispatch problems. Electric Power Components and Systems 38(3), 340–365 (2010)

[4] Blum, C.: Ant colony optimization: Introduction and recent trends. Phys. Life Reviews 2, 353–373 (2005)

[5] Bratton, D., Kennedy, J.: Defining a Standard for Particle Swarm Optimization. In: Proceedings of the 2007 IEEE Swarm Intelligence Symposium, Honolulu, Hawaii, USA (2007)

[6] Clerc, M.: Particle Swarm Optimization. ISTE Publishing, Amsterdam (2006)

[7] Currie, D.J.: Global Ecology and Biogeography. Blackwell Publishing Ltd., U.K (2012)

[8] Dorigo, M., Stuetzle, T.: Ant Colony Optimization. MIT Press (2004)

[9] Otero, F.E.B., Freitas, A.A., Johnson, C.G.: cAnt-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes. Springer, Heidelberg (2008)

[10] Holden, N., Freitas, A.A.: A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In: IEEE Swarm Intelligence Symposium (SIS 2005), pp. 100–107 (2005)

[11] Goel, L., Panchal, V.K., Gupta, D.: Embedding Expert knowledge to Hybrid Bio-Inspired Techniques- An Adaptive Strategy Towards Focused Land Cover Feature Extraction. International Journal of Computer Science & Information Security 8(2), 244–253 (2010) ISSN: 1947-5500

[12] Goel, L., Gupta, D., Panchal, V.K.: Performance Governing Factors of BBO for Land Cover Feature Extraction: An Analytical Study. In: World Congress on Information and Communication Technologies (WICT), pp. 165–170. IEEE Xplore (2011), doi:10.1109/WICT.2011.6141237

[13] Goel, L., Gupta, D., Panchal, V.K.: Biogeography and Plate Tectonics based Optimization for Water body Extraction in Satellite Images. In: Deep, K., Nagar, A., Pant, M., Bansal, J.C. (eds.) Proceedings of the International Conf. on SocProS 2011. AISC, vol. 131, pp. 1–13. Springer, Heidelberg (2012)

[14] Goel, L., Gupta, D., Panchal, V.K.: Information Sharing in Swarm Intelligence Techniques: A Perspective Application for Natural Terrain Feature Elicitation. International Journal of Computer Applications 32(2), 34–40 (2011)

[15] Goel, L., Gupta, D., Panchal, V.K.: Dynamic model of Blended Biogeography based Optimization for Land Cover Feature Extraction. In: Parashar, M., Kaushik, D., Rana, O.F., Samtaney, R., Yang, Y., Zomaya, A. (eds.) IC3 2012. CCIS, vol. 306, pp. 8–19. Springer, Heidelberg (2012)

[16] Goel, L., Gupta, D., Panchal, V.K.: Hybrid bio-inspired techniques for land cover feature extraction: A remote sensing perspective. Applied Soft Computing 12(2), 832–849 (2012)
[17] Goel, L., Gupta, D., Panchal, V.K., Abraham, A.: Taxonomy of Computational Intelligence: A Remote Sensing Perspective. In: World Congress on Nature and Biologically Inspired Computing (NaBIC), November 5-9, pp. 200–206. IEEE Publications, Mexico City (2012), doi:10.1109/NaBIC.2012.6402262
[18] Goel, L., Gupta, D., Panchal, V.K.: Extended Species Abundance Models of Biogeography Based Optimization. In: IEEE Conference on Computational Intelligence, Modelling and Simulation (CIMSim), pp. 7–12. IEEE Xplore and CSDL, Kuantan (2012)
[19] Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
[20] Gupta, D., Das, B., Panchal, V.K.: A Methodical Study for the Extraction of Landscape Traits Using Membrane Computing Technique. In: GEM 2011, WORLDCOMP (2011)
[21] Gupta, S., Arora, A., Panchal, V.K., Goel, S.: Extended Biogeography based Optimization for Natural Terrain Feature Classification from Satellite Remote Sensing Images. In: Aluru, S., Bandyopadhyay, S., Catalyurek, U.V., Dubhashi, D.P., Jones, P.H., Parashar, M., Schmidt, B. (eds.) IC3 2011. CCIS, vol. 168, pp. 262–269. Springer, Heidelberg (2011)
[22] Hand, D.J.: Construction and Assessment of Classification Rules. Wiley (1997)
[23] Johal, N.K., Singh, S., Kundra, H.: A hybrid FPAB/BBO Algorithm for Satellite Image Classification. International Journal of Computer Applications 6(5), 31–36 (2010)
[24] Kang, F., Li, J., Xu, Q.: Damage detection based on improved particle swarm optimization using vibration data. Applied Soft Computing 12(8), 2329–2335 (2012)
[25] Kiefer, R.W., Lillesand, T.M.: Principles of Remote Sensing (2006)
[26] Kumar, S., Gupta, D., Panchal, V.K., Kumar, S.: Enabling Web Services for Classification of Satellite Images. In: International Conference on Semantic Web and Web Services (SWWS 2009), Orlando, FL, USA (2009)
[27] Long III, W., Shobha Srihar, N.: Land cover classification of SSC image: unsupervised and supervised classification using ERDAS Imagine. In: Proceedings of Geoscience and Remote Sensing Symposium, Unsupervised and Supervised Classifications (IGARSS 2004), vol. 4, pp. 20–24 (2004)
[28] Ma, H.: An analysis of the equilibrium of migration models for biogeography-based optimization. Information Sciences 180, 3444–3464 (2010)
[29] Ma, H., Simon, D.: Blended Biogeography based optimization for constrained optimization. Engineering Applications of Artificial Intelligence 24(3), 517–525 (2011)
[30] Ma, H., Ni, S., Sun, M.: Equilibrium Species Counts and Migration Model Tradeoffs for Biogeography based Optimization. In: IEEE Conference on Decision and Control, pp. 3306–3310 (2009)
[31] Ǿhrn, A., Komorowski, J.: A Rough Set tool kit for analysis of data. In: Proc. 3rd International Joint Conference on Information Sciences, Durham, NC, pp. 403–407 (1997)
[32] Omkar, S.N., Manoj, K.M., Mudigere, D., Muley, D.: Urban Satellite Image Classification using Biologically Inspired Techniques. In: Proceedings of IEEE International Symposium on Industrial Electronics, Vigo, Spain, pp. 1767–1772 (2007)
[33] Panchal, V., Singh, P., Kaur, N., Kundra, H.: Biogeography based satellite image classification. International Journal of Computer Science and Information Security 6(2), 269–274 (2009)

[34] Panchal, V.K., Singhal, N., Kumar, S., Bhakna, S.: Rough-fuzzy Sets Tie-up for Geospatial Information. In: ISRO's International Conference on Emerging Scenario in Space Technology & Applications (ESSTA 2008), Chennai, vol. 1 (2008)
[35] Pappula, L.: Application of real coded genetic algorithm for target sensing. In: Sixth International Conference on Sensing Technology (ICST), Kolkata, India, pp. 69–72 (2012)
[36] Parpinelli, S., Lopes, H.S., Freitas, A.A.: Data Mining with an Ant Colony Optimization Algorithm. IEEE Transactions on Evolutionary Computation, Special Issue on Ant Colony Algorithms 6(4), 321–332 (2002)
[37] Pawlak, Z.: Rough Set Theory and its Applications to Data Analysis. Cybernetics and Systems 29(7), 661–688 (1998)
[38] Pawlak, Z.: Rough Sets. International Journal of Computer and Information Science 11, 341–356 (1982)
[39] Simon, D.: Biogeography Based Optimization. IEEE Transactions on Evolutionary Computation 12(6), 702–713 (2008)
[40] Simon, D.: A Dynamic System Model of Biogeography based Optimization. Applied Soft Computing 11(8), 5652–5661 (2011)

Developing Issues for Ant Colony System Based Approach for Scheduling Problems

Ana Madureira[1], Ivo Pereira[1], and Ajith Abraham[2,3]

[1] GECAD - Knowledge Engineering and Decision Support Group, Institute of Engineering – Polytechnic of Porto, Porto, Portugal
{amd,iaspe}@isep.ipp.pt
[2] Machine Intelligence Research Labs (MIR Labs). Scientific Network for Innovation and Re-search Excellence, Auburn, USA
[3] IT4Innovations - Center of Excellence, VSB-Technical University of Ostrava, Czech Republic
ajith.abraham@ieee.org

Abstract. This paper describes some developing issues for ACS based software tools to support decision making process and solve the problem of generating a sequence of jobs that minimizes the total weighted tardiness for a set of jobs to be processed in a single machine. An Ant Colony System (ACS) based algorithm performance is validated with benchmark problems available in the OR library. The obtained results were compared with the optimal (best available results in some cases) and permit to conclude about ACS efficiency and effectiveness. The ACS performance and respective statistical significance was evaluated.

Keywords: Scheduling, Optimization, Weighted Tardiness, Swarm Intelligence, Ant Colony System.

1 Introduction

An important aspect of manufacturing organizations is the improvement of resource utilization. A classical approach of resource utilization optimization is through Scheduling Theory developments. As defined in Baker [1], scheduling is concerned with the problem of allocating scarce resources to activities over time. Scheduling problems are in general nontrivial and exhaustive enumeration of the scheduling solutions set is not usually efficient.

Scheduling problems are generally complex, large scale, constrained, and multi-objective in nature, and classical operational research techniques are often inadequate to effectively solving them [2]. With the advent of computation intelligence, there is a renewed interest in solving scheduling problems through Swarm Intelligence (SI) based techniques.

SI is an innovative computational and behavioral paradigm for solving distributed problems based on self-organization. SI main principles are similar to those underlying the behavior of natural systems consisting of many individuals, such as ant colonies and flocks of birds [3][4]. SI is continuously incorporating new ideas, algorithms, and principles from the engineering and basic science communities [3][4].

SI represents a family of approximate optimization techniques that gained a lot of popularity in the past two decades in Metaheuristics research area which is identified as a field of optimization in Computer Science and Operations Research that are related to algorithms and Theory of Computational Theory. They are among the most promising and successful optimization techniques.

In this paper a set of general guidelines for developing a software tool for solving an optimization problem and support computational study and decision making is described. An Ant Colony System (ACS) based algorithm is proposed to solve the Single Machine Weighted Tardiness Scheduling Problem (WT) and its efficiency and effectiveness is analyzed.

The remaining sections are organized as follows. In Section 2 the Weighted Tardiness problem and some approaches presented in the literature for its resolution are presented. Theoretical foundations, the biological motivation and fundamental aspects of SI paradigm with focalization on the design and implementation of an ACS algorithm and some recent applications of ACS to WT resolution are summarized in section 3. Section 4 a set of general guidelines for developing a software tool for solving an optimization problem is systematized. In Section 5 the ACS proposed approach developing for WT is described. Section 6 presents computational study and discusses obtained results. Finally, the paper presents some conclusions and puts forward some ideas for future work.

2 Problem Definition

One important scheduling problem consists in sequencing a set of jobs for processing on a single processor or machine. The study of Single Machine Scheduling Problem (SMSP) is identified to be very important for several technological and economic reasons, probably the most relevant of which is that good solutions to this problem provide a relevant support to manage and model the behavior of complex systems. In these systems it is important to understand the working of their components, and quite often the SMSP appears as an elementary component in a larger scheduling problem [1][2]. Sometimes the basic SMSP is solved independently, and then results are incorporated into the larger and more complex problem. For example, in a multistage multiple machine problems there are often a critical machine, the bottleneck, whose processing capacity is lower than the necessary. The analysis and treatment of the bottleneck as a SMSP may determine the properties of the entire schedule.

Let us consider the problem of scheduling n jobs for processing without interruption, on a single machine that can handle only one job at a time. For each job j (j=1,..,n), let p_j be its processing time, d_j its due date and w_j the penalty incurred for each unit of time late. The processing of the first job begins at time t =1. The tardiness of a job is given by T_j=Max $\{t_j + p_j - d_j, 0\}$ where t_j is the start time of job j. The objective is to find a sequence that minimizes the sum of Weighted Tardiness (WT) defined on equation 1:

$$Min \sum W_j T_j, \text{ with } T_j = Max\{t_j + p_j - d_j, 0\} \tag{1}$$

The single machine problem, here considered is characterized by the following main conditions:

- a set of n independent jobs (j=1,..,n) is available for processing at time zero and the job descriptors are known in advance;
- a machine is continuously available and is never kept idle while working is waiting;
- the set-up times for the jobs are independent of job sequence and can be included in processing times;
- jobs are processed to completion without preemption.

Under these conditions there is a one-to-one correspondence between a sequence of these n jobs and a permutation of the job indices. In this work we consider a solution such a sequence of jobs where i is the ith job position in the sequence. The total number of different solutions to the SMSP under these conditions is n!.

The SMSP for minimizing total weighted tardiness 1||WT is NP-complete [2][5]. Optimal algorithms for this problem would therefore require a computation time that grows exponentially with the problem size, presenting exponential complexity. Hence, only small sized instances of this problem can be solved in an efficient way. Several branch-and-bound procedures and dynamic programming techniques have been proposed in literature [5][6]. As indicated in [6] the simple dispatching heuristics (EDD, SWPT, COVERT or AU) do not consistently produce good quality solutions. In recent years, much attention has been dedicated to Metaheuristic that are considered to be efficient tools for solving hard combinatorial optimization problems.

Ant Colony Optimization (ACO) is probably the most successful example of artificial/engineering Metaheuristic based optimization techniques with numerous applications to real-world problems[4], and for minimization of total weighted tardiness. Merkle and Middendorf [7] describe a contribution for solving permutation problems to SMSP for Total Weighted minimization. Liao and Juan [8] propose an ACO to minimize the tardiness in a SMSP with utilization of setup times. In Yagmahan and Yenisey [9] a multi-objective scheduling problem approach based on ACO for scheduling to reduce the total scheduling cost is proposed. Anghinolfi and Paolucci [10] describe a new ACO approach to face the single machine total weighted tardiness scheduling with sequence dependent setup times problem. In Srinivasa Raghavan and Venkataramana [11] a parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization is proposed. Additionally, some relevant works could be identified, see for example [1][12][13].

3 Ant Colony Optimization

Ant Colony Optimization (ACO) has been formalized as a Metaheuristic by Dorigo and collaborators [3][14] that are inspired and mimic natural metaphors to solve complex optimization problems . A Metaheuristic can be defined as a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide range of different optimization problems [4].

A subset family of Metaheuristics, SI is considered an innovative and creative approach to problem solving that takes inspiration from the collective intelligence of swarm of biological populations, and was discovered through simplified social behaviors model simulation of insects and other animals [4][15]. ACO algorithm is among the most promising SI inspired optimization class of optimization techniques.

Table 1. Analogy between Natural and Artificial Ants

Natural Ant Colony	*Artificial Ant Colony*
Ant	Agent
Ant Colony	Set of Ants/Iterations
Pheromone	Diversity Mechanism
Path	Solution
Evaporation	Pheromone update

The ACO algorithm takes inspiration from the foraging behavior of some ant species (Table 1). These ants deposit pheromone on the ground in order to mark some favorable path that should be followed by other members of the colony. ACO exploits a similar mechanism for solving optimization problems. The ACO algorithm is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. For this reason description of ACO algorithms are normally accompanied through Traveling Salesman Problem (TSP) notation and illustrative examples [16-18]. This algorithm, initially proposed by Marco Dorigo in his PhD thesis [14], is a member of ACO family and it constitutes some Meta-Heuristic optimizations.

The first proposed ACO algorithm is known as Ant System [16] that was aiming to search for an optimal path in a graph. It was based on the foraging behavior of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several algorithms have emerged, drawing on several aspects of the behavior of ants [4].

Table 2. Non-exhaustive ACO algorithms list [4]

Algorithm	*Authors*	*Year*
Ant System(AS)	Dorigo et al.	1991
Elitist AS	Dorigo et al.	1992
Ant-Q	Gambardella & Dorigo	1995
Ant Colony System	Dorigo & Gambardella	1996
MAX-MIN AS	Stutzle&Hoos	1996
Hyper-Cube AS	Blum et al.	2001

The designation ACO is a generic term that includes algorithms based on the behavior of ants. Since the early 90s, when the first ACO algorithm - Ant System - was proposed in [16], different algorithms (Table 2) and successful applications have been described and a substantial theoretical results have becoming available that provides useful guidelines to researchers and practitioners in further applications of ACO based algorithms [17-19].

The general ACO algorithm is described in Table 3. After initialization, the ACO iterates over three main steps: at each iteration, a number of solutions are constructed by the ants; these solutions could be then improved, optionally, through a local search, and finally the pheromone is updated through two possible events: evaporation and by increasing the pheromone levels associated with a chosen set of good solutions.

Table 3. Ant Colony Optimization Algorithm

```
Set ACO parameters.
Initialize pheromone trails
While termination criteria not met do
  Construct AntSolutions
  Apply Localsearch (optional)
  Update Pheromones
EndWhile
```

A more detailed description of the three phases can be stated as follows[4]:

- ConstructAntSolutions: A set of m artificial ants constructs solutions from elements of a finite set of available solution components C = {cij }, i = 1, . . . , n, j = 1, . . . , |Di|. A solution construction starts from an empty partial solution sp = ∅. At each construction step, the partial solution sp is extended by adding a feasible solution component from the set N(sp) ⊆ C, which is defined as the set of components that can be added to the current partial solution sp without violating any of the constraints in Ω. The process of constructing solutions can be regarded as a walk on the construction graph GC = (V, E) as stated in [4]. The selection of a solution component from N(sp) is guided by a stochastic mechanism, which is biased by the pheromone associated with each of the elements of N(sp). The rule for the stochastic choice of solution components vary across the different proposed ACO algorithms but, in all of them, it is inspired by the Goss model (experimental setup for the double bridge experiment) of the behavior of real ants assuming that at a given moment in time m1 ants have used the first bridge and m2 the second one, the probability p1 for an ant to choose the first bridge is given by [4] :

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h} \quad (2)$$

where parameters k and h are to be fitted to the experimental data. Monte Carlo simulations showed a very good fit for $k \approx 20$ and $h \approx 2$.

- ApplyLocalSearch: Once solutions have been constructed, and before updating the pheromone, it is common to improve the solutions obtained by the ants through a local search. This phase, which is highly problem-specific, is optional although it is usually included in state-of-the-art ACO algorithms.
- UpdatePheromones: The aim of the global pheromone update is to increase the pheromone values associated with good or promising solutions, and to decrease those that are associated with bad ones. Usually, this is achieved by decreasing all the pheromone values through pheromone evaporation, and by increasing the pheromone levels associated with a chosen set of good solutions.

Several ACO algorithms have been proposed in the literature, which differ in some decisions characterizing the construction of solutions and update pheromone procedures [4]. Among the most successful variants we have chosen the ACS algorithm to apply to the SMSP to WT resolution.

The most interesting contribution of ACS [4][13] is the introduction of a local pheromone update and the pheromone update performed at the end of the construction process (named offline pheromone update).

ACS algorithm can be stated as follows [13]: m ants are initially positioned on n cities chosen according to some initialization rule (randomly, for example). Each ant builds a tour (feasible solution) by repeatedly applying a stochastic greedy rule (the state transition rule). While constructing its tour/path, an ant also modifies the amount of pheromone on the visited edges (cities) by applying the local updating rule. Once all ants have terminated their tour/path, the amount of pheromone on edges/cities is modified again, by global updating rule applying. Ants are guided, in building their solutions, by both heuristic information (they prefer to choose short edges), and by pheromone information (an edge with a high amount of pheromone is a very desirable choice). The pheromone updating rules are designed to give more pheromone to edges/cities which should be visited by ants.

The local pheromone update is performed by all ants after each construction step. Each ant applies it only to the last edge traversed:

$$\tau_{ij} = (1 - \varphi) \,.\, \tau_{ij} + \varphi \,.\, \tau_0 \quad (3)$$

where $\varphi \in [0,1]$ is the pheromone decay coefficient, and τ_0 is the initial value of the pheromone.

The main goal of the local pheromone update is to introduce diversity in the search process performed by subsequent ants during an iteration by decreasing the pheromone concentration on the traversed edges, ants encourage subsequent ants to choose other edges and, hence, probably to produce different solutions. This mechanism makes it less likely that several ants produce identical solutions during one iteration.

The offline pheromone update, is applied at the end of each iteration by only one ant, which can be either the iteration-best(Lib) or the best-so-far(Lbs). However, the update formula is slightly different:

$$\tau_{ij} = \begin{cases} (1-\rho) * \tau_{ij} + \rho * \Delta\tau_{ij} & \textit{if (i, j)belongs to the best tour,} \\ \tau_{ij} & \textit{otherwise} \end{cases} \tag{4}$$

where τij = 1/Lbest , where Lbest can be either Lib or Lbs.

Another important difference between ACS and AS is in the decision rule used by the ants during the construction process.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum\limits_{C_{ij} \in N(S^{p})} \tau_{ij}^{\alpha} * \eta_{ij}^{\beta}} & \textit{if } c_{ij} \in N(S^{p}) \\ 0 & \textit{otherwise} \end{cases} \tag{5}$$

In ACS, the so-called pseudorandom proportional rule is used by the ants during the construction process: the probability for an ant to move from city *i* to city *j* depends on a random variable *q* uniformly distributed over [0,1], and a parameter q_0; if $q \leq q_0$, $j = \text{argmax}_{c_{ij} \in \text{N(Sp)}} \{\tau_{il}\eta_{il}^{\beta}\}$ otherwise Equation 5 is used.

Additional information about ACO based algorithms details of implementation could be found in [4][13].

4 Developing Issues for Optimization Approaches

The first aspect to be considered when identifying a decision making problem - optimization problem - and the need for specifying a tool for its resolution refers to problem modeling.

The mathematical model is built from the formulation of the problem and can be inspired by theoretical models related in the literature. This will reduce the problem to well-studied optimization models that are in general simplifications of real world problems.

Once the problem is modeled, the following stages are considered relevant for optimization approaches development (Figure 1), following a set of general guidelines systematized by Talbi[20], for solving a given optimization problem.

Initially should be addressed, based on the state of the art of optimization methods (exact or approximation), the question of which optimization technique is best suited to solve the problem considering the complexity and difficulty of the optimization problem (NP class, size and structure of the instances) and the requirements of the optimization problem (search time, quality of solutions and robustness).

The use of exact methods is suitable when the identified instances of the problem are solved in the time required. Meta-heuristics are a feasible alternative to obtain satisfactory solutions in circumstances where the complexity of the problem or available search time did not allow the use of exact methods.

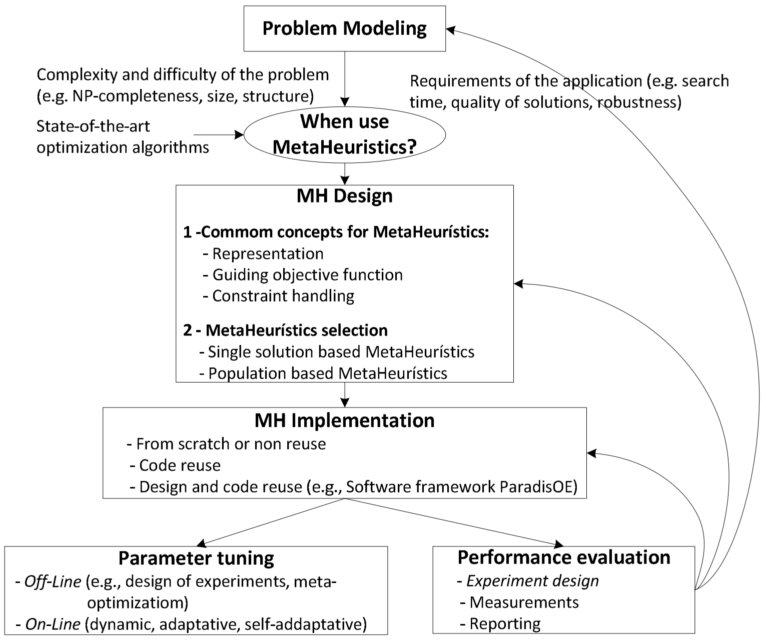

Problem Modeling
Complexity and difficulty of the problem (e.g. NP-completeness, size, structure)
Requirements of the application (e.g. search time, quality of solutions, robustness)
State-of-the-art optimization algorithms
When use MetaHeuristics?
MH Design
1 -Commom concepts for MetaHeurístics:
- Representation
- Guiding objective function
- Constraint handling
2 - MetaHeurístics selection
- Single solution based MetaHeurístics
- Population based MetaHeurístics
MH Implementation
- From scratch or non reuse
- Code reuse
- Design and code reuse (e.g., Software framework ParadisOE)
Parameter tuning
- Off-Line (e.g., design of experiments, meta-optimizatiom)
- On-Line (dynamic, adaptative, self-addaptative)
Performance evaluation
- Experiment design
- Measurements
- Reporting

Fig. 1. Developing issues for optimization problem solving [20]

From the moment the necessity of specifying a meta-heuristic is identified, some questions, common to all Meta-Heuristics, related to the encoding/representation of the solutions, the definition of objective function and constraints handling must be stated.

The development of software tools for the MH is a relevant task considering the variety of optimization problems identified and continuous evolution of models associated with optimization problems. The problem may be modified or require further refinements: some objectives and constraints can be inserted, deleted or changed.

It is clear the interest and the need in developing systems or automatic tools for decision support based MH. For managers it is important to select, implement and apply optimization algorithms without requiring deep knowledge on programming and optimization. For experts in optimization and software development is useful to evaluate and compare different algorithms, transform/adapt algorithms, develop new algorithms, combine and parallelize algorithms. Generally, the literature identifies three main approaches used for the development and implementation of Metaheuristics [20]: From scratch or no reuse (considering the simplicity of MH implementation, but requires time and effort and it is error prone); Code reuse (consists on reusing free programs codes and libraries Open Source, adapting to the treated problem is often time consuming, error prone, and the coding effort using libraries remains important); and Design and code reuse(Software frameworks, its main objective is to overcome above related problems).

In general, the effective resolution of a problem requires the application of different methods of tuning parameters among others. The tuning of parameters can allow greater flexibility and robustness but requires a careful initialization. The parameters can have a major influence on the efficiency and effectiveness of the search. Becomes not obvious, a priori, the setting of parameters to use. The values for the parameters depend on the problem, instances structure and the time available to solve the problem. There are no universal values for the parameters considered for Metaheuristic based algorithms. Being widespread view that its definition must result from a careful experimental effort, towards their tuning.

Performance analysis corresponds to the last stage of development of MH. The theoretical analysis based on the worst case and average provides some insights in solving some optimization models. However, it is considered that the performance evaluation of the MH must be supported by a comprehensive set of computational tests, following these aspects/phases: definition of the test plan (test objectives, selection of input variables and instances); definition of measurement criteria (quality of solutions, robustness and computational effort) and the reporting and analysis of results (graphic display of results, interpretation of results, statistical analysis).

5 ACS Proposed Approach Developing for Weighted Tardiness

The scheduling problem to deal with is included into the class of combinatorial optimization problems common in industrial practice. Due to its complex nature and the resolution of such problems to optimality, in an acceptable time for the process of decision making, is virtually impossible. Thus, there is an important issue that refers to the resolution of this class of problems obtaining satisfactory quality solutions on reasonable computing time, for which there is no knowledge of the existence of efficient methods.

A software tool was developed and implemented in Java, to perform the computational study aiming to analyse and evaluate the performance of ACS, on resolution of SMSP for minimization of total weighted tardiness.

Table 4. Ant Colony System for WT

```
Begin
  Set ACS parameters.
  Initialize pheromone trails
  While termination criteria not met do
    Construct Ant Solutions
      Each ant build a solution
      Apply LocalSearch
      Return Constructed Solution
    Apply Localsearch
    Global Update Pheromone
  EndWhile
  Return Best solution
End
```

The solutions are encoded by the natural representation (string), each position corresponds to a job index and the position of the job index is the correspondent processing order. The number of positions on the string corresponds to the number of jobs (problem size).

The initial colony generation process consists in applying some mechanism generator to a starting ant solution. The initial solution is defined by the priority rule EDD rule, in which an initial solution (ant) is defined by the due dates increasing ordering, thus giving priority to tasks with small due dates.

As mentioned above, we implemented an ACS algorithm (Table 4) for WT resolution [21]. The ACS differs from the previous proposed Ant System due to three main aspects [11]: the state transition rule, the global updating rule, and the local updating rule. When applied to the SMSP for WT minimization, each ant constructs a feasible sequence by selecting an unscheduled job j to be on the ith position of the partial sequence constructed so far. This process is influenced by specific heuristic information ηij, as well as the pheromone trails τij.

5.1 Solution Construction

During the construction process the decision of adding job j to the partial sequence is made through ACS state transition rule, which is given by equation 6:

$$j = \begin{cases} \arg max_{j\in\omega}\left\{\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}\right\} & \text{if } q \leq q_0 \\ S & \text{otherwise} \end{cases} \tag{6}$$

where ω is the set of unscheduled jobs, α controls the relative importance of the pheromone trails, β determines the influence of the heuristic information, q is a random number uniformly distributed over [0,1], q_0 is a parameter ($q_0 \in [0,1]$) and S is a job selected according to the probability defined by equation 7:

$$P_{ij} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}}{\sum_{j\epsilon\omega}\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}} & if\ j \in \omega \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The state transition rule favors job selection in terms of pheromone amount versus heuristics information. The parameter q_0 provides a way to balance between exploration of new jobs and exploitation of accumulated knowledge. In other words, when ant m has to choose a job to append to the partial sequence, a random number q is generated, with $q \in [0,1]$, and if $q \leq q_0$ the best job is chosen (exploitation), otherwise a job is chosen (exploration).

The heuristic used by ants to compute the heuristic information is given by equation 8:

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{8}$$

where d_{ij} is the total weighted tardiness for the partial sequence of jobs generated so far. The proposed solution construction procedure could be summarized on Table 5.

Table 5. Solution Construction Algorithm

```
Begin
  While number of jobs not met do
    Select a new job
    Add new job to the sequence
    Perform Local Pheromone Update
  EndWhile
  Apply LocalSearch
  Return Constructed Solution
End
```

The ants construct the solution as follows: each ant starts from a randomly selected job. Then, at each construction step the ant selects a new job through a transition rule (equation 6). Each ant keeps a "memory" of its path, and the subsequent job is chosen among the unscheduled jobs. At each construction step, an ant probabilistically chooses the next job to add to the sequence. The probabilistic rule is biased by pheromone values and heuristic information. Once a job is added to the sequence, the local pheromone is updated. This process is repeatedly applied until all jobs are scheduled. Before returning the generated solution, it still undergoes a local search algorithm. The goal is to improve the current solution by iteratively moving to a neighbor solution selected from a neighborhood of solutions according to a defined rule, which in this case, is the minimization of the total weighted tardiness. This process is quite costly in terms of computational time, but improves significantly the generated solution. However, we will address this process in more detail in the subsequent sections.

5.2 Pheromone Update Rule

The ACS pheromone update rule consists of both local and global update rule. The local pheromone update is performed by each ant after adding a new job to the partial sequence, and is given by equation 9:

$$\tau_{ij} = (1-\rho) \, . \, \tau_{ij} + \rho \, . \, \tau_0 \qquad (9)$$

where $\rho \in [0,1]$ is the pheromone decay coefficient, and τ_0 is the initial trail intensity (equation 10):

$$\tau_0 = \frac{1}{n \, .WT_{EDD}} \qquad (10)$$

where WT is the total weighted tardiness for a sequence generated by the EDD rule and n is the number of jobs. The main goal of the local update rule is to make the decision of appending job j on position i less desirable for the others ants so that the exploration of different sequences is favored [12].

The global pheromone update is applied at the end of each iteration by only one ant, which can be either the *iteration-best* or *best-so-far* [4]. We followed the *best-so-far* strategy, where the ant with the best solution so far contributes to the pheromone trail update, according to the equation 11:

$$\tau_{ij} = (1-\rho)\,.\,\tau_{ij} + \rho\,.\,\Delta\tau_{ij} \tag{11}$$

where $\Delta\tau_{ij}=1/WT^*$ for all edges (i, j) belonging to the best solution found so far (WT^* is the total weighted tardiness of the best solution).

5.3 Local Search

Local Search performs a blind search, since they only accept sequential solutions which improve the value of the objective function. Essentially, consists of moving from one solution to another, in the neighborhood, according to some defined rules or local changes. The sequence of solutions trajectory depends heavily on the initial solutions and on the neighborhood generation mechanisms adopted which defines the neighboring structure [22]. In this work, we considered adjacent pairwise interchange as a neighborhood structure to solve the problem at hand [21]. The main weakness of basic Local Search algorithms is their inability to escape from local optima.

The local search strategy described above is applied to all sequences constructed by the artificial ants. This strategy produces better results but is more costly in terms of computational time.

5.4 Illustrative Example

A chemical industry produces different types of products, but can only make one at a time. The production manager has to decide on the issue of sequencing of 5 tasks on a single machine (Table 6). For each task j (j = 1, ..., 5), p_j is the processing time, d_j the delivery date and w_j the penalty associated with task j per unit of time delay. The number of admissible solutions for this instance of problem is 5! = 120. The size of the problem is reduced to facilitate their understanding. It becomes clear that in this situation would be feasible the enumeration of all admissible solutions and select the best. However, this is no longer feasible for instances with larger dimension.

Consider the ACS with 25 ants, a pheromone evaporation rate of 80%, α values (scale the heuristic value) and β (importance of pheromone) equal to 1. Initially the pheromone is zero.

From proposed ACS execution, we obtained the solutions presented in Table 7, where it is possible to verify the quality evolution of the paths chosen by the ants. The paths are constructed incrementally and randomly according to the values of pheromone and heuristic information, with an increased probability of choosing a solution with better heuristic value and greater pheromone. It is possible to confirm that, in the end of the process, ants choose better paths, with a propensity to choose the best so far.

Table 6. Illustrative example

Job j	p_j	d_j	w_j
1	2	5	1
2	4	7	6
3	1	11	2
4	3	9	3
5	3	8	2

Table 7. ACS Step by step

Ant i	$Solution_i$	f
1	[2 4 5 3 1]	12.0
2	[1 3 2 4 5]	13.0
3	[4 2 1 3 5]	14.0
4	[1 2 4 3 5]	10.0
5	[1 2 3 4 5]	13.0
6	[4 3 2 5 1]	20.0
7	[1 2 4 3 5]	10.0
8	[3 1 2 4 5]	13.0
9	[2 5 4 3 1]	11.0
10	[1 3 2 4 5]	13.0
11	[3 1 4 2 5]	28.0
12	[3 1 2 4 5]	13.0
13	[2 5 4 3 1]	11.0
14	[3 4 2 5 1]	20.0
15	[2 3 4 5 1]	14.0
16	[4 5 2 3 1]	26.0
17	[2 5 4 3 1]	11.0
18	[2 5 3 4 1]	14.0
19	[1 2 4 3 5]	10.0
20	[3 1 2 4 5]	13.0
21	[3 2 5 4 1]	14.0
22	[1 2 4 3 5]	10.0
23	[1 2 4 3 5]	10.0
24	[3 1 2 4 5]	13.0
25	[1 2 4 3 5]	10.0

Table 8. Final Pheromone Matix

Job	1	2	3	4	5
1	0,089	0,083	0,016	0,013	0,013
2	0,011	0,011	0,011	0,1	0,013
3	0,013	0,013	0,011	0,013	0,073
4	0,013	0,013	0,083	0,011	0,013
5	0,013	0,011	0,013	0,013	0,011

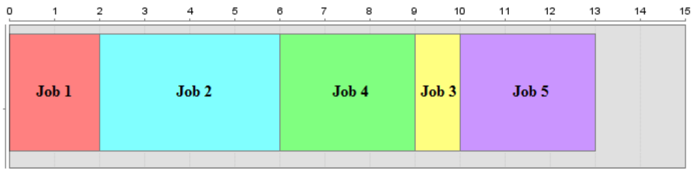


Fig. 2. ACS scheduling plan

At the end of the twenty five iterations, the path with more pheromone is [1 2 4 3 5], with a heuristic $f = 10$, which represents the best solution found by ACS (Table 8). This path was chosen by 6 ants. The scheduling plan is illustrated in Figure 2.

6 Computational Study

A software tool was developed to perform the computational study aiming to analyse and evaluate the performance of ACS, on resolution of SMSP for minimization of total weighted tardiness. The computational tests were carried out on a PC with Intel Xeon W3565 at 3.20 GHz, with the ACS coded in Java. The ACS performance was tested on 75 benchmark instances of WT problem for different sizes n=40, n=50, n=100, available at OR-Library [23]. We select for testing the first 25 instances for each size and not their instances where better results were obtained.

In this section a computational study is carried out in order to analyse SI based algorithms – ACS - on the resolution of benchmark problems considering quality of solutions and computational times.

Performance analysis of EC is a necessary task to perform and must be done on a fair basis. A theoretical approach is generally not sufficient to evaluate an MH based algorithm. To evaluate the performance experimentally and/or comparing in a systematized way, the following three steps must, generally, be considered [20]:

- **Experimental Design**: the goals of the experiments, the selected instances, and optimization criteria have to be defined.
- **Measurement**: the measures to compute are selected. After executing the different experiments, statistical analysis is applied to the obtained results. The analysis of performance must be done based on state-of-the-art optimization algorithms dedicated to the problem.

- **Reporting**: the results must be systematized in a comprehensive way, and an analysis is carried out following the defined goals. Another important issue is related with insurance of the reproducibility of the computational experiments.

ACS based algorithm is evaluated on the resolution of WT instances and its efficiency and effectiveness will be analysed. Statistical analysis is performed in order to estimate the significance and confidence of the obtained results.

We pretend to evaluate the adequacy of Swarm Intelligence based algorithms to the WT resolution. ACS effectiveness and efficiency is evaluated on 75 benchmark instances of WT problem for different sizes (25 instances with 40, 50 and 100 jobs, respectively).

We consider that academic benchmark problems are an effective evaluation framework since they have been used by multiple authors and diverse application areas over the years, allowing an efficient comparing framework with previous work related on literature. Additionally, they permit an insight of global behavior and performance on a class of scheduling problems which are our main objective.

6.1 Parameter Tuning

The ACS algorithm has a certain number of parameters that need to be set appropriately [18]. As such, we performed a preliminary study to identify which set of values would yield better results for minimizing total weighted tardiness, for each size in consideration. The study focused on testing different values for α and *β*, which are used to regulate the relative influence of the pheromone and heuristic information; *m*, the number of ants; *ρ*, the pheromone evaporation rate; and q_0, the probability of choosing the next job as defined on equation 6. In Table 9 we present the different tested values, as well as the standard deviation *σ* for obtained results.

The tests were performed by using four different sets of values, following some conclusions referred in [10]. For each set we computed n=1 simulations for each instance under analysis, as shown in Table 9. The conclusions from the obtained results could be summarized:

- *α*: This parameter is usually set to 1, and most of the times is not even considered;
- *β*: The value of *β* equal to 5 produced better results during most of the runtimes;
- *m*: The value of ants defined for each instance resulted in a good anytime performance;
- *ρ*: Higher values of *ρ* produced better results;
- q_0: Good values of q_0 tend to be close to 1. As such, we used 0.98, which proved to be a good choice.

The standard deviation presented in Table 9 shows the variation in the results relatively to the average deviation from the optimum value. In fact, a low deviation indicates that the results tend to be very close to the average value.

After analyzing the results we concluded that the last set of values (highlighted in bold), could yield better results in the computational tests, as such the implemented ACS algorithm was parameterized with this values. The obtained results are presented and discussed in the next section.

Table 9. Parameter tuning

No. jobs	*α*	*β*	*m*	*ρ*	*q_0*	*Σ*
40	1	1	30	0.80	0.90	0.1546 ± 0.1616
	1	2	30	0.10	0.98	0.1746 ± 0.1588
	1	5	30	0.30	0.98	0.1273 ± 0.0932
	1	**5**	**30**	**0.60**	**0.98**	**0.0909 ± 0.0839**
50	1	1	50	0.80	0.90	0.1939 ± 0.2582
	1	2	50	0.10	0.98	0.1942 ± 0.1948
	1	5	50	0.30	0.98	0.1688 ± 0.1523
	1	**5**	**50**	**0.60**	**0.98**	**0.1226 ± 0.1073**
100	1	1	80	0.80	0.90	0.5253 ± 0.5659
	1	2	80	0.10	0.98	0.4176 ± 0.3776
	1	5	80	0.30	0.98	0.4285 ± 0.3868
	1	**5**	**80**	**0.60**	**0.98**	**0.3969 ± 0.3753**

6.2 Discussion of Results

Initially, we developed n=5 simulations for each instance under analysis. In order to analyze the obtained results, performance measures were computed: the best, the average, and worst value and the deviation error from the best obtained value to the optimal (best known available in OR-Library [23]). The relative percentage of deviation error is determined by *%error= (Best-Optimal)/Best* formula.

Table 10. Results for 5 runs

No. jobs	*Average time (s)*	*σ*	*No. of optimal*
40	15	0.0366 ± 0.0391	8
50	50	0.0552 ± 0.0489	4
100	1086	0.1872 ± 0.1714	0

The obtained solutions values by the proposed ACS algorithm are presented in Figure 3 for n=40, Figure 4 for n=50 and Figure 5 for n=100, also in Table 10 is presented the average computational time, the standard deviation, and the number of optimum values.

The following figures presents the results obtained from the performed tests. Each figure shows the average deviation, as well as the deviation of each intances from the optimum value. Through the figures is also possible to identify the instances were the optimum value was reached, i.e., the ones were the deviation value is zero. Some of this instances appear to have reached the optimum value, when in fact there is a slight deviation that is not noticeable in the figures. For that reason we present in Table 10 the total number of optima for n=40, n=50, and n=100.

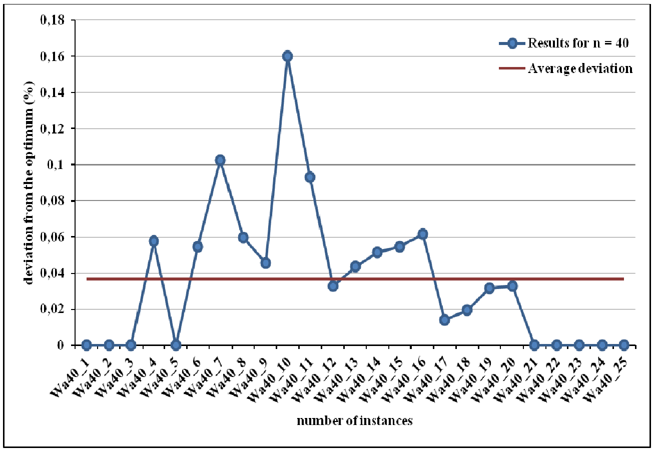


Fig. 3. Results obtained with 5 runs for n=40

In general most of the instances were solved in relatively short computational time. For n=40 the average time was 15 seconds, as for n=50 the average time was 50 seconds. Only the instances of n=100 took more time to be solved (~18 minutes).

As an extension to our study, we tested ACS again, and see if the solution already obtain could improve. Therefore, the number of simulations was increased to n=20 and the exactly same set of parameters were used. The results obtained are presented in Table 11 supported by Figure 6 for n=40, Figure 7 for n=50 and Figure 8 for n=100.

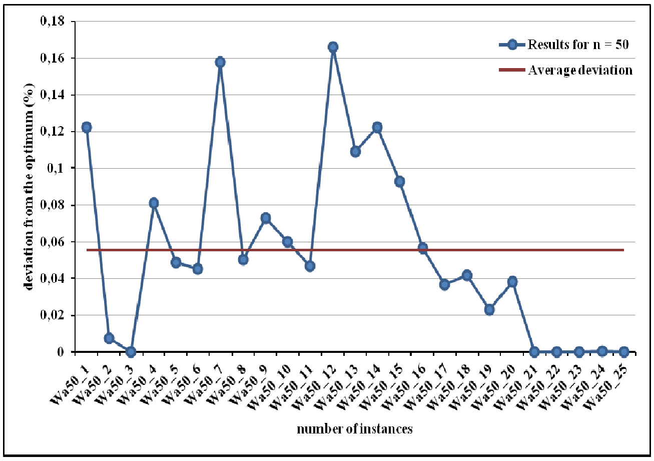


Fig. 4. Results obtained with 5 runs for n=50

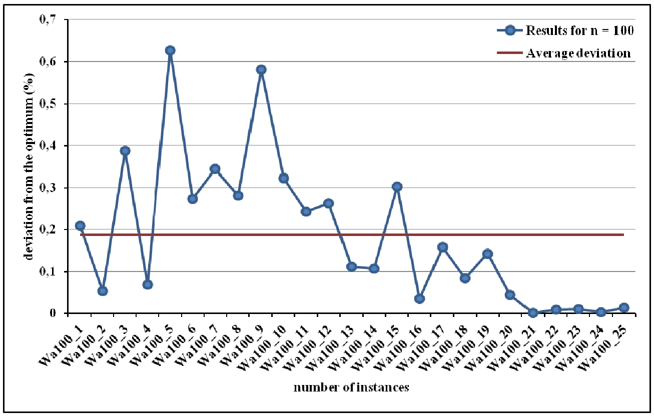


Fig. 5. Results obtained with 5 runs for n=100

Table 11. Results for 20 runs

No. jobs	*Average time (s)*	*σ*	*No. of optimal*
40	14	0.0172±0.0209	11
50	49	0.0374±0.0328	5
100	1072	0.1524±0.1523	0

The average computational time remained the same, and the results improved significantly. As shown in Table 11, and presented in Figure 6, Figure 7, and Figure 8 the number of optima increased. Moreover, in most of the instances the deviation from the optimum value decreased. These results support the assumption that the outcome of running the ACS more times would be to find the best known solutions on all instances.

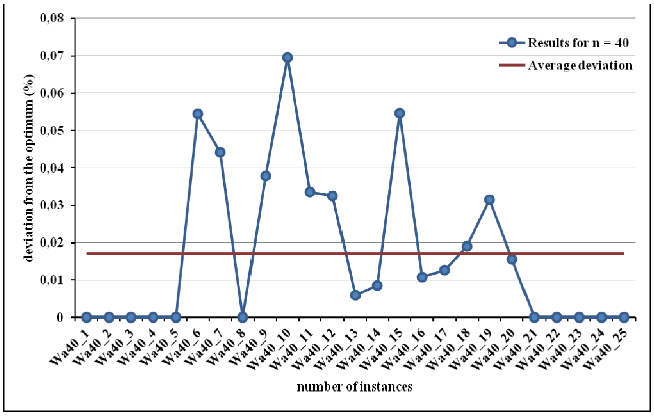


Fig. 6. Results obtained with 20 runs for n=40

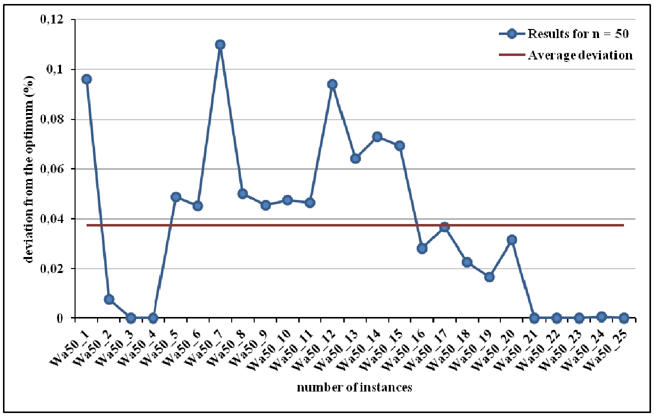


Fig. 7. Results obtained with 20 runs for n=50

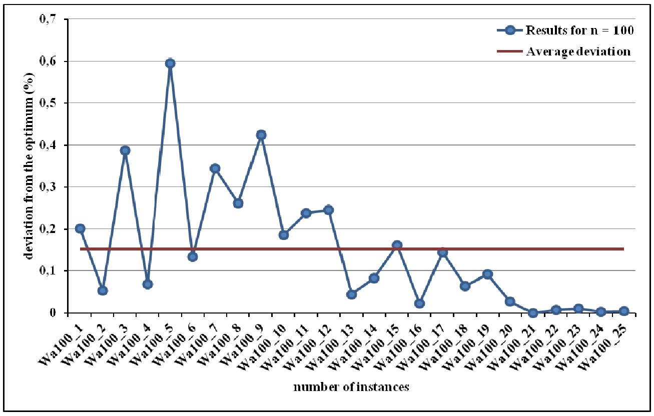


Fig. 8. Results obtained with 20 runs for n=100

6.3 Significance Analysis

The boxplot from Figure 9 allows the analysis of confidence interval, making its synthesis and compared mean values by ACS and optimal solutions in terms of minimization for *Weighted Tardiness* (WT) with 40 jobs. From the boxplot analysis we can conclude that ACS has been effective on the resolution of WT considering that its obtained mean values were similar to the optimal solutions (in some instances the best known). It is not clear, from the graph analysis, if exist significant difference of the performance of ACS related with optimal solutions.

Additionally, some statistical sampling were retrieved to summarize important features from the observed instances From the analysis of statistical sampling summary based on WT minimization (Table 12), it is possible to conclude that does not exist statistic evidence of the difference significance between optimal solutions and ACS performance on WT resolution for 40 jobs. This conclusion can be supported either by central tendencies and dispersion measures. This evidence can be observed even on median and dispersion indicators. Regarding variability, through standard deviation and interquartile range analysis it possible to conclude that ACS presents similar variability.

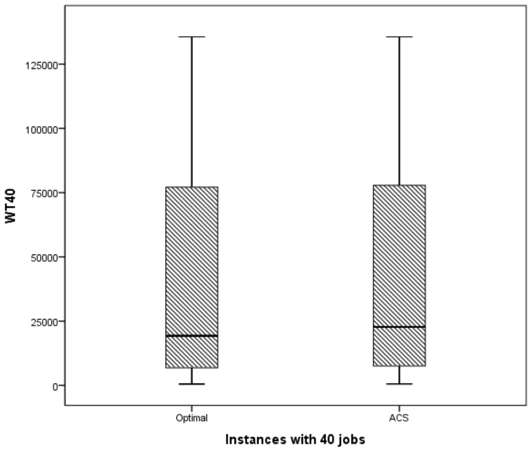


Fig. 9. Boxplot for WT with 40 jobs

Considering a significance level α=5%, it is possible to conclude that, in general the difference in performance proposed ACS and optimal solution is not statistically significant. A paired-samples t-test indicated that, with a confidence level of 95%, there is no statistically significant difference between performance of ACS (M = 48607.76, SD=46928.024), and optimal solutions related on literature (M=43494.40, SD= 44745.846) t(24) = -1, p=.327. These results could suggest that ACS have been effective on WT resolution for 40 jobs, considering that does not exist statistical evidence that its performance is significantly different than optimal solutions for the same instances for WT with 40 jobs.

Table 12. Statistical Sampling Summary based on WT40

	Optimal	**ACS**
Mean	43494,40	48607,76
Median	19312,00	26914,00
Variance	2002190770,083	2202239475,023
Std. Deviation	44745,846	46928,024
Interquartile Range	70854	71289
Skewness	,791	,634
Kurtosis	-,737	-1,073

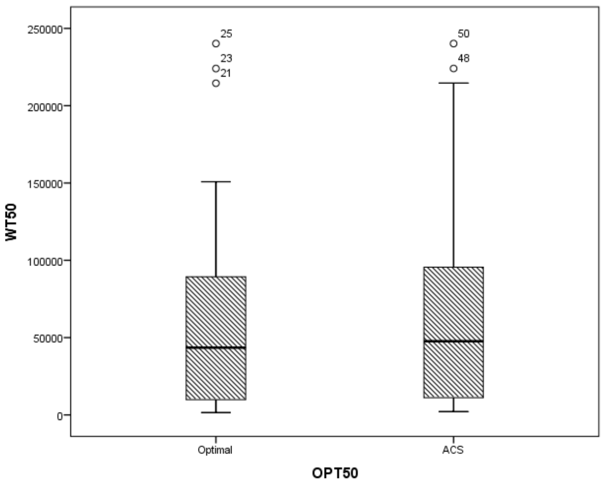


Fig. 10. Boxplot for WT with 50 jobs

Table 13. Statistical Sampling Summary based on WIT50

	Optimal	**ACS**
Mean	67110,40	70139,44
Median	43504,00	47627,00
Variance	5296763112,167	5234325587,590
Std. Deviation	72778,864	72348,639
Interquartile Range	87856	91550
Skewness	1,266	1,179
Kurtosis	,724	,564

From boxplot, Figure 10, analysis we can conclude that there are outliers or extreme values and the analysis of location, dispersion and asymmetry of data, making its synthesis by ACS, and optimal solutions, that ACS has been effective on the resolution of WT considering that its obtained mean values were similar to the optimal solutions (in some instances the best known). It is not clear, from the graph analysis, the existence of significant difference of the performance of ACS related with optimal solutions.

From the analysis of statistical sampling summary based on WT minimization (Table 13), it is possible to conclude that exist some statistic evidence of the difference significance between optimal solutions and ACS performance on WT resolution for 50 jobs. This conclusion can be supported either by central tendencies and dispersion measures. This evidence can be observed even on median and dispersion indicators. Regarding variability, through standard deviation and interquartile range analysis it possible to conclude that ACS presents different variability.

Considering a significance level α=5%, it is possible to conclude that, in general the difference in performance proposed ACS and optimal solution is significant.

A paired-samples t-test indicated that, with a confidence level of 95%, there is statistically significant difference between performance of ACS (M=70139.44, SD=72348.64), and optimal solutions related on literature (M= 67110.40, SD= 72778,86) t(24) = -4,652, p<0.001. These results suggest that ACS performance has decreased on WT resolution for 50 jobs when compared with 40 jobs considering that exist statistical evidence that its performance is significantly different than optimal solutions for the same instances for WT with 50 jobs.

The boxplot from Figure 11 depicts the values of obtained solutions by ACS on the WT resolution of 25 instances in analysis with 100 jobs. From the boxplot analysis it is possible to conclude about difference of the performance of ACS related with optimal solutions.

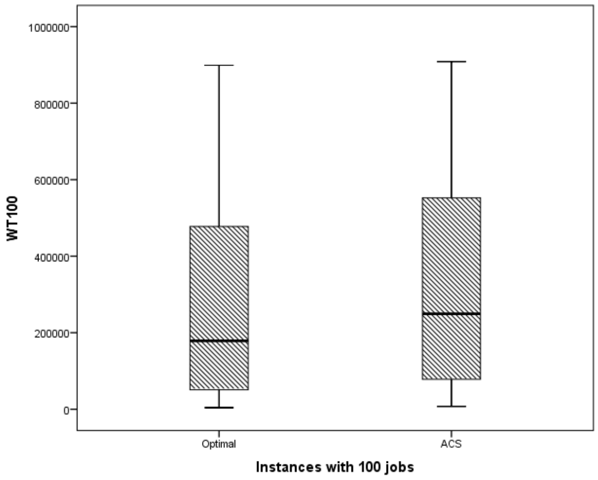


Fig. 11. Boxplot for WT with 100 jobs

Table 14. Statistical Sampling Summary based on WT100

	Optimal	**ACS**
Mean	268357,52	302118,04
Median	178840	249518
Variance	69219509079,2	70906890332,5
Std. Deviation	263096	266283,5
Interquartile Range	462725	480355
Skewness	,809	,594
Kurtosis	-,337	-,711

From the analysis of statistical sampling summary based on WT minimization (Table 14), it is not possible to conclude about statistic evidence of the difference significance between optimal solutions and ACS performance on WT resolution for 100 jobs. This conclusion can be supported either by central tendencies and dispersion measures. This evidence can be observed even on median and dispersion indicators. Regarding variability, through standard deviation and interquartile range analysis it possible to conclude about differences on performance by ACS when comparing with optimal solutions.

Considering a significance level α=5%, it is possible to conclude that, in general the difference in performance proposed ACS and optimal solution is statistically significant. A paired-samples t-test indicated that, with a confidence level of 95%, there is statistically significant difference between performance of ACS (M=302118,04, SD=266283,5), and optimal solutions related on literature (M= 268357,5, SD= 263096,01) t(24) = -5,959, p<0.001. These results suggest that exist statistical evidence that its performance is significantly different than optimal solutions for the same instances for WT with 100 jobs.

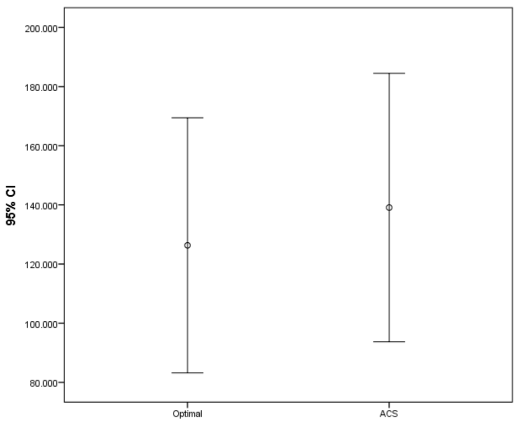


Fig. 12. Error Bar of the WT for all instances

Table 15. Statostical Sampling Summary based on WT

	Optimal	**ACS**
Mean	126320,77	139080,95
Median	53208,00	75267,00
Variance	35134539958,2	38921816643,2
Std. Deviation	187442,1	197286,1
Interquartile Range	142301	139647
Skewness	2,241	2,045
Kurtosis	4,912	3,764

Previous computational and statistical tests indicate that ACS presents effectiveness on the resolution of WT for 40 jobs, but its performance decreases for 50 and 100 jobs, which can indicate that parameter tuning, must be improved in order to increase ACS performance for those class of instances. Following, in order to evaluate the overall performance of ACS on the resolution of 75 instances of WT scheduling problem with 40, 50 and 100 jobs, additional statistical analysis will be conducted to compare the overall performance of ACS when compared with respective optimal solutions.

From the analysis of statistical sampling summary based on WT minimization (Table 15), it is possible to conclude that exist statistic evidence of the difference on the ACS performance of WT resolution. This conclusion can be supported either by central tendencies and dispersion measures. This evidence can be observed even on median and dispersion indicators. Regarding variability, through standard deviation and interquartile range analysis it possible to conclude that ACS presents similar variability.

Considering a significance level $\alpha=5\%$, it is possible to conclude that, in general the difference in performance proposed ACS and optimal solution is statistically significant. A paired-samples t-test indicated that, with a confidence level of 95%, there is statistically significant difference between performance of ACS (M= 139080.95, SD=197286.13), and optimal solutions related on literature (M=126320.77, SD= 187442.1) t(74) = -5.000, $p<0.001$. These results suggest that exist statistical evidence that its performance is significantly different than optimal solutions for the same WT instances with 100 jobs.

From the obtained results we can conclude that proposed ACS and its parameterization is adequate for 40 jobs instances considering that obtained results indicate that the difference in performance of ACS and optimal solution is not statistically significant. For instances with greater dimensions parameter tuning must be increased, considering a degradation of computational time.

7 Conclusions and Future Work

In this paper, we described some guidelines for ACS based software developing tools to study the effectiveness and efficiency of ACS in the optimization of total weighted tardiness for SMSP. More than developing algorithms with unquestionable practice utility, the main purpose of this paper was to illustrate, through more simple scheduling problems, the potential effectiveness and efficiency of using MH approaches, with special emphasis on SI based techniques for scheduling problem solving. The obtained results show that proposed ACS algorithm was effective for the instances studied, being possible to find good solutions in short time, i.e., a few CPU seconds. As future work, we will extend our ACS algorithm to the job-shop scheduling problem, where the jobs are distributed across different machines.

Acknowledgments. This work is supported by FEDER Funds through the "Programa Operacional Factores de Competitividade - COMPETE" program and by National Funds through FCT "Fundação para a Ciência e a Tecnologia" under the project: FCOMP-01-0124-FEDER-PEst-OE/EEI/UI0760/2011 and PTDC/EME-

GIN/109956/2009. . This work is partially supported in the framework of the IT4 Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 by operational programme 'Research and Development for Innovations' funded by the Structural Funds of the European Union and state budget of the Czech Republic, EU.

References

1. Baker, K.R.: Introduction to Scheduling, Brussels, vol. 32 (1992)
2. Baker, K.R., Trietsch, D.: Optimization methods for the single machine probelm. In: Principles of Sequencing and Scheduling, 1st edn., pp. 34–56. Wiley, New York (2009)
3. Dorigo, M.: Swarm Intelligence, vol. (4). Springer, New York (2007)
4. Dorigo, M., Birattari, M., Stützle, T.: Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique. IEEE Computational Intelligence Magazine (2006)
5. Lawer, E.L.: A pseudopolinomial algorithm for sequencing Jobs to Minimize Total Tardiness. Annals of Discrete Mathematics, 331–342 (1997)
6. Reynolds, R.G.: An Introduction to Cultural Algorithms. In: Proceedings of the 3rd Annual Conference on Evolutionary Programming, pp. 131–139. World Scienfific Publishing (1994)
7. Merkle, D., Middendorf, M.: On solving permutation scheduling problems with ant colony optimization. International Journal of Systems Science 36(5), 255–266 (2005)
8. Liao, C., Juan, H.: An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. Computers & Operations Research 34, 1899–1909 (2007)
9. Yagmahan, B., Yenisey, M.M.: Ant colony optimization for multi-objective flow shop scheduling problem. Computers & Industrial Engineering 54, 411–420 (2008)
10. Anghinolfi, D., Paolucci, M.: A new ant colony optimization approach for the single machine total weighted tardiness scheduling problem. International Journal of Operations Research 5(1), 1–17 (2008)
11. Srinivasa Raghavan, N.R., Venkataramana, M.: Parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization. The International Journal of Advanced Manufacturing Technology 41(9-10), 986–996 (2009)
12. den Besten, M., Stützle, T., Dorigo, M.: Ant colony optimization for the total weighted tardiness problem. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.-P. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 611–620. Springer, Heidelberg (2000)
13. Dorigo, M., Gambardella, L.M.: Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
14. Dorigo, M.: Optimization, Learning and Natural Algorithms, PhDThesis, Politecnico di Milano, Italy, in Italian (1992)
15. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press (2004)
16. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Dipartimentodi Elettronica, Politecnico di Milano, Italy, Tech. Rep. 91-016 (1991)
17. Dorigo, M., Gambardella, L.M.: Ant colonies for the traveling salesman problem. BioSystems 43(2), 73–81 (1997)
18. Gambardella, L.M., Dorigo, M.: Solving symmetric and asymmetric TSPs by ant colonies. In: Baeck, T., et al. (eds.) Proc. 1996 IEEE International Conference on Evolutionary Computation (ICEC 1996), pp. 622–627. IEEE Press, Piscataway (1996)

19. Stützle, T., et al.: Parameter Adaptation in Ant Colony Optimization, IRIDIA, Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2010-002 (January 2010)
20. El-Ghazali Talbi, Metaheuristics – From Design to Implementation. Wiley (2009)
21. Madureira, A., Pereira, I., Falcão, D.: Ant Colony System Based Approach to Single Machine Scheduling Problems — Weighted Tardiness Scheduling Problem. In: International Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC 2012), México, de November 5-9 (2012)
22. Pirlot, M.: General Local Search Method. European Journal of Operational Research 92, 493–522 (1996)
23. OR-Library - `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`

Multiobjective Optimization of Green Sand Mould System Using Chaotic Differential Evolution

T. Ganesan[1], I. Elamvazuthi[2], Ku Zilati Ku Shaari[1], and P. Vasant[3]

[1] Department of Chemical Engineering
[2] Department of Electrical & Electronics Engineering
[3] Department of Fundamental & Applied Sciences
Universiti Teknologi Petronas, 31750 Tronoh, Perak, Malaysia
tim.ganesan@gmail.com

Abstract. Many industrial optimization cases present themselves in a multi-objective (MO) setting (where each of the objectives portrays different aspects of the problem). Therefore, it is important for the decision-maker to have a solution set of options prior to selecting the best solution. In this work, the weighted sum scalarization approach is used in conjunction with three meta-heuristic algorithms; differential evolution (DE), chaotic differential evolution (CDE) and gravitational search algorithm (GSA). These methods are then used to generate the approximate Pareto frontier to the green sand mould system problem. The Hypervolume Indicator (HVI) is applied to gauge the capabilities of each algorithm in approximating the Pareto frontier. Some comparative studies were then carried out with the algorithms developed in this work and that from the previous work. Analysis on the performance as well as the quality of the solutions obtained by these algorithms is shown here.

Keywords: multi-objective (MO), industrial optimization, green sand mould system, weighted sum approach, differential evolution (DE), chaotic differential evolution (CDE), gravitational search algorithm (GSA), Hypervolume Indicator (HVI), approximate Pareto frontier.

1 Introduction

In recent times, many concerns have been raised when dealing with emerging technologies in engineering optimization which present themselves in a multi-objective (MO) setting [1], [2]. Strategies in MO optimization can be rudimentarily classified into two groups. First being methods that use the concept of Pareto-optimality to trace the non-dominated solutions at the Pareto curve, for instance in; Zitzler and Thiele's [3] Strength Pareto Evolutionary Algorithm (SPEA) and Non-dominated Sorting Genetic Algorithm II (NSGA-II) by Deb *et al* [4]. The second class of techniques is known as the weighted (or scalarization) techniques. During the application of these methods, the objective functions are aggregated into a single weighted function which is then solved for various scalar (weight) values. Some well-known scalarization techniques include the Weighted Sum method [5], [6], Goal Programming [7] and Normal-Boundary Intersection method (NBI) [8]. Using these techniques, the scalars (or weights) are used to consign relative trade-offs to the objectives during the aggregation procedure. Hence, alternative near-optimal solution options are generated for various values of the scalars. See [9], [10] and [11] for detail investigations and explanations on MO techniques in engineering optimization.

In MO optimization problems, determining the most efficient solution set can be a very daunting process. Many varieties of concepts (such as; diversity and convergence) have been proposed in the last years. These ideas were then used as indicators to evaluate the solution set produced by the optimization algorithm [12]. Such evaluations were then used to benchmark the algorithm's performance. These concepts unfortunately could not absolutely state and rank the superiority of solution sets produced by an algorithm against other such sets by other algorithms. The only concept that can be used generally for the overall ranking of solution sets is the idea of 'Pareto-dominance'. The Hypervolume Indicator (HVI) [13] is a set measure reflecting the volume enclosed by a Pareto front approximation and a reference set (see [14], [15], and [16]). The HVI thus guarantees strict monotonicity regarding Pareto dominance [17], [18]. This makes the ranking of solution sets and hence algorithms possible for any given MO problem.

This work aims to produce a set of solutions that dominantly approximates the Pareto frontier in the objective space of the green sand mould system. This problem was presented and attempted by using of genetic algorithm (GA) and particle swarm optimization (PSO) techniques (using weighted sum scalarization method) in Surekha *et al* [19].

In green sand mould systems, the quality of the product obtained from the moulding process is very dependent on the physical properties of the moulding sand (such as; hardness, permeability, green compression strength and bulk density). Faulty extent of the mentioned properties may result in casting defects such as; poor surface finish, blowholes, scabs, pinhole porosity, etc. Controllable variables such as; percentage of water, percentage of clay, grain fineness number and number of strokes heavily influence the physical properties of the moulded sand. Therefore, by classifying these parameters as the decision variables and the mould sand properties as the objective function, the MO optimization problem was formulated in Surekha *et al* [19]. The purpose of this formulation is for the determination of the best controllable parameters for optimal final-product of the moulding process. A more rigorous study on the optimization and model development of mould systems can be seen in [20] and [21].

In this work, the green mould sand system problem was tackled using Differential Evolution (DE) [22] and Gravitational Search Algorithm (GSA) [23] in conjunction with the weighted sum approach to generate a series of solutions that dominantly approximate the Pareto frontier. The dominance ranking among the frontier approximations produced by the algorithms were carried out using the Hypervolume Indicator (HVI) metric [13], [24]. Comparison studies were then conducted on the individual best solutions as well as the frontier approximations obtained in this work against those obtained in Surekha *et al* [19].

Generic Search Algorithm (GSA) introduced recently by E.Rashedi *et al* [23] is currently among the most applied meta-heuristic techniques in industrial optimization. GSA belong to the group of swarm-based stochastic search methods (such as; Particle Swarm Optimization (PSO) [25] and Cuckoo Search Algorithm (CSA) [26]). GSA operates on a population of solutions based on Newtonian law of gravity and mass interactions. This algorithm regards agents as objects consisting of different masses. In recent times, GSA has been broadly applied in many industrial settings (see [27]).

Differential Evolution (DE) population-based evolutionary algorithm that has been derived from Genetic Algorithms (GA)[28]. DE was developed in the nineties by Storn and Price [22]. DE has been used extensively to solve problems which are non-differentiable, non-continuous, non-linear, noisy, flat, multidimensional, have many local minima, constraints or high degree of stochasticity. Lately, DE has been applied to a variety of areas including optimization problems in chemical and process engineering [29],[30],[31].

This paper is organized as follows. In Section 2 of this paper, the computational techniques are presented. In Section 3 the HVI metric is discussed and this is followed by the description on the green mould sand MO problem in Section 4. Section 5 discusses computational results and finally, the concluding remarks are given in Section 6.

2 Computational Techniques

2.1 Differential Evolution (DE)

DE is a class of evolutionary meta-heuristic algorithms first introduced in 1995 by Storn and Price [22]. This core idea of this technique is the incorporation of perturbative methods into evolutionary algorithms. DE starts by the initialization of a population of at least four individuals denoted as *P*. These individuals are real-coded vectors with some size *N*. The initial population of individual vectors (the first generation denoted *gen* = 1) are randomly generated in appropriate search ranges. One principal parent denoted x^p_i and three auxiliary parents denoted x^a_i is randomly selected from the population, *P*. In DE, every individual, *I* in the population, *P* would become a principle parent, x^p_i at one generation or the other and thus have a chance in mating with the auxiliary parents, x^a_i. The three auxiliary parents then engage in 'differential mutation' to generate a mutated vector, V_i.

$$V_i = x^a_1 + F(x^a_2 - x^a_3) \tag{1}$$

where *F* is the real-valued mutation amplification factor which is usually between 0 and 1. Next V_i is then recombined (or exponentially crossed-over) with x^p_i to generate child trial vector, x^{child}_i. The probability of the cross-over, CR is an input parameter set by the user. In DE, the survival selection mechanism into the next generation is called 'knock-out competition'. This is defined as the direct competition between the principle parent, x^{p}_{i} and the child trial vector, x^{child}_{i} to select the survivor of the next generation as follows:

$$x_i(gen+1) = \begin{cases} x_i^{child}(gen) \leftrightarrow & f(x_i^{child})\ betterthan\ f(x_i^{p}) \\ x_i^{p}(gen) \leftrightarrow & otherwise \end{cases} \tag{2}$$

Therefore, the knock-out competition mechanism also serves as the fitness evaluation scheme for the DE algorithm. The parameter setting for the DE algorithm is given in Table 1: The algorithm of the DE method is shown in Algorithm 1.

Table 1. DE Parameter Setting

Parameters	Values
Individual Size, *N*	6
Population Size, *P*	7
Mutation amplification factor, *F*	0.3
Cross-over Probabbility, *CR*	0.667

Algorithm 1. Differential Evolution (DE)

```
Step 1: Initialize individual size N, P, CR and F
Step 2: Randomly initialize the population vectors, $x^{G}_{i.}$
Step 3: Randomly select one principal parents, $x^{P}_{i}$
Step 4: Randomly select three auxilary parents, $x^{a}_{i}$
Step 5: Perform differential mutation & generate mutated vector, $V_i$
Step 6: Recombine $V_i$ with $x^{P}_{i}$ to generate child trial vector, $x^{child}_{i}$
Step 7: Perform 'knock-out' competition for next generation survival selection
Step 8: If the fitness criterion is satisfied and $t= T_{max}$ , halt and print solutions
else proceed to step 3
```

2.2 Gravitational Search Algorithm (GSA)

The GSA algorithm is a meta-heuristic algorithm first developed in 2009 by E.Rashedi *et al*[23]. This technique was inspired by the law of gravity and the idea of interaction of masses. This algorithm uses the Newtonian gravitational laws where the search agents are the associated masses. Thus, the gravitational forces influence the motion of these masses, where lighter masses gravitate towards the heavier masses (which signify good solutions) during these interactions. The gravitational force hence acts as the communication mechanism for the masses (analogous to 'pheromone deposition' for ant agents in ACO [32] and the 'social component' for the particle agents in PSO [25]). The position of the masses correlates to the solution space in the search domain while the masses characterize the fitness space. As the iterations increase, and gravitational interactions occur, it is expected that the masses would conglomerate at its fittest position and provide an optimal solution to the problem.

Initially the GSA algorithm randomly generates a distribution of masses, $m_i(t)$(search agents) and also sets an initial position for these masses, x_i^d. For a minimization problem, the least fit mass, $m_i^{worst}(t)$ and the fittest mass, $m_i^{best}(t)$ at time t are calculated as follows:

$$m^{best}(t) = \min_{j\in[1,N]} m_j(t) \tag{3}$$

$$m^{worst}(t) = \max_{j\in[1,N]} m_j(t) \tag{4}$$

For a maximization problem, its simply vice versa. The inertial mass, $m_i'(t)$ and gravitational masses, $M_i(t)$ are then computed based on the fitness map developed previously.

$$m_i'(t) = \frac{m_i(t) - m^{worst}(t)}{m^{best}(t) - m^{worst}(t)} \tag{5}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{6}$$

such that,

$$M_{ai} = M_{pi} = M_{ii} = M_i : i \in [1, N] \tag{7}$$

Then the gravitational constant, $G(t+1)$ and the Euclidean distance $R_{ij}(t)$ is computed as the following:

$$G(t+1) = G(t)\exp\left(\frac{-\alpha t}{T_{\max}}\right) \tag{8}$$

$$R_{ij}(t) = \sqrt{(x_i(t))^2 - (x_j(t))^2} \tag{9}$$

where α is some arbitrary constant and T_{max} is the maximum number of iterations, $x_i(t)$ and $x_j(t)$ are the positions of particle i and j at time t. The interaction forces at time t, $F_{ij}^d(t)$ for each of the masses are then computed:

$$F_{ij}^{d}(t) = G(t)\left(\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}\right) \times \left(x_j^d(t) - x_i^d(t)\right) \tag{10}$$

where ε is some small parameter. The total force acting on each mass *i* is given in a stochastic form as the following:

$$F_i^d(t) = \sum_{\substack{j=1 \\ i \neq j}}^{N} rand(w_j) F_{ij}^d(t) : rand(w_j) \in [0,1] \tag{11}$$

where *rand(w$_j$)* is a randomly assigned weight. Consequently, the acceleration of each of the masses, $a_i^d(t)$ is then as follows:

$$a_i^d(t) = \left(\frac{F_i^d(t)}{M_{ii}(t)}\right) \tag{12}$$

After the computation of the particle aceleration, the particle position and velocity is then calculated:

$$v_i^d(t+1) = rand(w_j) + v_i^d(t) + a_i^d(t) \tag{13}$$

$$x_i^d(t+1) = x_i^d(t(t) + v_i^d(t(t+1) \tag{14}$$

where *rand(w$_j$)* is a randomly assigned weight. The iterations are then continued until the all mass agents are at their fittest positions in the fitness landscape and some stopping criterion which is set by the user is met. The GSA algorithm is presented in Algorithm 2 and the parameter settings are given in Table 2.

Table 2. GSA Parameter Setting

Parameters	Values
Initial parameter (G_o)	100
Number of mass agents, n	6
Constant parameter, α	20
Constant parameter, ε	0.01

Algorithm 2. Gravitational Search Algorithm (GSA)

```
Step 1: Initialize no of particles, $m_i$ and initial positions, $x_i(0)$
Step 2: Initialize algorithm parameters $G(0)$, $\alpha$.
Step 3: Compute gravitational & inertial masses based on the fitness map
Step 4: Compute the gravitational constant, $G(t)$
Step 5: Compute distance between agents, $R_{ij}(t)$
Step 6: Compute total force, $F_i^d(t)$ and the acceleration $a_i^d(t)$ of each agent.
Step 7: Compute new velocity $v_i(t)$ and position $x_i(t)$ for each agent
Step 8: If the fitness criterion is satisfied and $t = T_{max}$, halt and print solutions
        else proceed to step 3
```

2.3 Chaotic Differential Evolution (CDE)

Specific modifications were performed in the DE algorithm to enhance its diversification capabilities by the addition of the chaotic component. First, the population of vectors, x^G_i was generated. The consequence steps are similar to the regular DE algorithm where one principal parent, x^p_i and three auxiliary parents x^a_i are randomly selected. Differential mutation is then performed and the mutated vector, V_i is generated. The V_i is then recombined with x^p_i to generate child trial vector, x^{child}_i. The obtained x^{child}_i is used as the input to the chaotic logistic map (Jakobson, 1981 [33]). This chaotic logistic mapping is presented as follows:

$$N_i(t) = x_i^{child}(t) \tag{15}$$

$$R_i(t) = \lambda N_i(t) \tag{16}$$

$$N_i(t+1) = R_i(t) N_i(t)\left[1 - N_i(t)\right] \tag{17}$$

$$R_i(t+1) = R_i(t) + \lambda' \tag{18}$$

where $N(t)$ and $R(t)$ are variables in the logistic chaotic map , λ' and λ are relaxation constants specified by the user. Then the logistic mapping is contined until a specific number of iteration is satisfied. The final value at maximum number of iteration of $N(t_{max})$ is incorporated into the child trial vector, x^{child}_i. Hence, the child trial vector, x^{child}_i undergoes another round of mutation by the chaotic map. Next, the 'knock-out' competition for next generation survival selection is performed. The fitness function for the child trial vector, x^{child}_i is evaluated. Thus, another variant of the DE algorithm with enhanced diversification capabilities was developed. In this work, this algorithm is called the Chaotic DE (CDE). The execution steps for the CDE techniques is given in Algorithm 3.

Algorithm 3. Chaotic Differential Evolution (CDE)

```
Step 1: Initialize population size.
Step 2: Randomly initialize the population vectors, $x^G_{i.}$
Step 3: Randomly select one principal parents, $x^p_i$
Step 4: Randomly select three auxilary parents, $x^a_i$
Step 5: Perform differential mutation & generate mutated vector, $V_i$
Step 6: Recombine $V_i$ with $x^p_i$ to generate child trial vector, $x^{child}_i$
Step 7: Iterate chaotic logistic map.
Step 8: If $n > N_{max}$, proceed to next step
        else go to Step 7.
Step 9: Evaluate fitness of the new $x^{child}_{i.}$
Step 10: If the fitness criterion is satisfied halt and print solutions
        else proceed to step 3
```

The parameter settings specified in the CDE algorithm is as in Table 3:

Table 3. CDE Parameter Setting

Parameters	Values
Individual Size, N	6
Population Size, P	7
Mutation amplification factor, F	0.15
Max. No. of Evaluations , T_{max}	3000
Constant, λ , λ' , N_{max}	5,0.01, 400

3 Hypervolume Indicator

The Hypervolume Indicator (HVI) is the only strictly Pareto-compliant indicator that can be used to measure the quality of solution sets in MO optimization problems [13], [24]. Strictly Pareto-compliant can be defined such that if there exists two solution sets to a particular MO problem, then the solution set that dominates the other would a higher indicator value. The HVI measures the volume of the dominated section of the objective space and can be applied for multi-dimensional scenarios. When using the HVI, a reference point needs to be defined. Relative to this point, the volume of the space of all dominated solutions can be measured. The HVI of a solution set $x_d \in X$ can be defined as follows:

$$HVI(X) = vol\left(\bigcup_{(x_1,..x_d)\in X} [r_1, x_1] \times ... \times [r_d, x_d] \right) \tag{19}$$

where $r_{1,...,}r_d$ is the reference point and *vol*(.) being the usual Lebesgue measure. In this work the HVI is used to measure the quality of the approximation of the Pareto front by the GSA and the DE algorithms when used in conjunction with the weighted sum approach.

4 Application Data

In the green sand mould system, the response parameters of the mould heavily influence the quality of the final product. In Surekha *et al* [19], these parameters are selected as the objective functions. The responses parameters are; green compression strength (f_1), permeability (f_2), hardness (f_3) and bulk density (f_4). These objectives on the other hand are influenced by on the process variables which are; the grain fineness number (A), percentage of clay content (B), percentage of water content (C) and number of strokes (D). The objective functions and the range of the decision variables are shown as follows:

$$\begin{aligned} f_1 &= 17.2527 - 1.7384A - 2.7463B + 32.3203C + 6.575D + 0.014A^2 + 0.0945B^2 - 7.7857C^2 \\ &- 1.2079D^2 + 0.0468AB - 0.1215AC - 0.0451AD + 0.5516BC + 0.6378BD + 2.689CD) \end{aligned} \tag{20}$$

$$\begin{aligned} f_2 &= 1192.51 - 15.98A - 35.66B + 9.51C - 105.66D + 0.07A^2 + 0.45B^2 - 4.13C^2 \\ &+ 4.22D^2 + 0.11AB + 0.2AC + 0.52AD + 1.19BC + 1.99BD - 3.1CD \end{aligned} \tag{21}$$

$$\begin{aligned} f_3 &= 38.2843 - 0.0494A + 2.4746B + 7.8434C + 7.774D + 0.001A^2 \\ &- 0.00389B^2 - 1.6988C^2 - 0.6556D^2 - 0.0015AB + 0.0151AC - 0.0006AD \\ &- 0.075BC - 0.1938BD + 0.65CD \end{aligned} \tag{22}$$

$$\begin{aligned} f_4 &= 1.02616 + 0.01316A - 0.00052B - 0.06845C + 0.0083D - 0.00008A^2 \\ &+ 0.0009B^2 + 0.0239C^2 - 0.00107D^2 - 0.00004AB - 0.00018AC + 0.00029AD \\ &- 0.00302BC - 0.00019BD - 0.00186CD \end{aligned} \tag{23}$$

$$\begin{aligned} 52 &\le A \le 94 \\ 8 &\le B \le 12 \\ 1.5 &\le C \le 3 \\ 3 &\le D \le 5 \end{aligned} \tag{24}$$

To obtain the size distributions of the silica sand and the grain fineness number, sieve analysis tests were carried out in Parappagoudar *et al.* [34]. Similarly, the authors also conducted gelling index tests for the determination the strength of clay. Next, experiments were conducted by varying the combination of the parameters using the central composite design. The mathematical model of the green mould system was developed where; the objective functions as given in equations (21)-(23) and the constraints as given in equation (24). The MO optimization problem statement for the green mould system problem is shown as follows:

$$Max \quad (f_1, f_2, f_3, f_4) \qquad subject\ to$$
$$52 \le A \le 94$$
$$8 \le B \le 12$$
$$1.5 \le C \le 3$$
$$3 \le D \le 5 \qquad (25)$$

The algorithms used in this work were programmed using the C++ programming language on a personal computer (PC) with an Intel dual core processor running at 2 GHz.

5 Results and Analysis

In this work, the solution sets which are the approximations of the Pareto frontier were obtained using the DE and the GSA methods. The quality of these solutions was measured using the HVI. The nadir point (or the reference point) used in the HVI is a specific point where all the solutions sets produced by the algorithms dominate this point. The nadir point selected in this work is $(r_1, r_2, r_3, r_4) = (15, 50, 40, 1)$.

The individual solutions (for specific weights) of the GSA algorithm were gauged with the HVI and the best, median and worst solution was determined. The individual solutions for the GSA algorithm and their respective HVI values are shown in Table 4.

Table 4. Individual Solutions Generated by the GSA Algorithm

Description		Best	Median	Worst
Objective Function	f_1	34.6562	25.8028	25.5917
	f_2	205.619	210.839	210.877
	f_3	81.1777	78.8635	78.8113
	f_4	1.46948	1.47897	1.47937
Decision Variable	x_1	52.0015	52.0023	52.0009
	x_2	8.00543	8.00205	8.0007
	x_3	2.49446	1.51268	1.5
	x_4	3.19889	3.00254	3
HVI		59134.56	32342.82	31702.15

The associated weights (w_1, w_2, w_3, w_4) for the best, median and worst solution are (0.2, 0.3, 0.4, 0.1), (0.1, 0.6, 0.2, 0.1) and (0.1, 0.2, 0.5, 0.2). The computational time for the best, median and worst solution is 0.52, 0.13 and 0.04 seconds respectively. For the approximation of the Pareto frontier, 21 solutions for various weights were obtained for both the algorithms. The approximate Pareto frontiers obtained using the GSA algorithm is shown in Figure 1:

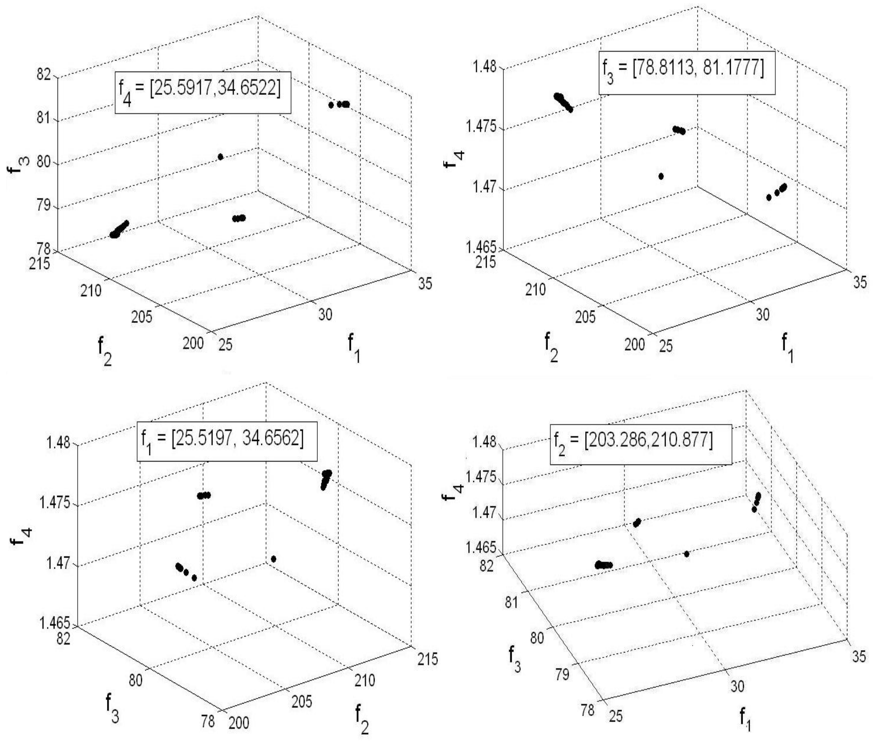


Fig. 1. Pareto frontiers of the objectives obtained by the GSA method

The individual solutions for the DE algorithm and their individual HVI values are given in Table 5.

Table 5. Individual Solutions Generated by the DE Algorithm

Description		Best	Median	Worst
Objective Function	f_1	50.2281	45.8468	39.4176
	f_2	137.769	137.19	137.105
	f_3	86.289	85.3638	83.7878
	f_4	1.50073	1.50953	1.52143
Decision Variable	x_1	53.0457	55.5626	59.4985
	x_2	9.39392	9.13418	8.74959
	x_3	2.99849	2.99044	2.99944
	x_4	4.39392	4.13418	3.74959
HVI		71665.77	62166.44	48561.85

The associated weights (w_1, w_2, w_3, w_4) for the best, median and worst solutions produced by the DE algorithm are (0.4, 0.2, 0.3, 0.1), (0.4, 0.2, 0.1, 0.3) and (0.2, 0.2, 0.1, 0.5). The computational time for the best, median and worst solution is 0.19, 0.42 and 6.02 seconds respectively. The approximate Pareto frontiers obtained using the DE algorithm is shown in Figure 2:

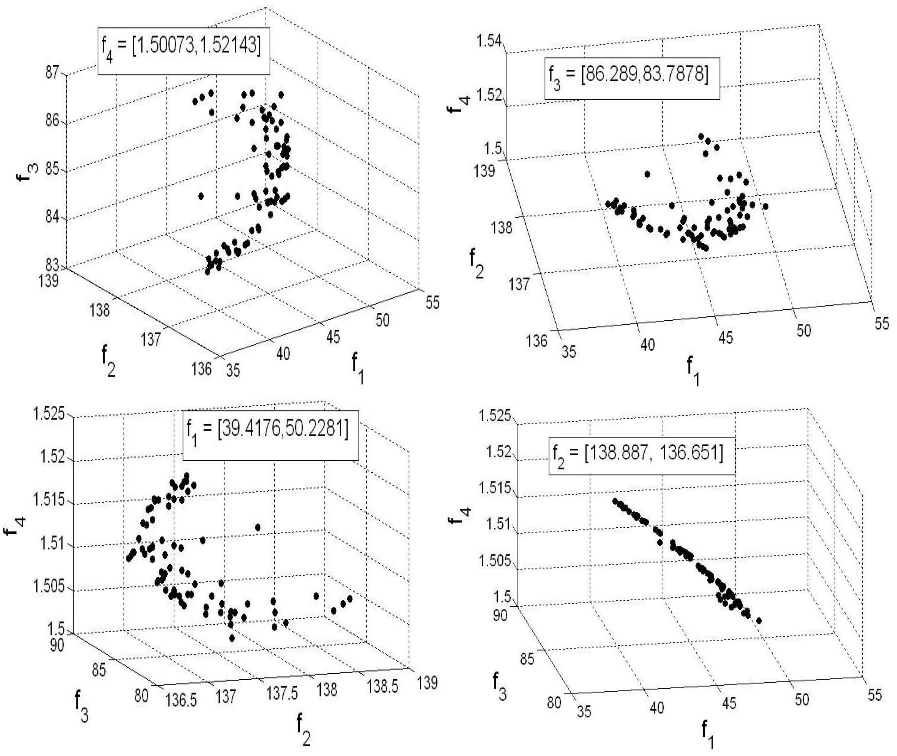


Fig. 2. Pareto frontiers of the objectives obtained by the DE method

The individual solutions for the CDE algorithm and their individual HVI values are given in Table 6.

Table 6. Individual Solutions Generated by the CDE Algorithm

Description		Best	Median	Worst
Objective Function	f_1	49.6548	42.1823	30.0356
	f_2	147.743	171.331	93.44
	f_3	86.0621	83.9258	81.5481
	f_4	1.49053	1.48441	1.62126
Decision Variable	x_1	52.0523	52.5328	53.3306
	x_2	9.29142	8.73265	9.86615
	x_3	2.84375	2.76542	1.62126
	x_4	4.29142	3.73264	4.86615
HVI		76534.697	70176.35	16859.13

The associated weights (w_1, w_2, w_3, w_4) for the best, median and worst solution are (0.2, 0.1, 0.2, 0.5), (0.3, 0.4, 0.1, 0.2) and (0.5, 0.1, 0.3, 0.1). The computational time for the best, median and worst solution is 2.94, 3.19 and 3.3 seconds respectively. The approximate Pareto frontiers obtained using the CDE algorithm is given in Figure 3.

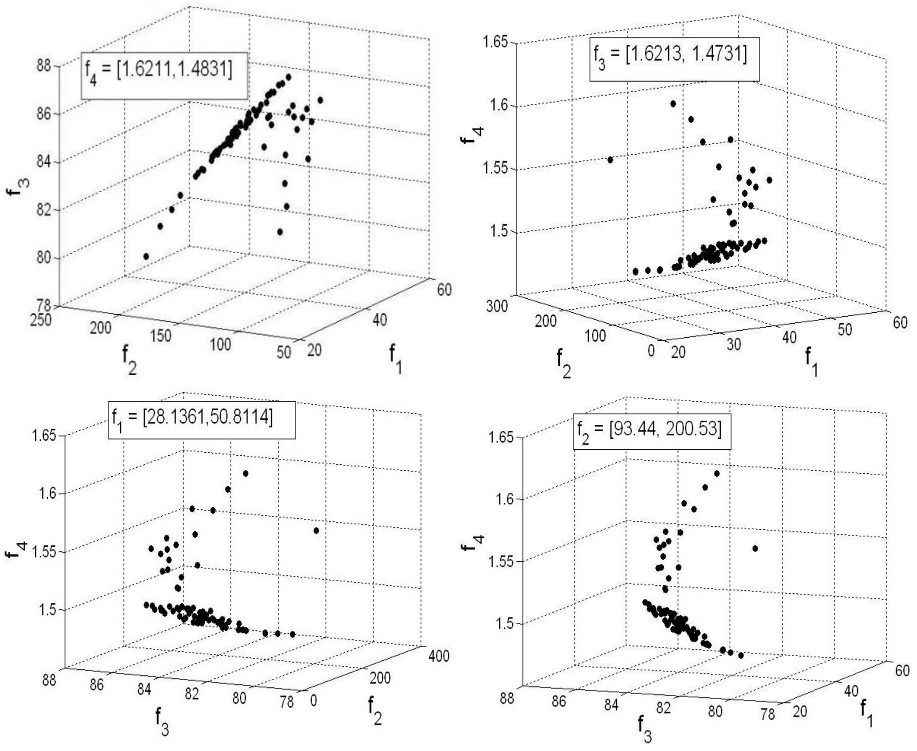


Fig. 3. Pareto frontiers of the objectives obtained by the CDE method

It can be observed using the HVI that the best solution obtained by CDE algorithm dominates the best solution produced by the DE and GSA algorithms by 6.79% and 29.42%. The comparison of the best candidate solutions obtained by the methods employed in this work against the PSO method in Surekha *et al* [19] is shown in Table 7.

In Table 7, it can be seen that the best solutions produces by CDE, DE and GSA algorithms are more dominant the PSO approach [19]. The HVI computed for the entire frontier of each solution set produced by an algorithm gives the true measure of dominance when compared with another algorithm. In this work, the HVI for the entire frontier was computed for each of the algorithm. The execution time for each algorithm to generate the entire frontier was also obtained. The HVI for the entire frontier for the solution sets produced by the PSO [19], DE, CDE as well as GSA and the associated execution time is shown in Table 8.

Table 7. The Comparison of the Best Solutions Obtained by the Algorithms

Description		PSO[19]	DE	GSA	CDE
Objective Function	f_1	55.411	50.228	34.656	49.655
	f_2	107.894	137.77	205.62	147.74
	f_3	84.794	86.289	81.178	86.062
	f_4	1.508	1.501	1.4695	1.491
Decision Variable	x_1	52.	53.046	52.002	52.052
	x_2	11.999	9.394	8.005	9.291
	x_3	2.845	2.998	2.494	2.844
	x_4	4.999	4.394	3.199	4.291
HVI		8876.72	71666	59134	76535

Table 8. The HVI Obtained by the Algorithms And the Computational Time For The Entire Frontier

	PSO [19]	DE	GSA	CDE
HVI	130, 108.48	492, 5326.52	309, 5494.45	529, 1120.41
Computational time (secs)	0.013	39.08	21.20	264.3

In Table 8, the frontier produced by the CDE algorithm is more dominant than the one produced by the GSA, DE and PSO [19] algorithms by 7.43%, 70.93% and 3966.7% respectively. The dominance ranking of the approximate Pareto frontiers produced by the algorithms is as follows:

$$CDE \prec DE \prec GSA \prec PSO[19] \tag{26}$$

where the symbol, $\prec$ is denoted as 'more dominant than'. A new optima is achieved by the CDE method (see Table 7) since it outperforms the DE, GSA and the PSO [19] methods. The individual best solution of the CDE method maximizes the objective f_2 while maintaining the objectives f_1 , f_3 and f_4 very effectively as compared to the one obtained by using DE. Thus, it can be said that the CDE method in this work outweighs the overall optimization capabilities of DE, GSA and PSO [19] (see HVI value in Table 8). In terms of computational time taken for the algorithm to produce the entire approximate Pareto frontier, the CDE method takes the longest time followed by the DE, GSA and the PSO [19] method respectively. Although the CDE algorithm produces the most dominant frontier, it sacrifices computational time as compared to the DE, GSA and PSO [19] methods.

In Surekha *et al* [19], the GA and the PSO method was used in conjunction with the Weighted Sum method on an Intel Pentium IV processor (single core). As mentioned previously, the algorithms presented in this work (CDE, DE and GSA), were executed on an Intel dual-core processor which is more superior than the machine used in Surekha *et al* [19]. However, it can be seen that the computational time for

CDE, DE and GSA algorithms are far greater as compared to the GA and PSO in [19]. Although the algorithms; CDE, DE and GSA were executed on a superior machine, these algorithms seem to be computationally inferior as compared to the GA and PSO [19] algorithms. This can be mainly attributed to the high complexity of the CDE, DE and GSA algorithms presented in this work.

In Surekha *et al* [19] as well as in this work the scalarization scheme adopted is the weighted sum method which produces a progression of the approximate Pareto-efficient solutions. As can be seen in Figure 2 and Figure 3, the frontiers produced by the DE and CDE algorithms are more uniform and diversely spaced as compared to the GSA algorithm in Figure 1. The frontier produced by the GSA algorithm seems to be conglomerated and localized at certain portions of the objective space. This spacing property as can be seen in this work heavily influences the ability of the algorithm to approximate the Pareto frontier. Localized solutions of the frontier such as the ones produced by the GSA algorithm miss out on certain solutions in the objective space. Therefore, this causes the GSA algorithm to have a lower HVI as compared to the CDE and DE algorithms. Hence, this causes the approximated Pareto frontier produced by the GSA algorithm to become less dominant as compared to the one produced by the DE and CDE algorithms.

One of the setbacks of the weighted sum method is that it **does not guarantee Pareto optimality** (only in the weak sense [35]). Besides, scalarization techniques (such as weighted sum as well Normal Boundary Intersection (NBI)) cannot approximate sections of the Pareto frontier that is **concave** [36]. Although the HVI metric is most effective in benchmarking the dominance of solution sets produced by algorithms, this metric is **very dependent on the choice of the nadir point**. The HVI metric's only weakness is in this aspect.

In this work, the CDE, DE and GSA algorithms performed **stable computations** during the program executions. All Pareto-efficient solutions produced by the algorithms developed in this work were **feasible** and no constraints were compromised. One of the advantages of using the DE and CDE algorithms as compared to the other algorithms used in this work is that it produces highly effective results in terms of approximating the Pareto frontier. However, although the DE and CDE algorithms performs well relative to algorithms used in this work, it can be clearly seen that the execution time is much higher than the one obtained by Surekha *et al* [19] using the PSO method. Since the DE and CDE methods are evolutionary-type algorithms, the diversification of the search space is high and thus resulting in high computational overhead as compared with the GSA and PSO methods (which are swarm-type algorithms). The DE method can be said to be the second best optimizer as compared to the CDE method

A new optima is achieved by the DE method (see Table 7) since it outperforms the GSA and the PSO [19] methods. Thus, it can be said that the DE method in this work outweighs the overall optimization capabilities of GSA and PSO [19] (see HVI value in Table 8). In terms of computational time taken for the algorithm to produce the entire approximate Pareto frontier, the DE method takes the longest time followed by the GSA and the PSO [19] method respectively. Although the DE algorithm produces the most dominant frontier, it sacrifices computational time as compared to the GSA and PSO [19] methods.

In this work, the DE and GSA algorithms performed stable computations during the program executions. All Pareto-efficient solutions produced by the algorithms developed in this work were feasible and no constraints were compromised. Although, the DE algorithm performs well relative to other algorithms used in this work, it can be clearly seen that the execution time is much higher than the one obtained by Surekha *et al.* [19] using the PSO method. Since DE is an evolutionary-type algorithm, the diversification of the search space is high and thus resulting in high computational time as compared with the GSA and PSO methods which are swarm-type algorithms. The GSA method can be said to be the second best optimizer as compared to the DE method. Besides, in comparison with the PSO algorithm the GSA method produces more superior results.

6 Concluding Remarks

In this work, a new local maximum and a more dominant approximation of the Pareto frontier was achieved using the CDE method. More Pareto-efficient solution options to the green mould system MO optimization problem were obtained. Besides, using the DE and CDE algorithms, the solution spread of the frontier was near-uniformly and diversely distributed. When gauged with the HVI metric, the DE and CDE algorithms produced the most dominant approximate of the Pareto frontier as compared to the GSA and the PSO [19] methods.

For future works, other meta-heuristic algorithms such as Genetic Programming (GP) [37], Analytical Programing (AP) [38], Hybrid Neuro-GP [39], MO evolutionary algorithm [40], [41] and Hybrid Neuro-Swarm [42] should be applied to the green mould system.

The solution sets obtained by these algorithms should be tested and evaluated using the HVI. The HVI metric should be tested using various nadir points for a better understanding of the effects of these points to the dominance measurements of solution sets.

During these numerical experiments, the spacing metric should be measured and compared for the observation of the uniformity of the spreads with respect to the algorithms. Besides, convergence and diversity metrics should also be utilized to compare the performance of the algorithms. More large-scale MO problems should be studied using the CDE, DE and GSA algorithms for a better understanding of the mentioned algorithm's performance and efficiency.

Acknowledgements. The authors would like to thank the Special Issue Editor and the anonymous referees whose suggestions had significantly improved this paper.

References

1. Eschenauer, H., Koski, J., Osyczka, A.: Multicriteria Design Optimization. Springer, Berlin (1990)
2. Statnikov, R.B., Matusov, J.B.: Multicriteria Optimization and Engineering. Chapman and Hall, New York (1995)

3. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms—A comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
5. Fishburn, P.C.: Additive Utilities with Incomplete Product Set: Applications to Priorities and Assignments. Operations Research Society of America (ORSA), Baltimore, MD, U.S.A (1967)
6. Triantaphyllou, E.: Multi-Criteria Decision Making: A Comparative Study, pp. 320–321. Kluwer Academic Publishers (now Springer), Dordrecht (2000)
7. Luyben, M.L., Floudas, C.A.: Analyzing the interaction of design and control. 1. A Multiobjective Framework and Application to Binary Distillation Synthesis, Computers and Chemical Engineering 18(10), 933–969 (1994)
8. Das, I., Dennis, J.E.: Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM Journal of Optimization 8(3), 631–657 (1998)
9. Eschenauer, H., Koski, J., Osyczka, A.: Multicriteria Design Optimization. Springer, Berlin (1990)
10. Sandgren, E.: Multicriteria design optimization by goal programming. In: Adeli, H. (ed.) Advances in Design Optimization, pp. 225–265. Chapman & Hall, London (1994)
11. Stanikov, R.B., Matusov, J.B.: Multicriteria Optimization and Engineering. Chapman and Hall, New York (1995)
12. Grosan, C.: Performance metrics for multiobjective optimization evolutionary algorithms. In: Proceedings of Conference on Applied and Industrial Mathematics (CAIM), Oradea (2003)
13. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
14. Knowles, J., Corne, D.: Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors. IEEE Transactions on Evolutionary Computation 7(2), 100–116 (2003)
15. Igel, C., Hansen, N., Roth, S.: Covariance Matrix Adaptation for Multi-objective Optimization. Evolutionary Computation 15(1), 1–28 (2007)
16. 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)
17. Fleischer, M.: The measure of Pareto optima. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 519–533. Springer, Heidelberg (2003)
18. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Grunert Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)
19. Surekha, B., Lalith, K.K., Panduy, A.K., Vundavilli, A.P.R., Parappagoudar, M.B.: Multi-objective optimization of green sand mould system using evolutionary algorithms. International Journal of Advance Manufacturing Technoloqy, 1–9 (2011)
20. Sushil, K., Satsangi, P.S., Prajapati, D.R.: Optimization of green sand casting process parameters of a foundry by using Taguchi method. International Journal of Advance Manufacturing Technology 55, 23–34 (2010)

21. Rosenberg, R.S.: Simulation of genetic populations with biochemical properties, Ph.D. thesis, University of Michigan (1967)
22. Storn, R., Price, K.V.: Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, ICSI, Technical Report TR-95-012 (1995)
23. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: A Gravitational Search Algorithm. Information Sciences 179, 2232–2248 (2009)
24. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
25. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE Proceedings of the International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)
26. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), pp. 210–214. IEEE Publications, USA (2009)
27. Chatterjee, A., Mahanti, G.K.: Comparative Performance of Gravitational Search Algorithm and Modified Particle Swarm Optimization Algorithm for Synthesis of Thinned Scanned Concentric Ring Array Antenna. Progress in Electromagnetics Research B 25, 331–348 (2010)
28. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, USA (1992)
29. Babu, B.V., Munawar, S.A.: Differential Evolution for the Optimal Design of Heat Exchangers. In: Proceedings of All-India seminar on Chemical Engineering Progress on Resource Development: A Vision 2010 and Beyond, Bhuvaneshwar (2000)
30. Babu, B.V., Singh, R.P.: Synthesis & Optimization of Heat Integrated Distillation Systems Using Differential Evolution. In: Proceedings of All- India seminar on Chemical Engineering Progress on Resource Development: A Vision 2010 and Beyond, Bhuvaneshwar (2000)
31. Angira, R., Babu, B.V.: Optimization of Non-Linear Chemical Processes Using Modified Differential Evolution (MDE). In: Proceedings of the 2nd Indian International Conference on Artificial Intelligence, Pune, India, pp. 911–923 (2005)
32. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed Optimization by Ant Colonies. In: Proceedings of the First European Conference of Artificial Intelligence, pp. 134–142. Elsevier Publishing, Paris (1991)
33. Jakobson, M.: Absolutely continuous invariant measures for one-parameter families of one-dimensional maps. Communications on Mathematical Physics 81, 38–39 (1981)
34. Parappagoudar, M.B., Pratihar, D.K., Datta, G.L.: Non-linear modeling using central composite design to predict green sand mould properties. Proceedings IMechE B Journal of Engineering Manufacture 221, 881–894 (2007)
35. Shukla, P.K.: On the Normal Boundary Intersection Method for Generation of Efficient Front. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007, Part I. LNCS, vol. 4487, pp. 310–317. Springer, Heidelberg (2007)
36. Zitzler, E., Knowles, J.D., Thiele, L.: Quality Assessment of Pareto Set Approximations. In: Branke, J., Deb, K., Miettinen, K., Słowiński, R. (eds.) Multiobjective Optimization. LNCS, vol. 5252, pp. 373–404. Springer, Heidelberg (2008)
37. Koza, J.R.: Genetic Programming: On the Programming of Computers by means of Natural Selection. MIT Press, USA (1992)
38. Zelinka, I.: Analytic programming by Means of SOMA Algorithm. In: Proc. 8th, International Conference on Soft Computing Mendel 2002, Brno, Czech Republic, pp. 93–101 (2002)

39. Ganesan, T., Vasant, P., Elamvazuthi, I.: Optimization of Nonlinear Geological Structure Mapping Using Hybrid Neuro-Genetic Techniques. Mathematical and Computer Modelling 54(11-12), 2913–2922 (2011)
40. Qu, B.Y., Suganthan, P.N.: Multi-objective evolutionary algorithms based on the summationof normalized objectives and diversified selection. Information Sciences 180, 3170–3181 (2010)
41. Li, K., Kwong, S., Cao, J., Li, M., Zheng, J., Shen, R.: Achieving Balance Between Proximity and Diversity in Multi-Objective Evolutionary Algorithm. Information Sciences 182, 220–242 (2011)
42. Elamvazuthi, I., Ganesan, T., Vasant, P.: A comparative study of HNN and Hybrid HNN-PSO techniques in the optimization of distributed generation (DG) power systems. International Conference on Advanced Computer Science and Information System (ICACSIS), 195–200 (2011)

Categorical Feature Reduction Using Multi Objective Genetic Algorithm in Cluster Analysis

Dipankar Dutta[1], Paramartha Dutta[2], and Jaya Sil[3]

[1] Department of Computer Science and Information Technology, University Institute of Technology, The University of Burdwan, Golapbug (North), Burdwan, West Bengal, PIN-713104, India
dipankar_dutta@rediffmail.com,
http://www.dipankarduttas.yolasite.com

[2] Department of Computer and System Sciences, Visva-Bharati University, Santiniketan, West Bengal, PIN-731235, India

[3] Department of Computer Science and Technology, Bengal Engineering and Science University, Shibpur, Howrah, West Bengal, PIN-711103, India

Abstract. In the paper, real coded multi objective genetic algorithm based K-clustering method has been studied, K represents the number of clusters. In K-clustering algorithm value of K is known. The searching power of Genetic Algorithm (*GA*) is exploited to search for suitable clusters and centers of clusters so that intra-cluster distance (Homogeneity, H) and inter-cluster distances (Separation, S) are simultaneously optimized. It is achieved by measuring H and S using Mod distance per feature metric, suitable for categorical features (attributes). We have selected 3 benchmark data sets from UCI Machine Learning Repository containing categorical features only.

The paper proposes two versions of *MOGA* based K-clustering algorithm. In proposed $MOGA(H,S)$, all features are taking part in building chromosomes and calculation of H and S values. In $MOGA\_Feature\_Selection(H,S)$, selected features take part to build chromosomes, relevant for clusters. Here, K-modes is hybridized with *GA*. We have used hybridized *GA* to combine global searching capabilities of *GA* with local searching capabilities of K-modes. Considering context sensitivity, we have used a special crossover operator called "pairwise crossover" and "substitution". The main contribution of this paper is simultaneous dimensionality reduction and optimization of objectives using *MOGA*.

Keywords: Clustering, homogeneity and separation, real coded multi objective genetic algorithm, dimensionality reduction, Pareto optimal front.

1 Introduction

In course of machine learning, machine learns from training data using learning mechanism that can be supervised, unsupervised or reinforced. In artificial intelligence, learning is "the improvement of performance in some environment through the acquisition of knowledge resulting from experience in that environment" [62]. Clustering is an example of unsupervised learning where class labels are not available at the training phase.

GA is "search algorithms based on the dynamics of natural selection and natural genetics" [33]. Categories of *GAs* are - simple *GA* (*SGA*) and multi objective *GA* (*MOGA*).

When an optimization problem involves only one objective, the task of finding the best solution is a single objective optimization problem. Integer programming, dynamic programming, geometric programming, stochastic programming and various other methods can solve these types of problems. However, in the real world, life appears to be quite complex. Most of the problems are consisting of more than one interdependent objective, which are to be minimized or maximized simultaneously. These types of problems are multi objective optimization problem [10]. The use of *GAs* in clustering is an attempt to exploit effectively the large search space usually associated with cluster analysis and better interactions among the features for forming chromosomes. Almost all conventional methods start searching from a single point and through subsequent iterations converge to the best solution. However, they are sensitive to the choice of initial solution. *GAs* always works on a whole population of points (strings) and perform a nearly global search rather than performing a local, hill-climbing search. This phenomena improves the chance of reaching to a global optima, vis-a-vis, reduces the risk of being trapped in a local optima, thereby offering robustness.

Clustering is to identify of natural groups within a data set. Instances in the same groups are more similar compared with instances in different groups [39,51,53,94]. From optimization viewpoint, clustering is a non-deterministic polynomial-time hard (NP-hard) grouping problem [25]. Evolutionary algorithms like *GAs* are metaheuristics widely believed to be effective on NP-hard problems, being able to provide near-optimal solutions to such problems in reasonable time. Although *GAs* have been used in data clustering problems [14,27,35,45,67,69,78,89], most of them optimized single objective, which is hardly equally applicable to all kinds of data sets. So to solve many real world problems like clustering, it is necessary to optimize more than one objective simultaneously by *MOGA*. As the relative importance of different clustering criteria are unknown, it is better to optimize (Homogeneity, H) and (Separation, S) separately rather than combining them into a single measure to be optimized. Objects within the same clusters are similar therefore, intra-cluster distances (H) are low and at the same time, objects within the different clusters are dissimilar, so inter-cluster distances (S) are high. In the paper, clustering is considered as an intrinsically multi objective optimization problem [39] by involving homogeneity and separation. However, choosing optimization criterion is one of the fundamental dilemmas in clustering [56]. All features are not equally important from clustering viewpoint. So researchers select relevant features by some feature selection techniques [64,86,92] and then did clustering with relevant features only. Although some researchers have done clustering by *MOGA* or by other multi objective evolutionary algorithms [4,13,24,31,40,61,63,68,75,76,81,82,83] none of them indeed tried simultaneous feature selection. In [54,55,71,72,74] researchers have done feather selection and clustering simultaneously. But our approach is different from them (explain later). $MOGA\_Feature\_Selection(H,S)$ automatically distinguish between relevant and irrelevant features for clustering, without taking help of feature selection techniques described in [64,86,92], which is the main contribution of this work. In our earlier work [18,19,20,21,22] we have done clustering by *MOGA*. Among these in [21,22] we have done simultaneous feature selection and clustering. This work is an extension of work we have reported in [21].

For ready reference meaning of acronyms used in this paper are listed in table 1.

Table 1. Acronyms and their meanings

Acronyms	Meanings
#	don't care condition in chromosomes
a_{ij}	Element of i^{th} row and j^{th} column of matrix (M)
A_i	i^{th} attribute of data set
C	Set of clusters
C_j	j^{th} cluster
CM_j	j^{th} cluster mode
CCh	Child chromosome
Ch_j	j^{th} chromosome
CP	Crossover point
$d(CM_i,t_j)$	Mod distance per feature between CM_i and t_j
$d(t_i,t_j)$	Mod distance per feature between t_i and t_j
$d_{mod}(CM_i,t_j)$	Modified Mod distance per feature between CM_i and t_j
$dom(A_i)$	domain of A_i
GAs	Genetic Algorithms
H	Homogeneity
H_i	Average intra cluster distance or Homogeneity of i^{th} cluster
H_{mod}	Modified homogeneity
ia	invalid attributes
IP	Initial population
IPS	Initial population size
K	Number of clusters
m	Number of tuples in data set
$M_{m\times 2}$	Two dimensional matrix
$MOGA$	Multi Objective Genetic Algorithm
N_{pch}	Number of parent chromosomes
N_{sb}	Number of substitution
NC_{pre}	Chromosomes obtained from previous method
N_{co}	Number of crossover
nva	Number of valid attributes
P_{co}	Crossover probability for GA
P_{sb}	Substitution probability for GA
PC_{pre}	Previous generation Pareto chromosomes
PCh	Parent chromosome
PS	Population Size
R	Relation
rn	random number
S	Separation
S_{mod}	Modified separation
SGA	Simple Genetic Algorithm
t_i	i^{th} tuple
v_i^t	value of i^{th} feature of t^{th} tuple
va	valid attributes
X	Set of tuple

This paper is organized as follows. Section 2 reports previous relevant works. Section 3 defines the problem. Section 4 describes proposed approach for solving the problem. In section 5 testing and comparison of five clustering algorithms are carried out on some popular benchmark data sets. Finally section 6 summarizes the work with concluding remarks.

2 Previous Work

Data mining is to extract knowledge from data [16,29,36,73,96]. Clustering is one of the most studied areas of data mining research. Extensive surveys of the clustering problems and algorithms are presented in [30,36,46,51,80,94]. With the growing popularity of soft computing methods, researchers of late extensively use intelligent approaches in cluster analysis of data sets.

The methods are broadly categorized as (i) Partitioning methods, (ii) Hierarchical methods, (iii) Density-based methods, (iv) Grid-based methods, (v) Model-based methods and (vi) Clustering by Genetic Algorithm (*GA*).

(i) Partitioning methods: Data sets are divided among K many partitions or clusters. $K \leq m$ where m is the number of objects or tuples in data sets. In this category most popular methods are K-means [65,66], fuzzy C-means (*FCM*) [5], K-modes [47,48], fuzzy K-modes [49] and K-medoids [53]. Among these K-means and *FCM* cannot be applied for categorical data, because mean does not work for categorical domains. K-modes and fuzzy K-modes are popular one in this domain.

(ii) Hierarchical methods: Hierarchical tree of clusters is constructed in this method. Such methods can be further classified into agglomerative or bottom up or merging [9,53] and divisive or top down or splitting [53]. To improve the quality of cluster formation, combination of hierarchical methods with other nonhierarchical methods are reported in [34,97]. Hierarchical agglomerative clustering algorithms (single, average and complete linkage) [51] are used for clustering categorical data.

(iii) Density-based methods: Arbitrary shaped clusters can be formed by density based clustering methods like *DBSCAN* [23], *DENCLUE* [43] and *OPTICS* [2].

(iv) Grid-based methods: In this approach a grid structure is formed from data sets on which different clustering operations are performed. Few important algorithms are *STRING* [95], WaveCluster used wavelet transform method [90] and *CLIQUE*, combination of grid and density based approaches [1].

(v) Model-based methods: These methods try to optimize the fit between the given data and some mathematical model. Expectation-maximization [12] - an extension of K-means partitioning algorithm, conceptual clustering [26,32,70], and artificial neural network approaches falls under this category. Important studies on artificial neural network [42] include self organizing map [58,59,60] and competitive learning rule [85].

(iv) Clustering by Genetic Algorithms: Initially *GAs* were used to deal with single objective optimization problems such as minimizing or maximizing a variable of a function. Such *GAs* is simple *GA* (*SGA*). David Schaffer proposed vector evaluated genetic algorithm [87,88], which was the first inspiration towards *MOGA*. Several improvements on the vector evaluated genetic algorithm are proposed in [7,33,84,93]. Vector evaluated genetic algorithm dealt with multivariate single objective optimization. V. Pareto suggested Pareto approach [79] to cope with multi objective optimization problems. In this work chromosomes lying on the Pareto optimal front (dominant chromosomes) are selected by *MOGA*, which are giving near optimal solutions for clustering. According to the dominance relation between two candidate chromosomes, a candidate clustering chromosome Ch_1 dominates another candidate clustering chromosome Ch_2 if and only if: 1) Ch_1 is strictly better than Ch_2 in at least one of all the measures considered in the fitness function and 2) Ch_1 is not worse than Ch_2 in any of the measures considered in the fitness function. Researchers developed different implementation of *MOGA* e. g. *NSGA* [91], *NPGA* [44], *SPGA* [98], *PAES* [57], $NSGA-II$ [11], *CEMOGA* [3]. In this paper, we have developed our own *MOGA* to find out near optimal clusters by optimizing H and S values. From the viewpoint of the chromosome encoding scheme *MOGA* can be categorized in binary coded *MOGA* and real coded *MOGA*. In order to keep the mapping between the actual cluster modes and the encoded modes effective, real coded *MOGA* has been implemented in the work.

Traditional clustering algorithms often fail to detect meaningful clusters because most real-world data sets are characterized by a high-dimensional, inherently sparse data space [38]. To overcome this difficulty, researchers consider attribute (or feature) transformation and feature selection techniques [54,64,86,92]. By these approaches either computational complexity increases or loss of information becomes a serious issue that leads to misclassification of data. A trade off approach in clustering analysis has been explored for feature selection that effectively reduces the dimensionality of the data. However, most popular K-modes clustering algorithm [47,48] applied on original data sets may converge to values that are not optimal. To find a globally optimal clustering solution, *GA* has been used for clustering. Most of the clustering approaches, single objective or multi-objective, are converted to a single objective by weighed sum method [52] or by any other means. However, only a few *MOGA* clustering approaches have been proposed so far and their experimental results have demonstrated that *MOGA* clustering approaches significantly outperform existing single objective *GA* clustering approaches [37,38]. Earlier work on clustering by the single objective genetic algorithm are [14,28,35,67,69,78,89] and by *MOGA* are [4,13,18,19,20,21,22,31,40,61,63,68,75,76,81,82,83]. We have implemented three clustering algorithms - K-modes, $MOGA(H,S)$ and $MOGA\_Feature\_Selection(H,S)$. *DB* Index [8] has been used to compare performance of clustering algorithms. It shows the superiority of global optimization capability of *GA* over local search algorithm like K-modes.

Nuovo et. al. [77], integrates $NSGA-II$ [11] with *FCM* [5] to simultaneously reduce the dimensionality and find the best partitioning. However, this method does not use $NSGA-II$ in the clustering step directly (where *FCM* is used in its traditional form). $NSGA-II$ is used on the upper level to determine the features to be selected as well as the parameters of *FCM*. Moreover, this method is only applicable for continuous numeric data sets, not for categorical data as they have used *FCM*. In [4] multi objective fuzzy clustering using $NSGA-II$ [11] has been done, but it is suitable for continuous data and not doing simultaneous attribute selection. Multi objective fuzzy clustering on categorical data has been done in [13,31,68,75,76], but none of them have made a simultaneous feature selection. In [54,55,74] researchers proposed multi-objective optimization of feature sets and clustering schemes. They incorporate the number of features as part of the multi-objective criteria. In [54,55] researchers used four criteria of optimization and in [74] researchers used two criteria of optimization. Moreover, it has already been shown in [71] that those approaches above have a serious flaw by using the wrong optimization direction (information preservation). This paper theoretically proves why the multi-objective approaches proposed so far will fail in certain cases and give a solution by introducing the concept of information preservation into multi-objective feature selection for clustering. In a follow-up paper, the authors even extended this to feature construction instead of feature selection only [72]. In [54,55,74] researches had minimize number of features for easier interpretably and scalability of the solutions as well as better generalization. In [72] instead of minimize number of features researchers have maximize that. It helps to find out more descriptive pattern, prevents the algorithm from selecting trivial solutions and leads to more complete Pareto sets of diversive natural clusterings. So from the above discussion it is very clear that there are a lot of reasons for minimizing the number of features and as well as maximizing the number of features. So in our approach we are not considering number of features as optimization criteria. Rather it will be evolved by *MOGA*.

In the proposed approach, $MOGA\_Feature\_Selection(H,S)$ is selecting optimal features and cluster modes. Chromosomes lengths are not varying because of don't care conditions (#) in chromosomes. Thus, we have avoided variable length chromosomes and complexities associated with it. Experimental results prove the superiority of $MOGA\_Feature\_Selection(H,S)$ over $MOGA(H,S)$.

3 Definitions

A relation R and a list of features $A_1, A_2, ...A_n$ defines a relational schema $R(A_1, A_2, ...A_n)$, where n = total number of features including class label.

The domain of A_i is $dom(A_i)$, where $1 \leq i \leq n$.

X represents a data set, which is a set of tuples that is $X = \{t_1, t_2, ...t_m\}$, where m = total number of tuples or records.

Each tuple t is an n-dimensional attribute vector, which is an ordered list of n values i.e. $t = [v_1^t, v_2^t, ...v_n^t]$, where $v_i^t \in dom(A_i)$, with $1 \leq i \leq n$. v_i^t is i^{th} value in tuple t, which corresponds to the feature A_i. Each tuple, t belongs to predefined class represented by v_n^t where $v_n^t \in dom(A_n)$. For clustering v_n^t or class labels are not known, so $t = [v_1^t, v_2^t, ...v_{n-1}^t]$. Henceforth $n-1$ = total number of features excluding class label.

Formally, the problem is stated as every t_i of X ($1 \leq i \leq m$) is to be clustered into K number of non-overlapping groups $\{C_1, C_2, ...C_K\}$; where $C_1 \cup C_2 \cup ...C_K = X$, $C_i \neq \emptyset$, and $C_i \cap C_j = \emptyset$ for $i \neq j$ and $j \leq K$.

A solution is a set of Cluster Mode (CM) that is $\{CM_1, CM_2, ...CM_K\}$. $(n-1)$-dimensional feature vector, that is $[c_1^i, c_2^i, ...c_{n-1}^i]$ represents CM_i.

Calculating Mod distance between two points is most common way of measuring dissimilarity between two points for categorical data. We have modified formula of Mod distance calculation by dividing it with number of features $(n-1)$ to calculate Mod distance per feature. Equations (1), (2) and (3) calculate Mod distance per feature between one cluster center and one tuple, two cluster centers and two tuples respectively.

$$d(CM_i,t_j) = [[\sum_{l=1}^{(n-1)} MOD(c_l^i,v_l^j)]/(n-1)]^{1/2} \quad (1)$$

$$d(CM_i,CM_j) = [[\sum_{l=1}^{(n-1)} MOD(c_l^i,c_l^j)]/(n-1)]^{1/2} \quad (2)$$

$$d(t_i,i_j) = [[\sum_{l=1}^{(n-1)} MOD(v_l^i,v_l^j)]/(n-1)]^{1/2} \quad (3)$$

$MOD(c_l^i,v_l^j)$, $MOD(c_l^i,c_l^j)$ and $MOD(v_l^i,v_l^j)$ are equal to 0 if $c_l^i = v_l^j$, $c_l^i = c_l^j$ and $v_l^i = v_l^j$. Otherwise they are equal to 1.

4 Proposed Approaches

We have discussed $MOGA(H,S)$ and $MOGA\_Feature\_Selection(H,S)$ in details in this section.

Original data sets are changed to label the last feature as class label. After this step class label becomes n^{th} feature. In clustering class labels are unknown so we are considering first $(n-1)$ features of data sets.

4.1 $MOGA(H,S)$

Flowchart of $MOGA(H,S)$ is provided in figure 1 and described below.

Building Initial Population. In most of the literature on *GA*, fixed number of prospective solutions build initial population (*IP*). Here *IP* size is the nearest integer value of 10% of total number of tuples *m* in the data set. Although correlation between *IP* size and the number of instances in the data set are not known, in our opinion *IP* size guides searching power of *GA* and therefore its size should increase with the size of the data set. Here *IPS* means *IP* size and Ch_j represents the j^{th} chromosome in the population, where $1 \leq j \leq IPS$. Algorithm 1 shows the steps of building initial population. Each chromosome represent a solution, which is a set of Cluster Mode (*CM*) of *K* clusters.

As tuples of data set build cluster modes, it induces faster convergence of *MOGA*, compared to building chromosomes by randomly choosing categorical feature values from the same feature domain.

Reassign Cluster Modes. In every chromosome, a set of cluster modes (*CMs*) representing cluster modes i.e. $\{CM_1,CM_2,...CM_K\}$ are randomly initialized. Every chromosome is the input of *K*-modes algorithm 7 (discussed later). One iteration of *K*-modes algorithm produces new cluster modes denoted by $\{CM_1^*,CM_2^*,...CM_K^*\}$, which forms new chromosomes to replace chromosomes used as input of *K*-modes algorithm.

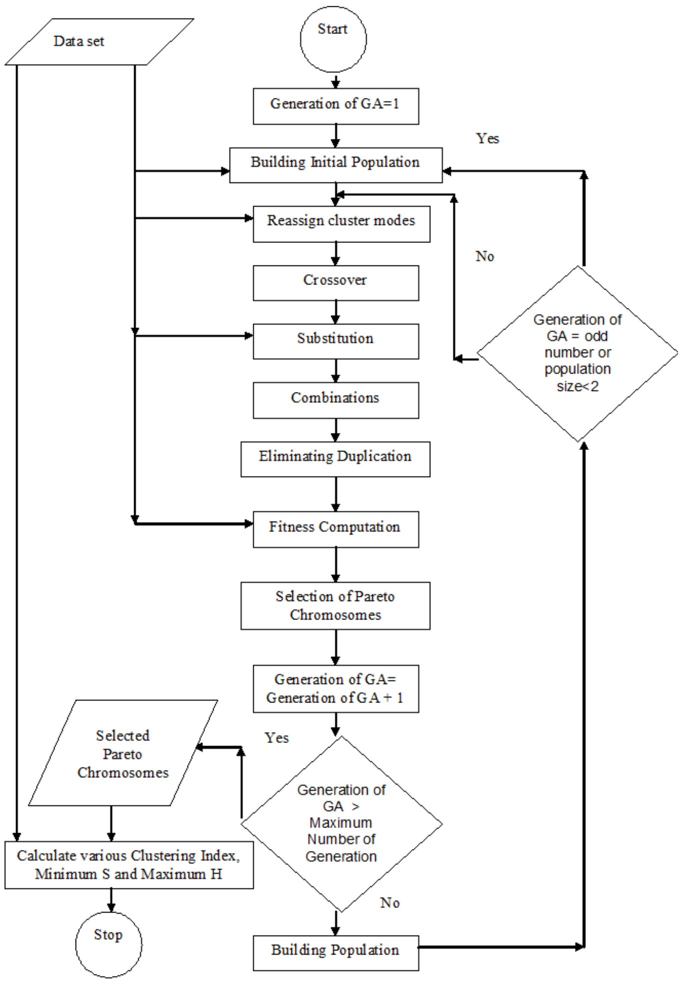

Data set
Start
Generation of GA=1
Building Initial Population
Yes
Reassign cluster modes
No
Crossover
Substitution
Generation of GA = odd number or population size<2
Combinations
Eliminating Duplication
Fitness Computation
Selection of Pareto Chromosomes
Generation of GA= Generation of GA + 1
Yes
Selected Pareto Chromosomes
Generation of GA > Maximum Number of Generation
Calculate various Clustering Index, Minimum S and Maximum H
No
Stop
Building Population

Fig. 1. Flowchart of $MOGA(H,S)$

Algorithm 1. Algorithm for building initial population for $MOGA(H,S)$

```
$m \leftarrow$ total number of tuples in the data set
$K \leftarrow$ number of clusters
$IPS \leftarrow \lfloor(1/10 \times m)\rfloor$
for $j = 1$ to $IPS$ do
    for $i = 1$ to $K$ do
        $rn_i \leftarrow GenerateRandomNumber(1,m)$/* Generate integer random number between 1 to $m$ */
    end for
    $Ch_j \leftarrow v_1^{rn_1}, v_2^{rn_1}, \ldots v_{n-1}^{rn_1} | v_1^{rn_2}, v_2^{rn_2}, \ldots v_{n-1}^{rn_2} | \ldots$
    $| v_1^{rn_K}, v_2^{rn_K}, \ldots v_{n-1}^{rn_K} |$ /* $v_l^{rn_i}$ represents value of $l^{th}$ feature of $rn_i^{th}$ tuple. ',' separates dimensions of $CM_i$ and '|' separates $CM_i$ */
end for
```

Crossover. Chromosomes generated by K-modes algorithm are input to the crossover. Context insensitivity is an important issue in a grouping task like clustering. Meaning of context insensitivity is "the schemata defined on the genes of the simple chromosome do not convey useful information that could be exploited by the implicit sampling process carried out by a clustering *GA*" [25]. In their survey paper Hruschka et. al. [46] shows drawback of conventional single point crossover operators often described in the literature considering context sensitivity. We have used a special crossover operator called "Pairwise crossover" described by Fr*ä*nti in [28] as "The clusters between the two solutions can be paired by searching the "nearest" cluster (in the solution B) for every cluster in the solution A. Crossover is then performed by taking one cluster centroid (by random choice) from each pair of clusters. In this way we try to avoid selecting similar cluster centroids from both parent solutions. The pairing is done in a greedy manner by taking for each cluster in A the nearest available cluster in B. A cluster that has been paired cannot be chosen again, thus the last cluster in A is paired with the only one left in B." This algorithm does not give the optimal pairing (2-assignment) but it is a reasonably good heuristic for the crossover purpose. In [21] we have used 0.9 as Crossover probability (P_{co}). Here P_{co}) is varying from 0.5 to 0.9. i. e. 50% to 90% chromosomes are engage in crossover. At the beginning of *GA*, P_{co} is 0.5 because chances of getting good chromosomes are less at the early stage of *GA*. With generation of *GA*, chances of getting good chromosomes are increasing. So P_{co} is increasing linearly over different generation and at the end it becomes 0.9.

Substitution. In [21] substitution probability (P_{sb}) of *MOGA* was 0.1. At the beginning of *GA* substitution probability (P_{sb}) is 0.1, then decreases linearly with different run of *GA* and at end it is 0.0. At early stage of *GA*, chances of getting good chromosomes are less so P_{sb} is high to introduce new information into chromosomes which may shift chromosomes from local optima zone to global optima zone. Over generations of *GA* chances of getting good chromosomes are increasing. Introducing random information to good chromosomes may destroy good building blocks of chromosomes. So P_{sb} is decreasing over generations of *GA*. Produced child chromosomes by crossover are parent chromosomes of this step. Here $dom(A_i)$ is categorical. In conventional mutation, any

Algorithm 2. Algorithm for crossover of $MOGA(H,S)$

```
/* All $Ch$ produced by $K$-modes described in 4.1 are $PChs$ of crossover */
$P_{co} \leftarrow$ (0.5+(0.9-0.5)*Generation/100); $N_{pch} \leftarrow$ Number of $PChs$ in population; $N_{co} \leftarrow \lfloor N_{pch} \times P_{co}) \rfloor$; $K \leftarrow$ number of clusters; $n \leftarrow$ number of features;
for $counter = 1$ to $N_{pch}$ do
    $PChFlag_{counter} \leftarrow True$/* To prevent repeated choosing of same $PCh$ */
end for
$counter \leftarrow 1$
while $counter < N_{co}/2$ do
    $rn1 \leftarrow GenerateRandomNumber(1,N_{pch})$; $rn2 \leftarrow GenerateRandomNumber(1,N_{pch})$/* Generate integer random numbers between 1 to $N_{pch}$ */
    if $rn1 \neq rn2$ AND $PChFlag_{rn1} = True$ AND $PChFlag_{rn2} = True$ then
        $counter \leftarrow counter + 1$; $PChFlag_{rn1} \leftarrow False$; $PChFlag_{rn2} \leftarrow False$/* To prevent repeated choosing of same $PCh$ */
        /* ******************************Building distance matrix (M)****************************** */
        for $i = 1$ to $K$ do
            for $j = 1$ to $K$ do
                $CM_i \leftarrow CM_i$ of $PCh_{rn1}$; $CM_j \leftarrow CM_j$ of $PCh_{rn2}$; $a_{ij} \leftarrow d(CM_i,CM_j)$ /* $a_{ij}$ is element of $i^{th}$ row and $j^{th}$ column of distance matrix (M) and equation 2 is calculating $d(CM_i,CM_j)$ */
            end for
        end for
        /* ******************************Constructing $CCh$ from $PCh$****************************** */
        for $counter1 = 1$ to $K$ do
            $flag1_{counter1} \leftarrow True$; $flag2_{counter1} \leftarrow True$
        end for
        for $counter1 = 1$ to $K$ do
            $minDistance \leftarrow 1000000$; $i1 \leftarrow -1$; $j1 \leftarrow -1$
            for $i = 1$ to $K$ do
                if $flag1_i = True$ then
                    for $j = 1$ to $K$ do
                        if $flag2_j = True$ then
                            if $a_{ij} < minDistance$ then
                                $minDistance \leftarrow a_{ij}$
                                $i1 \leftarrow i$
                                $j1 \leftarrow j$
                            end if
                        end if
                    end for
                end if
            end for
            $rn3 \leftarrow GenerateRandomNumber(0,1)$/* Generate random number between 0 to 1 */
            if $rn3 < 0.5$ then
                $CM_{i1}$ of $PCh_{rn1} \in CCh_{rn1}$
                $CM_{j1}$ of $PCh_{rn2} \in CCh_{rn2}$
            else
                $CM_{i1}$ of $PCh_{rn1} \in CCh_{rn2}$
                $CM_{j1}$ of $PCh_{rn2} \in CCh_{rn1}$
            end if
            $flag1_i \leftarrow False$; $flag2_j \leftarrow False$/* To prevent repeated choosing of same $CM$ of $PCh_i$ or $PCh_j$ */
        end for
    end if
end while
/* ***********************For $PChs$ not taking part in crossover operation****************************** */
for $counter = 1$ to $N_{pch}$ do
    if $PChFlag_{counter} = True$ then
        $CCh_{counter} \leftarrow PCh_{counter}$
    end if
end for
```

Algorithm 3. Algorithm for substitution of $MOGA(H,S)$

```
$K \leftarrow$ number of clusters
$N_{pch} \leftarrow$ Number of $PChs$ in population
$P_{sb} \leftarrow (0.1-(0.1-0.0)*Generation/100)/$
for $i=1$ to $N_{pch}$ do
  for $j=1$ to $K$ do
    $rn1 \leftarrow GenerateRandomNumber(0,1)$/* Generate random number between 0 to 1 */
    if $rn1 < P_{sb}$ then
      $rn2 \leftarrow GenerateRandomNumber(1,m)$/* Generate integer random number between 1 to $m$ */
      Substitute $CM_j$ of $PCh_i$ with $t_{rn2}$
    end if
  end for
end for
```

random value from $dom(A_i)$ can replace v_i resulting many chromosomes. Considering context sensitivity, instead of replacing any v_i of chromosome, substitution is replacing cluster modes by any tuples randomly. So unlike conventional mutation, substitution is not producing many chromosomes. Approximately number of substitution (N_{sb}) = $\lfloor(N_{pch} \times K \times P_{sb})\rfloor$. Algorithm 3 explains substitution.

Combination. At the end of every generation of $MOGA$ some chromosomes are lying on the Pareto optimal front. These chromosomes have survived and these are known as Pareto chromosomes. Chromosomes obtained from previous substitution method (algorithm 3 of section 4.1), say NC_{pre} and previous generation Pareto chromosomes (section 4.1) of $MOGA$, say PC_{pre} are considered to perform in the next generation. For the first generation of GA, PC_{pre} is zero because of the nonexistence of previous generation.

In general, for i^{th} generation of $MOGA$, if NC_{pre} is m and PC_{pre} at the end of $(i-1)^{th}$ generation of $MOGA$ is n then after combination, number of chromosomes are $n+m$ or $PC_{pre}+NC_{pre}$.

Eliminating Duplication. Survived chromosomes from previous generation may be generated again in the present generation of $MOGA$, resulting multiple copies of the same chromosome. So in this step system is selecting unique chromosomes from the combined chromosome set. One may argue with the requirement of this step. But if we remove this step then population size will be very high at the end of some iterations of $MOGA$. Algorithm 4 is describing the process.

Fitness Computation. As stated earlier, intra-cluster distance (Homogeneity) (H) and inter-cluster distances (Separation) (S) are two measures for optimization. Maximization of $1/H$ and S are the twin objectives of $MOGA$. It converts the optimization problem befitting into max-max framework. System is computing the values of H and S by using Mod distance per feature measures.

Algorithm 4. Algorithm for removing duplication of $MOGA(H,S)$

```
$n \leftarrow PC_{pre}$
$m \leftarrow NC_{pre}$
$NOC \leftarrow (n+m)$
**for** $j = 1$ to $NOC$ **do**
    $Flag_j \leftarrow True$
**end for**
$i \leftarrow 1$
**for** $j = 1$ to $NOC$ **do**
    **if** $Flag_j = True$ **then**
        $UniqueChromosome_i = Chromosome_j$
        $i \leftarrow (i+1)$
        **for** $l = 1$ to $NOC$ **do**
            **if** $Chromosome_l = Chromosome_j$ **then**
                $Flag_l \leftarrow False$
            **end if**
        **end for**
    **end if**
**end for**
```

Equation (1) is calculating Mod distance per feature between one cluster center and one tuple.

If $d_{lowest}(CM_i,t_j)$ is the lowest distance for any value of i (where $1 \leq i \leq K$), then t_j is assigned to C_i. Equation (4) defines H_i (average intra-cluster distance or homogeneity of i^{th} cluster).

$$H_i = [\sum_{j=1}^{m} d_{lowest}(CM_i,t_j)]/m_i \tag{4}$$

where m_i is the number of tuples belonging to i^{th} cluster and t_j is assigned to i^{th} cluster based on lowest distance.

Summation of H_i is H, defined in equation 5 as

$$H = \sum_{i=1}^{K} H_i \tag{5}$$

Say, t_j and t_p are two distinct tuples from a data set where t_j is assigned to C_x and t_p is assigned to C_y. S is defined in equation 6 using $d_{lowest}(CM_i,t_j)$ as

$$S = \sum_{j,p=1}^{m} d(t_j,t_p) \tag{6}$$

where $j \neq p$ and $x \neq y$.

For every chromosome in the population, algorithm 5 calculates $1/H$ and S.

Selection of Pareto Chromosomes. All chromosomes lying on the front of $max(1/H,S)$ are selected as Pareto chromosomes. In other words, maximizing $1/H$

Algorithm 5. Algorithm for calculating of H and S of $MOGA(H,S)$

```
$m \leftarrow$ total number of tuples in the data set
$K \leftarrow$ number of clusters
$n \leftarrow$ number of features
/* *********Assign tuples into clusters********* */
for $j = 1$ to $m$ do
    $AssingedClusterNumber_j \leftarrow -1$
end for
for $j = 1$ to $m$ do
    $distance \leftarrow 1000000000$
    $clusterNo \leftarrow -1$
    for $i = 1$ to $K$ do
        /* Equation (1) calculates $d(CM_i,t_j)$ */
        if $distance > d(CM_i,t_j)$ then
            $distance \leftarrow d(CM_i,t_j)$
            $clusterNo \leftarrow i$
        end if
        if $distance = d(CM_i,t_j)$ then
            $rn \leftarrow GenerateRandomNumber(0,1)$/* Generate random number between 0 to 1 */
            if $rn > 0.5$ then
                $distance \leftarrow d(CM_i,t_j)$
                $clusterNo \leftarrow i$
            end if
        end if
    end for
    $AssingedClusterNumber_j \leftarrow clusterNo$
end for
/* ***************Calculating $H$*************** */
$H \leftarrow 0$
for $i = 1$ to $K$ do
    $H_i \leftarrow 0$
    $m_i \leftarrow 0$
end for
for $j = 1$ to $m$ do
    for $i = 1$ to $K$ do
        if $i = AssingedClusterNumber_j$ then
            $H_i \leftarrow H_i + d(CM_i,t_j)$
            $m_i \leftarrow m_i + 1$
        end if
    end for
end for
for $i = 1$ to $K$ do
    $H \leftarrow H + H_i/m_i$
end for
/* ***************Calculating $S$*************** */
$S \leftarrow 0$
for $j = 1$ to $m$ do
    for $i = j + 1$ to $m$ do
        if $AssingedClusterNumber_i \neq AssingedClusterNumber_j$ then
            $S \leftarrow S + d(t_i,t_j)$/* Equation (3) calculates $d(t_i,t_j)$ */
        end if
    end for
end for
```

and S are two objectives of $MOGA$. As this process is applied after combination, elitism [10,11] is adhered to.

Building Intermediate Population. From 2^{nd} generation onwards for every even generation, selected Pareto chromosomes of previous generation build population. This helps to build a population of chromosomes close to the Pareto optimal front resulting fast convergence of $MOGA$. For every odd generation of $MOGA$ or if for any generation previous generation of $MOGA$ does not produce Pareto chromosomes greater than 2 then population are generated using algorithm 1. This introduces new chromosomes in population and induces diversity in the population. Population size also varies from generation to generation.

From the above discussion one may get wrong idea that we are destroying any search we are doing during a generation and there is no solution evolution. In other words, there is really no time for evolution. But notice that, we are preserving Pareto chromosomes at every generation (i^{th} generation of GA) and that is combined to present generation chromosomes ($(i+1)^{th}$ generation of GA) during Combination (section 4.1). Selection of Pareto chromosomes are done on combined population set (section 4.1). As pointed out that at every odd generation we generate the population using the same procedure we generate the initial population (section 4.1). But is these cases we are not loosing Pareto chromosomes selected by previous even generation because we stores them in a separate place.

4.2 MOGA_Feature_Selection (H, S)

In $MOGA(H,S)$, all features except class labels are taking part to build chromosomes. But all features in the data sets are not equally important in clustering. So $MOGA\_Feature\_Selection(H,S)$ select only important/ relevant features for clustering. This method provides three-way benefits.

i) Clustering performance improves significantly.

ii) Important features are easily identified.

iii) Due to the reduction of dimensionality in $MOGA\_Feature\_Selection(H,S)$, faster convergence than $MOGA(H,S)$ is achieved.

As some features are forming chromosomes, equation 1, 2 and 3 can be modified as 7, 8 and 9 respectively.

$$d_{mod}(CM_i,t_j) = [[\sum_{l=1}^{(n-1)} MOD(c_l^i, v_l^j)]/nva]^{1/2} \tag{7}$$

$$d_{mod}(CM_i,CM_j) = [[\sum_{l=1}^{(n-1)} MOD(c_l^i, c_l^j)]/nva]^{1/2} \tag{8}$$

$$d_{mod}(t_i,i_j) = [[\sum_{l=1}^{(n-1)} MOD(v_l^i, v_l^j)]/nva]^{1/2} \tag{9}$$

where nva = number of valid features in corresponding chromosome that is where $c_l^i \neq \#$. If $c_l^i = \#$ then $MOD(c_l^i, v_l^j) = 0$ or $MOD(c_l^i, c_l^j) = 0$ or $MOD(v_l^i, v_l^j) = 0$.

Algorithm 6. Algorithm for building initial population for $MOGA\_Feature\_Selection(H,S)$

```
$m \leftarrow$ total number of tuples in the data set
$K \leftarrow$ number of clusters
$n \leftarrow$ number of features
$IPS \leftarrow \lfloor (1/10 \times m) \rfloor$
$rn \leftarrow GenerateRandomNumber(1,(n-1))$/* Generate integer random number between 1 to $(n-1)$ */
valid features $(va) \leftarrow ChooseAttributes(rn,n-1)$ /* Choose $rn$ number of features from $(n-1)$ features */
invalid features $(ia) \leftarrow NotChosenAttributes(va,n-1)$ /* Choose other $(n-1-rn)$ number of attributes from $(n-1)$ features */
for $j = 1$ to $IPS$ do
    for $i = 1$ to $K$ do
        $rn_i \leftarrow GenerateRandomNumber(1,m)$/* Generate integer random number between 1 to $m$ */
    end for
    $Ch_j \leftarrow v_1^{rn_1}/\#, v_2^{rn_1}/\#, \ldots v_{n-1}^{rn_1}/\#|$
        $v_1^{rn_2}/\#, v_2^{rn_2}/\#, \ldots v_{n-1}^{rn_2}/\#|$
        $\ldots|v_1^{rn_K}/\#, v_2^{rn_K}/\#, \ldots v_{n-1}^{rn_K}/\#|$
    /* $v_l^{rn_i}$ represents value of $l^{th}$ feature of $rn_i^{th}$ tuple. ',' separates dimensions of any $CM_i$ and '$|$' separates $CM_i$.# represents don't care condition. For $va$ values of tuples are building chromosomes and for $ia$ # is building chromosome. */
end for
```

$MOD(c_l^i, v_l^j)$, $MOD(c_l^i, c_l^j)$ and $MOD(v_l^i, v_l^j)$ are equal to 0 if $c_l^i = v_l^j$, $c_l^i = c_l^j$ and $v_l^i = v_l^j$ respectively. Otherwise they are equal to 1.

Note that when all attributes are taking part in chromosome building, equations 7, 8 and 9 will become same as equations 1, 2 and 3 respectively.

Different steps of $MOGA\_Feature\_Selection(H,S)$ are described below. It is using equation 7, 8 and 9.

Building Initial Population. Initial population size is the nearest integer value of 10% of data set size. Random number of features build chromosomes. Algorithm 6 describes the process of building initial population.

Reassign Cluster Modes. Every chromosome delivers a set of cluster modes (CMs) that is $\{CM_1, CM_2, \ldots CM_K\}$. However, in $MOGA\_Feature\_Selection(H,S)$, all features are not building cluster modes. For valid features only K-modes algorithm forms new cluster modes (CMs) that is $\{CM_1^*, CM_2^*, \ldots CM_K^*\}$.

Crossover. Crossover probability (P_{co}) for $MOGA\_Feature\_Selection(H,S)$ is also varying from 0.5 to 0.9. "Pairwise crossover" is used in this case too. Parent chromosomes must have the same set of valid and invalid features to calculate distance between cluster modes. So we are using "repeated crossover". At first, one chromosome is selected randomly from the population to consider a set of valid and invalid features from population generated by reassign cluster modes method (section 4.2). As a second step, population for crossover is built by selecting chromosomes having the same set of valid and invalid features as the randomly selected chromosome. In this way a fraction of population having same set of valid and invalid features underdone for Crossover. Crossover is done and it changes some chromosomes. Child chromosomes replace parent chromosomes in the original population formed by reassign cluster modes method (section 4.2). Other chromosomes which are not participating in crossover are copied as it is. This process is repeated unless all chromosomes are selected for crossover. This process is particularly effective for high dimensional dataset. "Repeated crossover" is not required for $MOGA(H,S)$ as all features having valid values.

Substitution. Substitution probability (P_{sb}) is varying from 0.1 to 0.0 in this case too. Chromosomes formed by crossover become parent chromosomes to this. Substitution is same as $MOGA(H,S)$ (algorithm 3 of section 4.1) except in one place. Only valid features of cluster modes are substituted by feature values of any randomly chosen tuple.

Fitness Computation. Modified intra-cluster distance (Homogeneity) (H_{mod}) and modified inter-cluster distances (Separation) (S_{mod}) are two measures for calculating fitness values of $MOGA\_Feature\_Selection(H,S)$ clustering algorithm. Maximization of $1/H_{mod}$ and S_{mod} are the objectives of $MOGA$. It converts optimization problem into max-max framework.

Equations (4), (5) and (6) are modified into equations (10), (11) and (12) respectively.

$$H_{mod_i} = [\sum_{j=1}^{m} d_{mod_{lowest}}(CM_i, t_j)]/m_i \tag{10}$$

$$H_{mod} = \sum_{i=1}^{K} H_{mod_i} \tag{11}$$

$$S_{mod} = \sum_{j,p=1}^{m} d_{mod}(t_j, t_p) \tag{12}$$

As we are using Mod distance per feature, $1/H_{mod}$ and S_{mod} $1/H_{mod}$ and S_{mod} values of different chromosomes having different sets of valid features become comparable with one another.

Note that when all features are taking part in chromosome building equations 10, 11 and 12 become same as equations 4, 5 and 6 respectively.

Algorithm 5 calculates $1/H_{mod}$ and S_{mod} for every chromosome in the population. In this case equations 7, 8 and 9 have been used in place of equations 1, 2 and 3 respectively.

Combination. This is same as used in $MOGA(H,S)$ as described in section 4.1.

Eliminating Duplication. This is same as that used in $MOGA(H,S)$ as described in section 4.1.

Selection of Pareto Chromosomes. All chromosomes lying on the front of max $(1/H_{mod}, S_{mod})$ are Pareto chromosomes. In other words maximizing $1/H_{mod}$ and S_{mod} are two objectives of *MOGA*. As this process is applied after combination, elitism is adhered to.

Building Intermediate Population. From 2^{nd} generation onwards for every even generation, selected Pareto chromosomes of previous generation build population. This helps to build a population of chromosomes close to the Pareto optimal front, resulting convergence of *MOGA*. For every odd generation of *MOGA* or if for any generation previous generation of *MOGA* does not produce Pareto chromosomes greater than 2 then population are created using algorithm 6 of section 4.2. This introduces new chromosomes in population and induces diversity in the population. Population size also varies from one generation to another.

5 Testing and Comparison

For comparison purposes we have also implemented *K*-modes.

K-modes is most popular partition based traditional clustering method [47,48] for categorical attributes. Algorithm 7 describes steps of the *K*-modes.

Algorithm 7. Algorithm for *K*-modes

```
1: Choose $K$ initial cluster modes $\{CM_1, CM_2, ...CM_K\}$ randomly from $m$ tuples $\{t_1, t_2, ...t_m\}$ of
   $X$ where $t = [v_1^t, v_2^t, ..., v_{n-1}^t]$.
2: Assign tuple $t_j$, $j = 1,2,...m$ to cluster $C_i$, $i \in \{1,2,...K\}$ iff $d(CM_i, t_j) < d(CM_p, t_j)$, $p =$
   $1,2,...K$, and $i \neq p$. /* Using equation 1 */
   Resolve ties arbitrarily.
3: Compute new cluster modes $\{CM_1^*, CM_2^*, ...CM_K^*\}$ as follows:
   $CM_i^*$ =Mode of $t_j$
   where $t_j \in C_i$ and $j = 1,2,...m$.
4: If $CM_i^* = CM_i$, $i = 1,2,...K$ then stop. Otherwise repeat from step 2.
```

Note that in case the process does not stop at step 4 normally, then it is carried out for a maximum fixed number of iterations. We have taken a maximum fixed number of iterations as 100 which is same as the number of generations of *GA*. It is observed that in most of the cases it stops much before maximum fixed number of iterations. One may argue that this comparison is not fair, because *GA* works on minimum number of probable solutions and *K*-modes works on a single probable solution. But we don't have any option.

In literature different validity indices are available to measure the goodness of clusters. We have used three popular validity indices - Davies-Bouldin (*DB*) Index [8], *C*-Index [50] and Dunn Index [17]. Equation (13) is calculating *DB* Index.

$$DB\,Index = 1/K \sum_{i=1, i \neq j}^{K} max((H_i + H_j)/d(CC_i, CC_j)) \tag{13}$$

where K is the number of clusters, H_i is the average distance of all patterns in cluster i to their cluster mode CM_i, H_j is the average distance of all patterns in cluster j to their cluster mode CM_j and $d(CM_i, CM_j)$ is the distance of cluster modes CM_i and CM_j.

The Dunn index defines the ratio between the minimal intra-cluster distance to maximal inter-cluster distance. Equation (14) calculates the index.

$$Dunn\ Index = d_{min}/d_{max} \tag{14}$$

where d_{min} denote the smallest distance between two objects from different clusters, and d_{max} the largest distance of two objects from the same cluster. The Dunn index should be maximized.

The C-index is defined by equation (15)

$$C\ Index = (S - S_{min})/(S_{max} - S_{min}) \tag{15}$$

where S is the sum of distances over all pairs of objects form the same cluster, n is the number of those pairs and S_{min} is the sum of the n smallest distances if all pairs of objects are considered. Likewise S_{max} is the sum of the n largest distances out of all pairs. The C-index is limited to the interval [0, 1] and should be minimized.

For $K - modes$ and $MOGA(H,S)$ equations (13), (14) and (15) are using equations (1), (2) and (3) whereas for $MOGA\_Feature\_Selection(H,S)$ equations (13), (14) and (15) are using equations (7), (8) and (9).

As number of clusters (K) are equal to the number of classes in the datasets we can also use classification accuracy as a measure to estimate the goodness of cluster [18]. While testing, tuples of test set are provided with class labels. In a chromosome, every cluster is assigned a class label based on majority voting method. Predicted class label and actual class label are used to build a confusion matrix or matching matrix [15] for each chromosome in Pareto population. Each tuple t_j where $1 \leq j \leq m$ and m = total number of tuples in test set are assigned to a cluster based on $d_{lowest}(CM_i, t_j)$ where $1 \leq i \leq K$ and K = number of cluster to get predicted class label. From confusion matrix, performance of classifying all t_j is measured.

2 popular data sets from UCI Machine Learning Repository [41] compares the performance of 3 clustering algorithms. Table 3 summarizes features of data sets.

Table 2. Features of data sets used in this article (Here, # indicates Number of)

Name	#Tuples	#Attributes	#Classes
Soybean(Small)	47	35	4
Zoo	101	18	7

Table 4 compares H and S values of 3 algorithms. Tables 5, 6, 7 and 8 compares DB Index, C Index, Dunn Index and Classification Accuracy of 3 algorithms. Column A1 showing Clustering Index/Classification Accuracy values of $K - modes$, Column B1 and C1 showing Clustering Index/Classification Accuracy values for minimum H, Column B2 and C2 showing Clustering Index/Classification Accuracy values for maximum S, Column B3 and C3 showing average Clustering Index/Classification Accuracy values, Column B4 and C4 showing best Clustering Index/Classification Accuracy values.

Table 3. Features of data sets used in this article (Here, # indicates Number of)

Name	#Tuples	#Attributes	#Classes
Soybean(Small)	47	35	4
Zoo	101	18	7

Table 4. Comparison of performance of clustering algorithms in terms of H and S

Methods	$K-modes$		$MOGA(H,S)$		$MOGA\_Feature\_Selection(H,S)$	
Data set	H	S	H	S	H	S
Soybean(Small)	1.37	469.87	1.09	484.01	0.47	567.73
Zoo	1.71	2782.24	1.36	2849.65	0.14	3105.25

Table 5. Comparison of performance of clustering algorithms in terms of DB Index

Methods	$K-modes$	$MOGA(H,S)$				$MOGA\_Feature\_Selection(H,S)$			
Data set	A1	B1	B2	B3	B4	C1	C2	C3	C4
Soybean(Small)	1.38	1.28	1.35	1.31	1.17	0.72	1.34	0.95	0.65
Zoo	1.23	1.11	1.37	1.24	1.06	0.14	0.89	0.45	0.14

Table 6. Comparison of performance of clustering algorithms in terms of C Index

Methods	$K-modes$	$MOGA(H,S)$				$MOGA\_Feature\_Selection(H,S)$			
Data set	A1	B1	B2	B3	B4	C1	C2	C3	C4
Soybean(Small)	0.11	0.07	0.05	0.09	0.02	0.08	0.08	0.08	0.01
Zoo	0.11	0.15	0.09	0.13	0.08	0.02	0.06	0.06	0.00

Table 7. Comparison of performance of clustering algorithms in terms of Dunn Index

Methods	$K-modes$	$MOGA(H,S)$				$MOGA\_Feature\_Selection(H,S)$			
Data set	A1	B1	B2	B3	B4	C1	C2	C3	C4
Soybean(Small)	0.48	0.47	0.52	0.46	0.64	0.73	0.39	0.35	0.80
Zoo	0.24	0.16	0.25	0.21	0.37	Infinity	0.09	Infinity	Infinity

Table 8. Comparison of performance of clustering algorithms in terms of Classification accuracy

Methods	$K-modes$	$MOGA(H,S)$				$MOGA\_Feature\_Selection(H,S)$			
Data set	A1	B1	B2	B3	B4	C1	C2	C3	C4
Soybean(Small)	0.83	0.63	0.95	0.77	0.99	0.61	0.91	0.78	1
Zoo	0.65	0.59	0.66	0.63	0.73	0.54	0.56	0.55	0.76

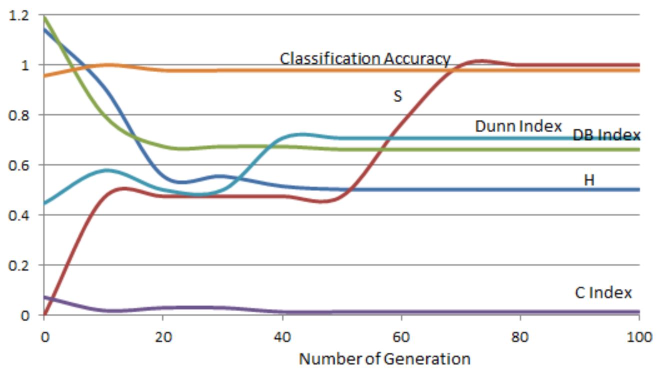


Fig. 2. H, S, *DB* Index, *C* Index, Dunn Index, Classification Accuracy Vs Number of Generation of $MOGA\_Feature\_Selection(H,S)$ of Soybean(Small) dataset

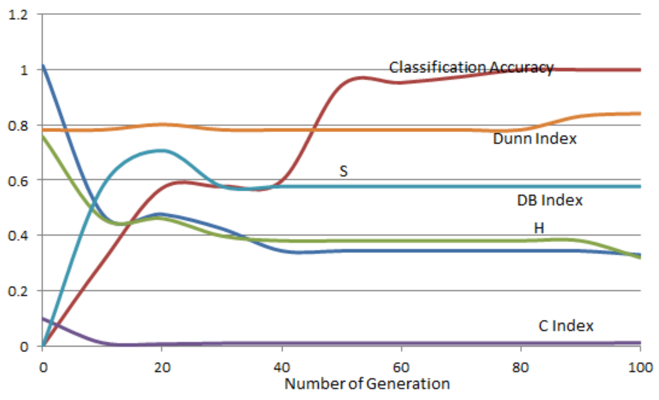


Fig. 3. H, S, *DB* Index, *C* Index, Dunn Index, Classification Accuracy Vs Number of Generation of $MOGA\_Feature\_Selection(H,S)$ of Zoo dataset

Following important observations are summarized here, based on the analysis of table 4, 6, 7 and 8. Result shows the superiority of $MOGA(H,S)$ over $K-modes$ as $K-modes$ is doing local optimization whereas $MOGA(H,S)$ is doing global optimization. $K-modes$ is doing optimization of single objective whereas $MOGA(H,S)$ is optimizing H and S. $MOGA\_Feature\_Selection(H,S)$ is better than $MOGA(H,S)$, which shows the effectiveness of simultaneous feature selection. Figure 2 and 3 shows H, S, DB Index, C Index, Dunn Index, Classification Accuracy Vs Number of Generation of $MOGA\_Feature\_Selection(H,S)$ of Soybean(Small) and Zoo datasets respectively. Where S values are normalized in the range [0,1]. Others are showing original values.

6 Conclusions

In this work, we have implemented two versions of a novel real coded hybrid elitist *MOGA* for K-clustering ($MOGA(H,S)$ and $MOGA\_Feature\_Selection(H,S)$). It is known that elitist model of *GAs* provides the optimal string as the number of iterations increases to infinity [6]. Due to the use of special population building process, hybridization of K-modes with *GA*, special crossover and substitution operator *GAs* are producing good clustering solution in a smaller number of iterations (100). $MOGA\_Feature\_Selection(H,S)$ is giving best results among 3 algorithms. It shows the effectiveness of simultaneous feature selection.

We have achieved one important data mining task of clustering by finding cluster modes. We have considered only categorical features in this work. Continuous features need other type of encoding, as Mod distance per feature measures is not suitable for those types of features. Dealing with missing features values and unseen data are other problem areas. It may be interesting to adapt *MOGA* based K-clustering unknown number of clusters. Authors are working in these directions.

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proceeding of ACM International Conference Management of Data, pp. 94–105 (1998)
2. Ankerst, M., Breunig, M., Kriegel, H.P., Sander, J.: OPTICS: ordering points to identify the clustering structure. In: Proceeding of International Conference Management of Data, pp. 49–60 (1999)
3. Bandyopadhyay, S., Pal, S.K., Aruna, B.: Multi-objective GAs, quantitative indices and pattern classification. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics 34, 2088–2099 (2004)
4. Bandyopadhyay, S., Maulik, U., Mukhopadhyay, A.: Multiobjective genetic clustering for pixel classification in remote sensing imagery. IEEE Transactions on Geoscience Remote Sensing 45, 1506–1511 (2007)
5. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell (1981)
6. Bhandari, D., Murthy, C.A., Pal, S.K.: Genetic algorithm with elitist model and its convergence. International Journal of Pattern Recognition and Artificial Intelligence 10, 731–747 (1996)

7. Cvetkovic, D., Parmee, I.C., Webb, E.: Multi-Objective Optimisation and Preliminary Airframe Design. In: Parmee, I.C. (ed.) Adaptive Computing in Design and Manufacture: The Integration of Evolutionary and Adaptive Computing Technologies with Product/System Design and Realisation, pp. 255–267. Springer, New York (1998)
8. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1, 224–227 (1979)
9. Day, W.H., Edelsbrunner, H.: Efficient algorithms for agglomerative hierarchical clustering methods. Journal of Classification 1, 1–24 (1984)
10. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, New York (2001)
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197 (2002)
12. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society 39, 1–38 (1977)
13. Deng, S., He, Z., Xu, X.: G-ANMI: A mutual information based genetic clustering algorithm for categorical data. Knowledge-Based Systems 23, 144–149 (2010)
14. Dhiraj, K., Rath, S.K.: Comparison of SGA and RGA based clustering algorithm for pattern recognition. International Journal of Recent Trends in Engineering 1, 269–273 (2009)
15. Dogra, S.K.: Confusion Matrix, QSARWorld–A Strand Life Sciences Web Resource, `http://www.qsarworld.com/qsar-ml-confusion-matrix.php`
16. Dunham, M.H.: Data Mining: Introductory and Advanced Topics. Prentice-Hall, Eaglewood Cliffs (2002)
17. Dunn, J.: Well separated clusters and optimal fuzzy partitions. Journal of Cybernetics 4, 95–104 (1974)
18. Dutta, D., Dutta, P., Sil, J.: Clustering by multi objective genetic algorithm. In: Proceeding of 1st IEEE International Conference on Recent Advances in Information Technology, pp. 548–553 (2012)
19. Dutta, D., Dutta, P., Sil, J.: Clustering data set with categorical feature using multi objective genetic algorithm. In: Proceeding of IEEE International Conference on Data Science and Engineering, pp. 103–108 (2012)
20. Dutta, D., Dutta, P., Sil, J.: Data clustering with mixed features by multi objective genetic algorithm. In: Proceeding of 12th IEEE International Conference on Hybrid Intelligent Systems, pp. 336–341 (2012)
21. Dutta, D., Dutta, P., Sil, J.: Simultaneous feature selection and clustering for categorical features using multi objective genetic algorithm. In: Proceeding of 12th IEEE International Conference on Hybrid Intelligent Systems, pp. 191–196 (2012)
22. Dutta, D., Dutta, P., Sil, J.: Simultaneous continuous feature selection and K clustering by multi objective genetic algorithm. In: Proceeding of 3rd IEEE International Advance Computing Conference, pp. 937–942 (2013)
23. Ester, M., Kriegel, H.P., Sander, J.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceeding of 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
24. Faceli, K., de Carvalho, A.C.P.L.F., de Souto, M.C.P.: Multi-objective clustering ensemble. International Journal of Hybrid Intelligent Systems 4, 145–156 (2013)
25. Falkenauer, E.: Genetic Algorithms and Grouping Problems. John Wiley & Sons, New York (1998)
26. Fisher, D.H.: Improving inference through conceptual clustering. In: Proceeding of National Conference on Artificial Intelligence, pp. 461–465 (1987)
27. Forsati, R., Doustdar, H.M., Shamsfard, M., Keikha, A., Meybodi, M.R.: A fuzzy co-clustering approach for hybrid recommender systems. International Journal of Hybrid Intelligent Systems 10, 71–81 (2013)

28. Fränti, P., Kivijärvi, J., Kaukoranta, T., Nevalainen, O.: Genetic Algorithms for Large-Scale Clustering Problems. The Computer Journal 40, 547–554 (1997)
29. Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer, Berlin (2002)
30. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press, New York (1990)
31. Gan, G., Wu, J., Yang, Z.: A genetic fuzzy K-Modes algorithm for clustering categorical data. Expert Systems with Applications 36, 1615–1620 (2009)
32. Gennari, J.H., Langley, P., Fisher, D.: Models of incremental concept formation. Artificial Intelligence 40, 11–61 (1989)
33. Goldberg, D.E.: Genetic Algorithms for Search, Optimization, and Machine Learning, 1st edn. Addison-Wesley Longman, Reading (1989)
34. Guha, S., Rastogi, R., Shim, K.: CURE, an efficient clustering algorithm for large databases. In: Proceedings of ACM International Conference on Management of Data, pp. 73–84 (1998)
35. Hall, L.O., Özyurt, I.B., Bezdek, J.C.: Clustering with a genetically optimized approach. IEEE Transactions on Evolutionary Computation 3, 103–112 (1999)
36. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2005)
37. Handl, J., Knowles, J.D.: Evolutionary multiobjective clustering. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 1081–1091. Springer, Heidelberg (2004)
38. Handl, J., Knowles, J.D.: Exploiting the Trade-off - The Benefits of Multiple Objectives in Data Clustering. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 547–560. Springer, Heidelberg (2005)
39. Handl, J., Knowles, J.: Multi-objective clustering and cluster validation. In: Jin, Y. (ed.) Multi-Objective Clustering and Cluster Validation. SCI, vol. 16, pp. 21–47. Springer, Heidelberg (2006)
40. Handl, J., Knowles, J.: An evolutionary approach to multiobjective clustering. IEEE Transactions on Evolutionary Computation 11, 56–76 (2007)
41. Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
42. Hilton, G., Sejnowski, T.J. (eds.): Unsupervised learning: foundations of neural computation. MIT Press, Cambridge (1999)
43. Hinneburg, A., Hinneburg, E., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. In: Proceeding of 4th International Conference on Knowledge Discovery and Data Mining, pp. 58–65 (1998)
44. Horn, J., Nafploitis, N., Goldberg, D.E.: A niched Pareto genetic algorithm for multiobjective optimization. In: Proceeding of IEEE Conference on Evolutionary Computation, pp. 82–87 (1994)
45. Hruschka, E.R., Ebecken, N.F.F.: A genetic algorithm for cluster analysis. Intelligent Data Analysis 7, 15–25 (2003)
46. Hruschka, E.R., Campello, R.J.G.B., Freitas, A.A., Carvalho, A.C.P.L.F., de: A Survey of Evolutionary Algorithms for Clustering. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 39, 133–155 (2009)
47. Huang, Z.: Clustering large data sets with mixed numeric and categorical values. In: Proceeding of 1st Pacific Asia Knowledge Discovery Data Mining Conference, pp. 21–34 (1997)
48. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Mining Knowledge Discovery 2, 283–304 (1998)
49. Huang, Z., Ng, M.K.: A fuzzy k-modes algorithm for clustering categorical data. IEEE Transactions on Fuzzy Systems 7, 446–452 (1999)
50. Hubert, L., Schultz, J.: Quadratic assignment as a general data-analysis strategy. British Journal of Mathematical and Statistical Psychology 29, 190–241 (1976)

51. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys 31, 264–323 (1999)
52. Jutler, H.: Liniejnaja modiel z nieskolmini celevymi funkcjami (linear model with several objective functions). Ekonomika i Matematiceckije Metody 3, 397–406 (1967) (in Polish)
53. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, New York (1990)
54. Kim, Y., Street, W.N., Menczer, F.: Feature selection in unsupervised learning via evolutionary search. In: Proceeding of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 365–369 (2000)
55. Kim, Y., Street, W.N., Menczer, F.: Evolutionary model selection in unsupervised learning. Intelligent Data Analysis 6, 531–556 (2002)
56. Kleinberg, J.: An impossibility theorem for clustering. In: Becker, S., Thrum, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, pp. 446–453. MIT Press, Cambridge (2002)
57. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Algorithm 8, 149–172 (2000)
58. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics 43, 59–69 (1982)
59. Kohonen, T.: Self-Organization and Associative Memory. Springer, New York (1984)
60. Kohonen, T., Kaski, S., Lagus, K., Solojärvi, J., Paatero, A., Saarela, A.: Self organization of a massive document collection. IEEE Transactions on Neural Networks 11, 574–585 (2000)
61. Korkmaz, E.E., Du, J., Alhajj, R., Barker, K.: Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. Intelligent Data Analysis 10, 163–182 (2006)
62. Langley, P.: Elements of Machine Learning. Morgan Kaufmann, San Francisco (1995)
63. Law, M.H.C., Topchy, A.P., Jain, A.K.: Multiobjective data clustering. In: Proceeding of IEEE Computer Socity Conference on Compututer Vision and Pattern Recognition, pp. 424–430 (2004)
64. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering 17, 1–12 (2005)
65. Lloyd, S.: Least squares quantization in PCM. IEEE Transactions on Information Theory 28, 129–137 (1982); Original version: Technical Report, Bell Labs (1957)
66. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceeding of 5th Berkeley Symposium Mathematical Statistics and Probability, pp. 281–297 (1967)
67. Maulik, U., Bandyopadhyay, S.: Genetic algorithm-based clustering technique. Pattern Recognition 3, 1455–1465 (2000)
68. Maulik, U., Bandyopadhyay, S., Saha, I.: Integrating clustering and supervised learning for categorical data analysis. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 40, 664–675 (2010)
69. Merz, P., Zell, A.: Clustering gene expression profiles with memetic algorithms. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 811–820. Springer, Heidelberg (2002)
70. Michalski, R.S., Carbonell, J.G., Mitchell, T.M.: An Overview of Machine Learning. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) Machine Learning, An Artificial Intelligence Approach, pp. 3–23. Springer, Berlin (1984)
71. Mierswa, I., Wurst, M.: Information preserving multi-objective feature selection for unsupervised learning. In: Proceeding of 8th ACM Annual Conference on Genetic and Evolutionary Computation, pp. 1545–1552 (2006)

72. Mierswa, I., Wurst, M.: Sound multi-objective feature space transformation for clustering. In: Proceeding of International Conference on Knowledge Discovery, Data Mining, and Machine Learning, pp. 330–337 (2006)
73. Mitra, S., Pal, S.K., Mitra, P.: Data mining in soft computing framework: a survey. IEEE Transactions on Neural Networks 13, 3–14 (2002)
74. Morita, M.E., Sabourin, R., Bortolozzi, F., Suen, C.Y.: Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition. In: Proceeding of the 7th International Conference on Document Analysis and Recognition, pp. 666–670 (2003)
75. Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S.: Multiobjective genetic fuzzy clustering of categorical attributes. In: Proceeding of 10th International Conference on Information Technology, pp. 74–79 (2007)
76. Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S.: Multiobjective genetic algorithm-based fuzzy clustering of categorical attributes. IEEE Transactions on Evolutionary Computation 13, 991–1005 (2009)
77. Nuovo, A.G.D., Palesi, M., Catania, V.: Multiobjective evolutionary fuzzy clustering for high-dimensional problems. In: Proceeding of IEEE International Conference of Fuzzy System, pp. 1–6 (2007)
78. Pan, H., Zhu, J., Han, D.: Genetic algorithms applied to multi-class clustering for gene expression data. Genomics, Proteomics, Bioinformatics 1, 279–287 (2003)
79. Pareto, V.: Manuale di Economia Politica. Piccola Biblioteca Scientifica, Milan (1906); Translated into English by Schwier, A.S., Page, A.N.: Manual of Political Economy. Kelley Publishers, London (1971)
80. Parsons, L., Haque, E., Liu, H.: Subspace clustering for high dimensional data, a review. ACM SIGKDD Explorations Newsletter 6, 90–105 (2004)
81. Ripon, K.S.N., Tsang, C.H., Kwong, S.: Multi-objective data clustering using variable-length real jumping genes genetic algorithm and local search method. In: Proceeding of IEEE International Joint Conference on Neural Networks, pp. 3609–3616 (2006)
82. Ripon, K.S.N., Tsang, C.H., Kwong, S, Ip, M.: Multi-objective evolutionary clustering using variable-length real jumping genes genetic algorithm. In: Proceeding of IEEE 18th International Conference on Pattern Recognition, pp. 1200–1203 (2006)
83. Ripon, K.S.N., Siddique, M.N.H.: Evolutionary multi-objective clustering for overlapping clusters detection. In: Proceeding of IEEE 11th International Congress on Evolutionary Computation, pp. 976–982 (2009)
84. Ritzel, B.J., Eheart, J.W., Ranjithan, S.: Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. Water Resources Research 30, 1589–1603 (1994)
85. Rumelhart, D.E., Zipser, D.: Feature discovery by competitive learning. Cognitive Science 9, 75–112 (1985)
86. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Elsevier, Amsterdam (2006)
87. Schaffer, J.D.: Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms, Ph.D. Dissertation, University of Vanderbilt (1984)
88. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proceeding of International Conference on Genetic Algorithms and their Applications, pp. 93–100 (1985)
89. Scheunders, P.: A genetic C-means clustering algorithm applied to color image quantization. Pattern Recognition 30, 859–866 (1997)
90. Sheikholeslami, G., Chatterjee, S., Zhang, A.: WaveCluster: A multi-resolution clustering approach for very large spatial databases. In: Proceeding of 24th International Conference on Very Large Data Bases, pp. 428–439 (1998)

91. Srinivas, N., Deb, K.: Multiobjective optimization using non-dominated sorting in genetic algorithms. Evolutionary Computation 2, 221–248 (1994)
92. Stewart, G.W.: On the early history of the singular value decomposition. SIAM Review 35, 551–566 (1993)
93. Surry, P.D., Radcliffe, N.J., Boyd, I.D.: A multi-objective approach to constrained optimisation of gas supply networks: The COMOGA method. In: Fogarty, T.C. (ed.) AISB-WS 1995. LNCS, vol. 993, pp. 166–180. Springer, Heidelberg (1995)
94. Tou, J.T., Gonzalez, R.C.: Pattern Recognition Principles. Addison-Wesley, Reading (1974)
95. Wang, W., Yang, J., Muntz, R.: STING: A statistical information grid approach to spatial data mining. In: Proceeding of 23rd International Conference on Very Large Data Bases, pp. 186–195 (1997)
96. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann, San Francisco (2005)
97. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. In: Proceeding of ACM International Conference on Management of Data, pp. 103–114 (1996)
98. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, Ph.D. Dissertation, Swiss Federal Institute of Technology (1999)

Tuning Meta-Heuristics Using Multi-agent Learning in a Scheduling System

Ivo Pereira[1], Ana Madureira[1], P.B. de Moura Oliveira[2], and Ajith Abraham[3,4]

[1] GECAD - Knowledge Engineering and Decision Support Group,
Institute of Engineering – Polytechnic of Porto, Porto, Portugal
{iaspe,amd}@isep.ipp.pt
[2] INESC TEC - INESC Technology and Science, Department of Engineering,
University of Trás-os-Montes e Alto Douro, Vila Real, Portugal
oliveira@utad.pt
[3] Machine Intelligence Research Labs (MIR Labs).
Scientific Network for Innovation and Research Excellence, Auburn, USA
[4] IT4Innovations - Center of Excellence, VSB-Technical University of Ostrava, Czech Republic
ajith.abraham@ieee.org

Abstract. In complexity theory, scheduling problem is considered as a NP-complete combinatorial optimization problem. Since Multi-Agent Systems manage complex, dynamic and unpredictable environments, in this work they are used to model a scheduling system subject to perturbations. Meta-heuristics proved to be very useful in the resolution of NP-complete problems. However, these techniques require extensive parameter tuning, which is a very hard and time-consuming task to perform. Based on Multi-Agent Learning concepts, this article propose a Case-based Reasoning module in order to solve the parameter-tuning problem in a Multi-Agent Scheduling System. A computational study is performed in order to evaluate the proposed CBR module performance.

Keywords: Case-based Reasoning, Learning, Metaheuristics, Parameter tuning, Scheduling.

1 Introduction

Recently, the interest in decentralized approaches for the resolution of complex real world problems, like Scheduling, is gaining much attention due to its wide applications. Several of these approaches belong to Distributed Systems research area, where a number of entities work together to solve problems in a cooperative way. In this area, it is possible to emphasize Multi-Agent Systems (MAS), concerning the coordination of agent's behaviors in order to share knowledge, abilities, and objectives, in the resolution of complex problems. Due to the exponential growing of system's complexity, it is important that MAS become more autonomous to deal with dynamism, overloads and failures recovery.

Multi-Agent Systems typically operate in open, complex, dynamic, and unpredictable environments. Therefore, learning becomes crucial. Learning is a relevant area from Artificial Intelligence (AI) as from human intelligence. Plaza et al. [1] defined learning as "the process of improving individual performance, precision (or quality) of solutions, efficiency (or speed) of finding solutions and scope of solvable problems". Although this definition is very useful, it is a severe and constricted view of learning. In a more general way, it is possible to define learning as the acquisition of new knowledge or updating existing knowledge.

As per Alonso et al. [2], intelligence implies a certain degree of autonomy, which requires the capacity of taking decisions autonomously. Thus, agents should have the appropriate tools to take such decisions. In dynamic domains it is not possible to predict every situation that an agent can find, so it is necessary that agents have the ability to adapt to new situations. This is especially true in MAS, where in many cases the global behavior emerges instead of being pre-defined. Consequently, learning is a crucial component of autonomy and pro-activeness, which must be a study target of agents and MAS [2].

The adaptation of ideas from different research areas, inspired from nature, led to the development of Meta-Heuristics (MH), which are techniques aiming to solve complex generic problems of combinatorial optimization, in which the scheduling problem is included.

Meta-heuristics are very useful to achieve good solutions in reasonable execution times. Sometimes they even obtain optimal solutions. However, to achieve near-optimal solutions, it is required the appropriate tuning of parameters.

Parameter tuning of MH has a great influence in the effectiveness and efficiency of the search process. The definition of the parameters is not obvious because they depend on the problem and the time that the user has to solve the problem [3]. Therefore, this paper proposes the use of a learning mechanism in order to perform the MH parameter tuning in the resolution of the scheduling problem. Case-based Reasoning (CBR) was chosen since it assumes that similar problems may require similar solutions.

As a MAS is used to model a dynamic scheduling system, with agents representing both tasks and resources, it is proposed that each resource agent have their own CBR module, allowing a multi-apprentice learning. With this type of learning, agents learn how to perform their own single-machine scheduling problem.

The proposed system adopts and provides parameter tuning of MH through CBR, with the possibility that parameters can change in run-time. According to the current situation being treated, the system must be able to define which MH should be used and define the respective parameters. It is even possible to change from one MH to another, according to current state and previous information, through learning and experience.

The paper is organized as follows: Section 2 describes the scheduling problem and Section 3 describes MH. Section 4 describes the Multi-Agent Learning and CBR is explained along section 5. In Section 6, the implemented MAS is explained with CBR integrated. A computational study is presented in Sections 7 and 8 and some conclusions and future work are presented.

2 Scheduling Problem

Scheduling problems are present in a large set of domains, from transports to manufacturing, computer environments, hospital settings, etc., most of them characterized by a vast amount of uncertainty leading to a considerable dynamism in the systems. Thereby, dynamic scheduling is getting an increased attention by researchers and practitioners [4][5].

The scheduling problem treated in this paper is named Extended Job-Shop Scheduling Problem (EJSSP), described by A. Madureira in 2003 [6], and has some major extensions and differences when compared to the classic JSSP, in order to better represent reality.

JSSP has a set of tasks processing in a set of machines, with each task following an ordered list of operations, each one characterized by the respective processing time and machine where is processed.

The main elements of JSSP problem are:

- a set of multi-operation jobs $J_1,\ldots,J_n$ to be scheduled on a set of machines $M_1,\ldots,M_n$
- d_j represents the due date of job J_j
- t_j represents the initial processing time of job J_j
- r_j represents the release time of job J_j

EJSSP problems consist in JSSP problems with additional restrictions, to better represent reality. Some of those restrictions are:

- Different release and due dates for each task
- Different priorities for each task
- Possibility that not every machine is used for all tasks
- A task can have more than one operation being processed in the same machine
- Two or more operations of the same task can be processed simultaneously
- Possibility of existence of alternatives machines, identical or not, for processing the operations

In this work, we define a job as a manufacturing order for a final product that can be Simple or Complex. It may be Simple Product, like a part, requiring a set of operations to be processed. Complex Products, require processing of several operations on a number of parts followed by assembly operations at several stages.

For a better understanding the EJSSP, the reader may consider the following example: a certain company produces some complex products. During the production process, new orders may arrive, some orders can be canceled, and some orders may be changed (due dates priorities, etc.). With the EJSSP modeling, it is possible to specify different priorities for each order, change the due dates, etc. But the most important contribution of this modeling strategy is that it is possible to have: i) many machines producing the same pieces; ii) more than one piece of each job processed in the same machine; iii) two or more pieces of the same job being processed at the same time. The three last aforementioned aspects are an important enhancement, not considered in the classic JSSP definition.

Scheduling problems belongs to the NP-complete class [4]. Methods for their resolution can be categorized in exact and approximation algorithms [5][6]. In the former, an exhaustive solutions space search is made and it is ensured the optimal solution, but on the other hand they are very time consuming. The latter includes heuristics and MH and do not guarantee the optimal solution since they have the objective to find a good solution in an acceptable amount of time. For this reason, they are used in this work integrated with MAS.

3 Metaheuristics Parameter Tuning

Meta-heuristics have gained popularity over the past two decades in the resolution of many types of real-life problems, including Scheduling, since they allow the resolution of large dimension problems by obtaining satisfactory solutions in satisfactory execution times. The term "meta-heuristic" was introduced by Fred W. Glover in 1986 [7].

These techniques have the objective of guiding and improving the search process in a way to overcome local optimal solutions, which represent a limitation of Local Search algorithm, and obtain solutions with satisfactory quality, very close to the optimal solution, in reasonable execution times [3][6].

Meta-heuristics consist on iterative or recursive methods with the objective of obtaining solutions the closest as possible to the global optimum for a given problem. Assuming that all solutions are interrelated, it is possible to obtain the set of solutions for a given problem.

In this work some of the most well known MH are used [3][6][8]: Tabu Search, Genetic Algorithms, Simulated Annealing, Ant Colony Optimization, and Particle Swarm Optimization.

Tabu Search (TS) was introduced by Fred Glover [7] and consists in a Local Search algorithm with the main objective to escape from local minimum. It uses a tabu list to memorize the last solutions trajectory, prohibiting the moves to solutions already visited in a short term memory.

Simulated Annealing (SA) was proposed by Kirkpatrick et al. [9] and Cerny [10]. It has connections to thermodynamics and metallurgy [11], and the original motivation is based on the process in which molten metal is slowly cooled, with a tendency to solidify in a structure of minimum energy. This MH has a statistical basis and is based on allowing the movement to a worst solution, with the objective to escape from local optimum.

In beginning of 1970, John Holland, together with his students and colleagues, developed research and studies based on natural selection of species, reaching a formal model designated by Genetic Algorithms (GA) [12]. In the 1980s, David Goldberg, Holland's student, implemented and published the first well successful applications of these algorithms [13].

Proposed by Dorigo et al. [14], Ant Colony Optimization (ACO) is based on a behavior that allows ants to find the shortest path between a food source and the respective colony [7] Ants deposit in the ground a substance named pheromone, and when choosing a path, they opt, with greater probability, by the one that have more quantity of pheromone, which corresponds to the path followed by the higher number of ants.

Particle Swarm Optimization (PSO) was developed by James Kennedy and Russell Eberhart [15] with the objective to simulate a simplified social system. The basic idea was to demonstrate the behavior that flocks of birds or schools of fishes assume in their random local trajectories, but globally determined. Flocks of birds or schools of fishes make coordinated and synchronized movements as a way of finding food or as a mechanism of self-defense.

As mentioned, MH can be used for the resolution of many kinds of problems. However, to solve a specific problem it is necessary to choose a MH, which is considered a difficult task, requiring a study about the problem type and about the chosen technique. Furthermore it is also necessary to define the respective parameters.

The parameter tuning of MH allows greater flexibility and robustness but requires a careful initialization, since parameters have a great influence on the efficiency and effectiveness of the search [3].

El-Ghazali Talbi [3] has identified two different approaches for MH parameter tuning: offline and online (Fig. 1). In offline tuning, the values for the parameters are defined before the execution of MH. In online tuning the parameters are controlled and updated in a dynamic or adaptive way, throughout the execution of MH.

Usually, when using MH, practitioners tune one parameter at a time and its optimal value is determined in an empiric way. However, this tuning strategy cannot guarantee the optimal parameter configuration.

To overcome this problem, design of experiments (DOE) [16] is used. Nevertheless, before using DOE it is necessary to take into account diverse factors which represent the parameters variation and the different values for each parameter (that can be quantitative and qualitative).

The greatest disadvantage about using DOE is the high computational cost when there is a large number of parameters, and when the domains of the respective values are also high since it is necessary to perform a large number of experiments [17]. To overcome this disadvantage, it is possible to use, e.g., racing algorithms [18][19].

On the other hand, in Meta-optimization, (meta) heuristics can be used to find the optimal parameters like in optimization problems. Meta-optimization consists in two levels: meta-level and base level. In the meta-level, solutions represent the parameters to optimize, such as the size of the tabu list in Tabu Search, the cooling rate in Simulated Annealing, the crossover and mutation rates of a Genetic Algorithm, etc. At this level, the objective function of a solution is the best solution found (or another performance indicator) by the MH with the specified parameters. Thus, for each solution in the meta-level there is an independent MH in the base level.

The drawback of offline approaches is the high computational cost, especially if used for each instance of the problem. In fact, the optimum values for the parameters depend on the problem to solve and on the different instances (e.g. larger instances may require different parameter settings). Thus, to increase the effectiveness and robustness of offline approaches, these should be applied to all instances (or class of instances) of a given problem [3].

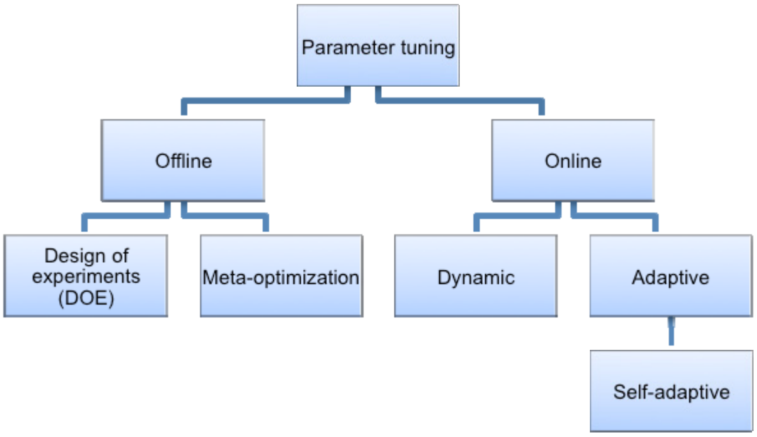


Fig. 1. Parameter tuning [3]

Online approaches arise in order to try to achieve better results and they can be divided in dynamic and adaptive approaches [3]. In dynamic approaches, changes in parameter values are performed at random or deterministic ways, without taking into account the search process. In adaptive approaches, parameter values change according to the search process through the use of memory. A subclass, often used in the evolutionary computation community, is identified as self-adaptive, consisting in parameters evolution during the search. Therefore, the parameters are encoded and are subject to change, such as solutions to the problem.

This problem of finding the most suitable parameter configuration is related with the notion of hyper-heuristic [20][21][22]. Hyper-heuristic methods try to automate the process of selecting, combining or adapting several heuristics (or MH) in order to solve problems in an efficient manner.

The term “hyper-heuristic” was introduced in 1997 [23] to describe a procedure combining different AI methods. This idea became pioneer in the 1960s with the combination of scheduling rules [24][25] and has been used to solve many optimization problems [21]. The term “hyper-heuristic” was independently used in 2000 [26] to describe “heuristics that choose heuristics” in the context of combinatorial optimization. In this context, a hyper-heuristic is a high-level approach which, given a particular instance of the problem and a number of low-level heuristics, can choose and apply an appropriate low level heuristic at each decision point [27][28].

In the literature it is possible to find a wide variety of hyper-heuristic approaches using high-level methodologies along with a set of low level heuristics applied to different optimization problems. However, there is no reason to limit the a high-level strategy to a heuristic. In fact, the sophisticated knowledge-based techniques such as CBR have been employed to this end with successful results for solving the university timetables problem [29]. This led to a more general definition for the term "hyper-heuristic", whose goal is to capture the idea of a method to automate the design of heuristics and the selection process: “A hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems" [20].

The defining characteristic on hyper-heuristics research is that it investigates methodologies operating within a search space of heuristics rather than directly on a search space of problem solutions. This feature provides the potential to increase the level of general research methods. Several approaches for hyper-heuristics have been proposed that incorporate different research paradigms and machine learning [20].

The research on hyper-heuristics in based on the compromise between search methodologies and machine learning. Machine learning is a well established field of AI and its exploitation to automate the design of heuristics is still at the beginning, but it is expected big developments in the future [20].

4 Multi-agent Learning

In AI, machine learning is a research area concerning the development of algorithms and techniques in order to provide computers with learning faculties. Commonly accepted in the literature, machine learning algorithms and techniques can be classified in three categories:

- Supervised learning (where data have labels or classes);
- Unsupervised learning (data have no labels);
- Reinforcement learning (where the objective is to maximize a reward).

Some authors refer another category, placed between Supervised and Unsupervised learning, named Semi-Supervised learning, that uses both labeled and not labeled data. It is also very common the reference to another category, known by Instance-based Learning [30] or Non-Parametric Methods [31], where CBR can be included, described in the next section.

It is possible to apply machine learning concepts to many research areas, including natural language processing, pattern recognition, market analysis, DNA sequences classification, speech and handwriting recognition, object recognition in computer vision, game playing and robot locomotion.

Panait and Luke [32] have focused machine learning application to problems related with MAS. They use machine learning in order to explore ways to automate the inductive process, e.g., put a machine agent to find by itself how to solve a task or minimize error. They have referred that machine learning is a popular approach for the resolution of MAS problems because the complexity intrinsic to many of those problems can make solutions prohibitively hard to obtain.

In the next subsections, it will be described four learning techniques used in MAS, namely Reactive learning, Social learning, Team learning and Concurrent learning.

4.1 Reactive Learning

In reactive systems, the cooperative behavior emerges from the interaction between agents. Instead of implementing coordination protocols or providing complex recognition models, it is assumed that agents work with value-based information (e.g. the distance they should keep from neighbors) which produces the social behavior. Once internal processing is avoided, these techniques allow MAS reacting to changes in an efficient way [2].

As a collateral effect, agents do not know the domain, which is crucial to take decisions in complex and dynamic scenarios. In this view, it is not possible to simulate complex social interactions and, in order to have high-level behaviors, agents need to summarize experiences in concepts. An entity that can conceptualize can also transform experience in knowledge and guide the vital resources until necessary [2].

4.2 Social Learning

Social learning is composed by learning mechanisms arising from AI and Biology.

In persistent MAS, where new agents enter a world already populated with experienced agents, a new agent starts with a blank state and has not had yet the opportunity to learn about the environment. However, a new agent does not need to discover everything about the environment since it can benefit from the accumulated learning from the experienced population of agents [2].

An important difference between artificial agents and animals is that, in the first, it is possible to simulate a completely cooperative scenario, where exists a common utility function. Even though cooperation occurs in many animal species, the possibility of conflicts emerging is always present, due to the competition in genes' self-replication of evolutionary process [2].

There are several different ways to an agent learn from other agents behaviors. Despite the existence of imitation (direct copy from other agents behaviors), this has proved to be complex since it involves not only the behaviors' understanding and reproducing but also the understanding of the changes in the environment caused by these behaviors [2].

4.3 Team Learning

In Team Learning it only exists an apprentice. However, it has the objective to discover a subset of behaviors for a team of agents, instead for a unique agent. It is a simple approach to Multi-Agent learning because the apprentice can use machine learning techniques, which avoid the difficulties emerging from the co-adaptation of multiple agents in Concurrent Learning approaches. Another advantage in the using of a unique apprentice agent is that it only cares about the team performance, and not with itself. For this reason, Team learning approaches can ignore the inter-agent credit assignment that is usually hard to determine [32].

However, Panait and Luke [32] also pointed some disadvantages in the use of Team learning. The main problem refers to the large state space for the learning process, which can be devastating for learning methods that explore the utility state space (such as Reinforcement learning) but cannot affect so drastically techniques that explore the behaviors space (such as Evolutionary computing). A second disadvantage refers to the learning algorithm centralization problem: every resource need to be available in the same place where the program will be executed. This can be uncomfortable for domains where data are inherently distributed.

Team learning can be divided in homogeneous and heterogeneous [32]. Homogeneous apprentices develop an unique identical behavior for each agent, even if agents are different. Heterogeneous apprentices must deal with a large search space, but with the guarantee to get better solutions through agents' specialization. The choice between approaches depends if experts are necessary in the team.

4.4 Concurrent Learning

The most common alternative to Team learning is Concurrent learning, where multiple apprentices try to improve parts from the team. Typically, each agent has its own learning process to modify the behaviors [32].

The main difficulty subjacent to Concurrent learning is to know in which domains it achieves better results when compared with Team learning. Jansen and Wiegand [33] argue that Concurrent learning can perform better in domains where decomposition is possible and helpful (such as Scheduling), and when it is useful to focus each sub-problem regardless others. This happens because Concurrent learning separates the search space into smaller ones. If the problem can be decomposed, such that agents' individual behaviors are relatively disjoint, it can result in a significant reduction of the search space and computational complexity. Another advantage is that decomposing the learning process into smaller pieces allows a greater flexibility using computational resources in each process learning, since they can, at least partially, be learned regardless others.

The main challenge of Concurrent learning consists in the adaption of each apprentice behaviors to the context of others, which its cannot control. In single agent scenarios, an apprentice explores his environment and improves his behavior. But things are quite different when using multiple apprentices: while agents learn, they change the behaviors, which can ruin the learned behaviors by other agents, making outdated assumptions [34][35]. A simple approach to deal with this co-adaptation is to treat other apprentices as part of the dynamic environment for which each apprentice must adapt [36].

In this research, we propose a concurrent learning approach, in which several agents learn about their internal behaviors and environment.

5 Case-Based Reasoning

Case-Based Reasoning (CBR) is an Artificial Intelligence technique that aims to solve new problems by using information about the resolution of previous similar problems [37]. As previously described, CBR represents a method of ML Instance-based Learning and uses the principle that similar problems may require similar solutions [38] on a direct analogy to learning based on past experience.

CBR roots are found in the work of Roger Schank about dynamic memory and how the memory of previous situations can affect problems' resolution and learning processes [39]. There are also references about the study of analogical reasoning [40].

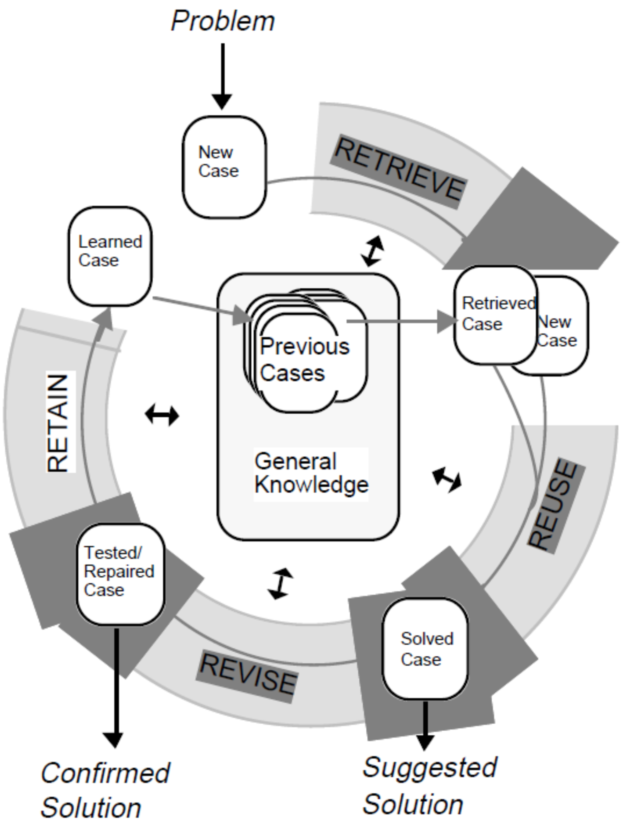


Fig. 2. The CBR cycle [42]

CYRUS system, developed by Janet Kolodner [37] was the first known CBR system. It was based on Schank's dynamic memory model [39] and, basically, consisted of a question-answer system with knowledge about the different travels and meetings of USA ex-Secretary of State Cyrus Vance. Another first system to use CBR was PROTOS, developed by Bruce Porter et al. [41], which dealt with ML classification problem.

The CBR cycle is illustrated in Fig. 2 and consists of four main phases [38][42]:

1. Retrieve the most similar case or cases
2. Reuse the retrieved information and knowledge
3. Revise the proposed solution
4. Retain the revised solution for future use

In CBR, previous solved cases and their solutions are memorized as cases in order to be reused in the future [38]. These cases are stored in a repository named casebase. Instead of defining a set of rules or general lines, a CBR system solves a new problem by reusing similar cases that were previously solved [43].

A new case of the problem to be solved is used to retrieve an old case from the casebase. In the Reusing phase, the retrieved case is analyzed in order to suggest a solution for the resolution of the new case. In the Revising phase, this suggested solution is tested, for example, by executing it in the system, and repaired if it fails. In the Retaining phase, the useful experience is retained for future use, and the casebase is updated with the new learned case (or by modifying some existing cases).

In the Reusing phase, it is possible to reuse a solution or a method. In solution reuse, the past solution is not directly copied to the new case, but there is some knowledge allowing the previous solution to be transformed into the new case solution. In case of method reuse, it is observed how the problem was solved in the retrieved case, which has information about the method used for the problem resolution, including an explanation about the used operators, sub-objectives considered, generated alternatives, failures, etc. The retrieved method is then reused to the new problem resolution, in the new context.

The objective of Revising phase is to evaluate the retrieved solution. If this solution is well succeeded it is possible to learn about the success, otherwise the solution is repaired using some problem domain's specific knowledge. The evaluating task applies the proposed solution in an execution environment and the result is evaluated. This is usually a step outside the CBR, once the problem may be executed in an application.

Finally, the Retaining phase consists in the integration of the useful information about the new case resolution into the casebase. It is necessary to know which information is important to retain, how to retain it, how to index the case for a future retrieve, and how to integrate the new case in the memory structure.

Burke et al. [44] referred that CBR is an appropriate approach for scheduling systems with expertise knowledge, and highlighted a research potential in dynamic scheduling.

Generally, CBR applications for scheduling domain can be classified in three categories [43]:

- Algorithms reuse - assume that it is probable that an effective approach for a specific problem's resolution will also be effective in the resolution of a similar problem. In these systems, a case consists in a representation of the problem and in a known effective algorithm for its resolution. Schmidt [45] designed a CBR structure to choose the most appropriate method for the resolution of scheduling problems in production scheduling. Schirmer [46] implemented a CBR system for selecting scheduling algorithms for the resolution of project scheduling problems. It was experimentally shown that some scheduling algorithms work better than others, in some instances of problems.
- Operators reuse - reuse the operators for the resolution of the new problem [44]. A case describes a context in which a useful scheduling problem is used for repairing/adapting a scheduling plan to improve its quality, in terms of constraints satisfaction [38]. Burke et al. [44] have proposed a case-based hyper-heuristic to solve timetabling problems. Beddoe et al. [38] have developed a CBR system to solve nurse scheduling problems.

- Solutions reuse - it is used the whole or part of previous problems' solutions to construct the solution of the new problem. A case contains the description of a problem and its solution, or part of solution. This method was used for the resolution of manufacturing scheduling problems [47][48] and university courses timetabling [44]. It was also used for constructing MH' initial solutions, as Genetic Algorithms [49] and Simulated Annealing [50].

6 Multi-agent Scheduling System

The developed MAS for the resolution of Scheduling problem consists in a hybrid autonomous architecture [51]. As illustrated in Fig. 3, there are three kinds of agents.

The proposed MAS have agents representing jobs/tasks and agents representing resources/machines. The system is able to find optimal or near optimal solutions through the use of MH, dealing with dynamism (arriving of new jobs, cancelled jobs, changing jobs attributes, etc.), change/adapt the parameters of the algorithm according to the current situation, switch from one MH to another, and perform a coordination between agents through cooperation or negotiation mechanisms.

Job agents process the necessary information about the respective job. They are responsible for the generation of the earliest and latest processing times on the respective job and automatically separate each job's operation for the respective Resource Agent.

Resource agents are responsible for scheduling the operations that require processing in the machine supervised by the agent. These agents implement MH in order to find the best possible single-machine schedules/plans of operations and communicate those solutions to the AgentUI for later feasibility check.

Since it is impossible to predict each problem to treat, the system should be capable of learning about its experience during lifetime, as humans do. To perform this learning mechanism, it is proposed the use of CBR within Resource agents.

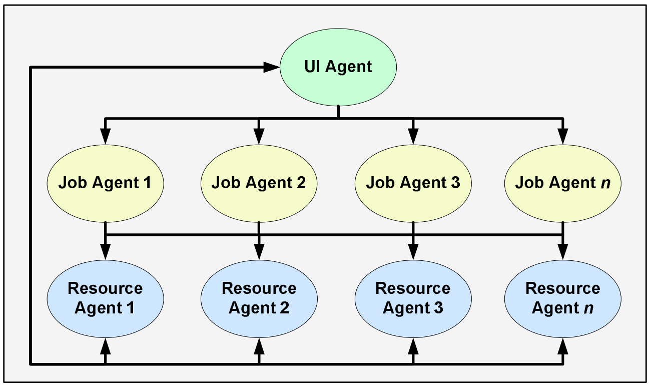


Fig. 3. Multi-agent Scheduling System

6.1 CBR Module

The proposed CBR approach [51] consists in retrieving the most similar case or cases to the new problem, regardless the MH to be used, as well as its parameters. It is important for the system to decide which technique and respective parameters may be used, because not every MH is suitable to all types of problems.

The main objective of CBR module is to choose a MH to be used by the respective Resource Agent in which the CBR is included. The secondary objective is to perform the parameter tuning of MH, according to the problem to solve. Based on past experience, each case contains the MH and the respective parameters. If the parameters were effective and efficient in the resolution of a similar case, then they have a great probability to be effective and efficient in the resolution of the new problem. It is possible to describe our CBR module as a hyper-heuristic approach but since it performs a self-parameterization of MH it is more appropriate to see it as a parameter tuning approach.

It is important to notice that, like previously described in Fig. 2, every new problem or perturbations occurred leads to a new case in the system, with the previous most similar cases being retrieved from the casebase. After that, the better case is reused, becoming a suggested solution. After the solution revision, the case is executed in the MAS. This revision is performed to allow escaping from local optimal solutions and MH stagnation, since it is used some disturbance in the parameters of the proposed solution. After the conclusion of the MAS execution, the case is confirmed as a good solution, being retained on the database as a new learned case, for future use.

Figure 4 illustrates the inclusion of CBR in the system. Each Resource Agent has its own CBR module. With this approach, different MH may be chosen in the resolution of the same Job-Shop problem. This can be considered as an advantage because the Resource Agents can have different number of operations to schedule. Some MH are more suitable to schedule problems with large number of operations than others.

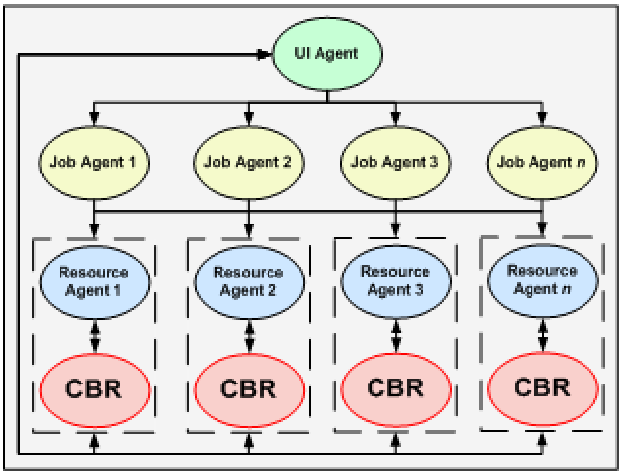


Fig. 4. CBR module within Resource agents

The most important part of a CBR module is its similarity measure because it decides how much two cases are similar between each other. The similarity measure of the proposed CBR module is very simple and is defined in equation (1).

$$Similiarity = \frac{NumberOperations_{CaseA}}{NumberOperations_{CaseB}} \tag{1}$$

As previously mentioned, each Resource Agent has a number of operations to schedule. This number of operations can be different, depending on the problem to treat, and is enough to define a problem. The MH and the respective parameters may be chosen according to the dimension of the problem to treat. So, with this similarity measure it is possible to have a ratio between two cases. The similarity is a value in the interval [0,1], whose limits correspond to non similar and completely similar cases, respectively. If there are more than one case very similar to the problem to be solved, the most effective and efficient case is reused.

If some perturbations occur in the problem, the MH and the parameters may change, because a different problem may be solved. For example, if new jobs arrive or if some jobs are canceled, the problem's dimension is different and so other MH and/or other parameters may be used. This decision is autonomously performed by the CBR module in run time.

7 Computational Results

The main objective of this computational study is to analyze the integration of CBR in an effective and efficient way, comparing the system's performance with CBR included versus the system's performance before the integration of CBR. Another objective is to obtain some conclusions about the usage of MH in the resolution of Job-Shop instances, after the integration of CBR.

For the computational study, all instances from OR-Library Job-Shop Scheduling problems were used [52] (a total of 82 instances), proposed by Adams, Balas and Zawack [53], Fisher and Thompson [54], Lawrence [55], Applegate and Cook [56], Storer, Wu and Vaccari [57], and Yamada and Nakano [58]. These instances cover problems with 10, 20, 30, and 50 jobs and they were executed five times (before and after CBR integration).

The machine used for the computational study is a HP Z400 Workstation, with the following main characteristics: Intel® Xeon® CPU W3565 @ 3.20 GHz, 6GB RAM, Samsung HD103SJ disk with 1TB, and Windows 7, 64-bit.

To conclude about the effectiveness and efficiency of the proposed CBR module, the average makespan (Cmax - conclusion time) and execution time were analyzed (Fig. 5). About the effectiveness, the average makespan was improved in 15,85% of the cases. This is considered a good improvement that can be better with the lifetime of CBR module.

Although a new module has been integrated into the MAS, the average execution times were improved in 2,44% of the cases when comparing to the previous obtained results (Fig. 5). It was not expected to improve this performance measure but it we can conclude that the parameters are becoming more efficient with the lifetime of CBR module.

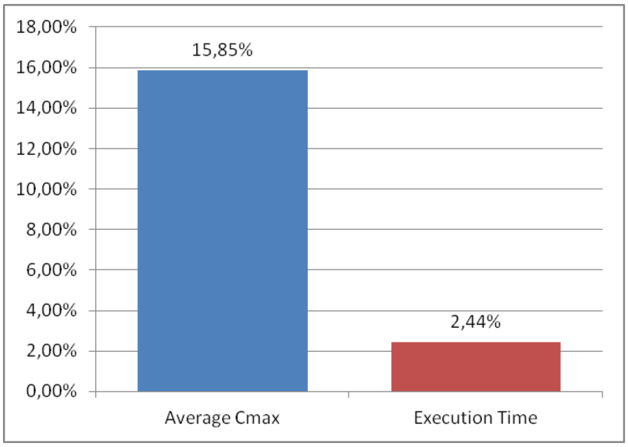


Fig. 5. Improvement of obtained average results (%)

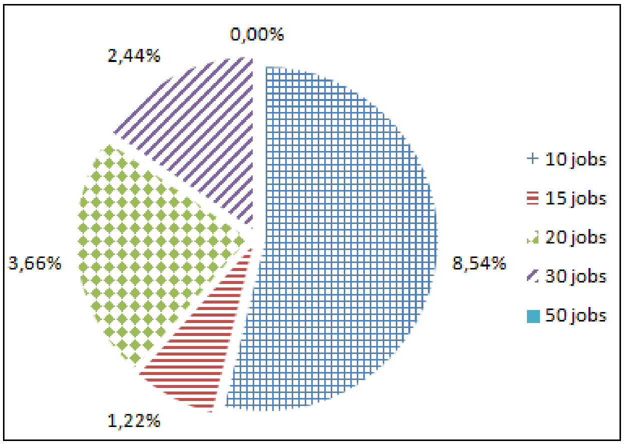


Fig. 6. Improvement of obtained average C_{max} results separated by instances dimension (%)

Figure 6 presents a detailed view about the improvement of average Cmax. For 10 jobs instances, 8,54% of the results were improved. For 20 jobs instances, 3,66% results were improved. For 15 and 30 instances the results were improved only by 1,22% and 2,44% respectively. The obtained results for 50 jobs instances were not improved at all.

In addition to the obtained conclusions about the effectiveness and efficiency of CBR it is also possible to analyze the usage of MH. With this it is possible to know which MH were used most. In a global perspective (Fig. 7), PSO was the most used MH, in 36,96%, and then GA with 17,98%. TS and SA were used in 15,73% of the instances. Finally, ACO was the less used with 14,61%.

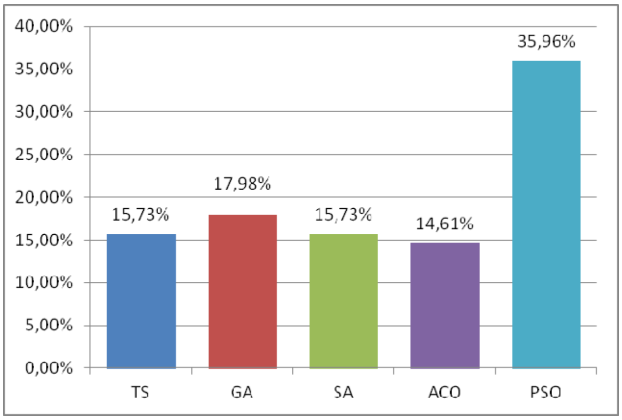


Fig. 7. Global use of MH

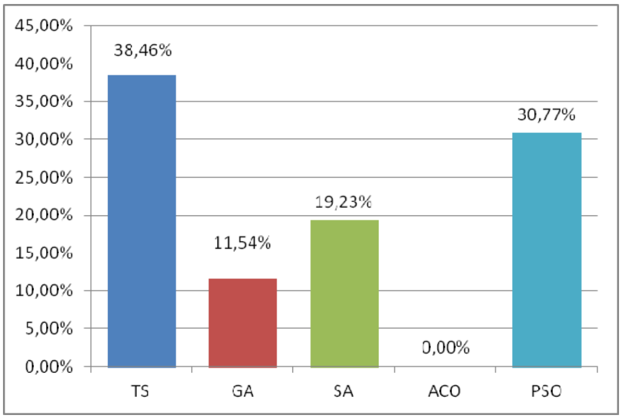


Fig. 8. MH use for 10 jobs instances

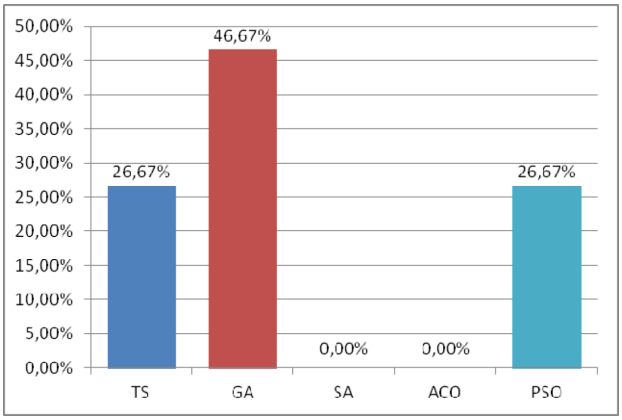


Fig. 9. MH use for 15 jobs instances

Figure 8 presents the MH use for 10 jobs instances. TS and PSO were the most used techniques in the smallest dimension problems' instances. ACO was not used at all.

Figure 9 presents the MH used for 15 jobs instances. GA was the most used MH with 46,67%. TS and PSO were the other techniques used in 26,67% of the cases. SA and ACO were not used in the resolution of this class of instances.

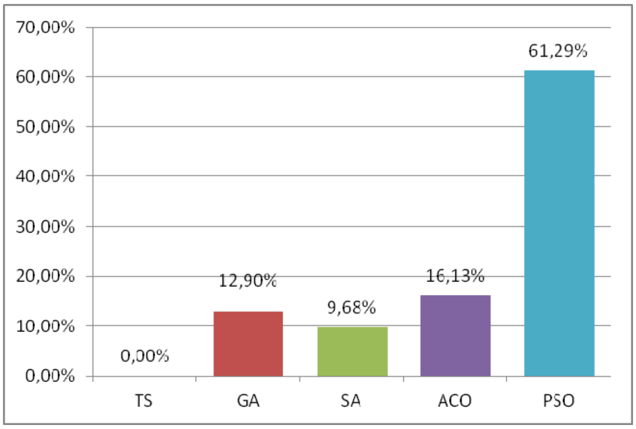


Fig. 10. MH use for 20 jobs instances

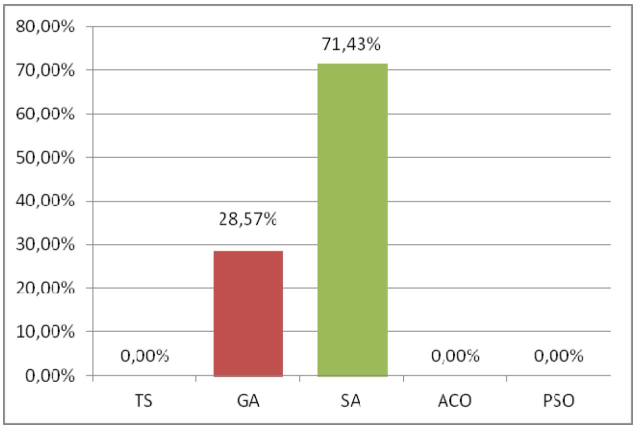


Fig. 11. MH use for 30 jobs instances

Only two MH were used in the resolution of 30 jobs instances, as shown in Fig. 11. SA was the most used technique in 71,43% of the cases. GA was the other used MH, in 28,57%.

Finally, for 50 jobs instances, ACO was the most used MH in 80% of the cases. The other used MH were SA and PSO in 10% of the cases each (Fig. 12).

Concluding, for small instances (10 and 15 jobs) TS, GA and PSO revealed to be the most used, but ACO was not used at all. For 20 and 30 jobs instances PSO and SA were the most used respectively. For large dimension instances ACO was the most used.

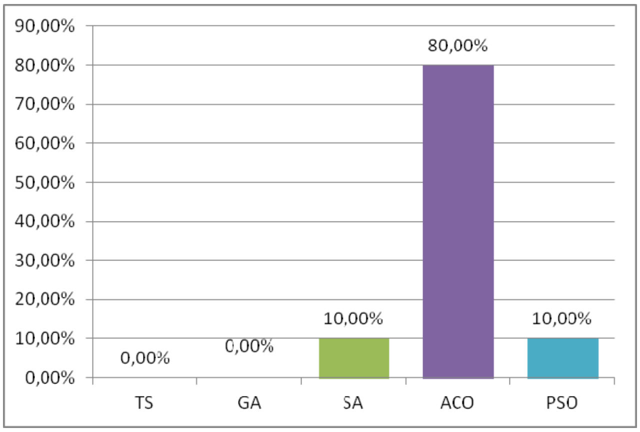


Fig. 12. MH use for 50 jobs instances

8 Conclusions

In this paper the use of CBR was proposed in order to perform MH parameter tuning in the resolution of Job-Shop scheduling problem.

The presented scheduling system consists in a MAS with different agents representing both jobs and resources. The proposed CBR module is included in resource agents with the objective to chose the best MH and perform the respective parameter tuning. The MH choice and parameters configuration is done based on past experience, since CBR assumes that similar cases may have similar solutions.

From the computational study presented it is possible to conclude that the system became more effective in 15,85% of the cases and more efficient in 2,44%.

It was also possible to conclude that, in the resolution of academic Job-Shop instances, PSO was, globally, the most used technique, in a global perspective. However, in the resolution of small dimension instances TS revealed to be the most used, and in the resolution of very large dimension instances ACO was the most used.

For future work we intend to do more experiments with the CBR module. It is expected that the effectiveness and efficiency of CBR improves during the lifetime.

Acknowledgments. This work is supported by FEDER Funds through the "Programa Operacional Factores de Competitividade - COMPETE" program and by National Funds through FCT "Fundação para a Ciência e a Tecnologia" under the project: FCOMP-01-0124-FEDER-PEst-OE/EEI/UI0760/2011 and PTDC/EME-GIN/109956/2009. The first author would like also to thanks FCT for the Ph.D. scholarship SFRH/BD/63404/2009. This work is partially supported in the framework of the IT4 Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 by operational programme 'Research and Development for Innovations' funded by the Structural Funds of the European Union and state budget of the Czech Republic, EU.

References

1. Plaza, E., Arcos, J., Martin, F.: Cooperative Case-Based Reasoning. In: Weiss, G. (ed.) ECAI 1996 Workshops. LNCS (LNAI), vol. 1221, pp. 180–201. Springer, Heidelberg (1997)
2. Alonso, E., D'inverno, M., Kudenko, D., Luch, M., Noble, J.: Learning in Multi-agent Systems. The Knowledge Engineering Review 16(3), 277–284 (2001)
3. Talbi, E.-G.: Metaheuristics - From Design to Implementation. Wiley (2009)
4. Baker, K.R., Trietsch, D.: Principles of Sequencing and Scheduling. John Wiley & Sons, Inc. (2009)
5. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems, 4th edn. Springer (2012)
6. Madureira, A.: Meta-heuristics application to scheduling in dynamic environments of discrete manufacturing, Ph.D. thesis, University of Minho, Braga, Portugal (2003) (in Portuguese)
7. Glover, F.: Future paths for integer prog. and links to artificial intelligence. Comp. & Ops. Res. 5, 533–549 (1986)
8. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35, 268–308 (2003)
9. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
10. Cerny, V.: A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm. J. Optim. Theory Appl. 45, 41–51 (1985)
11. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equations of State Calculations by Fast Computing Machines. Journal of Chemical Physics 21(6), 1087–1092 (1953)
12. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan (1975)
13. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)
14. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: an autocatalytic optimizing, Technical Report, TR91-016, Milano (1991)
15. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks (1995)
16. Box, G., Hunter, J.S., Hunter, W.G.: Statistics for Experimenters: Design, Innovation, and Discovery. Wiley (2005)
17. Schaffer, J.D., Caruana, R.A., Eshelman, L., Das, R.: A study of control parameters affecting online performance of genetic algorithms for function optimization. In: Schaffer, J.D. (ed.) International Conference on Genetic Algorithms, 3rd edn., pp. 51–60. Morgan Kaufman, San Mateo (1989)
18. Maron, O., Moore, A.W.: Hoeffding races: Accelerating model selection search for classification and function approximation. In: Advances in Neural Information Processing Systems, vol. 6, pp. 59–66. Morgan Kaufmann, San Francisco (1994)
19. Birattari, M., Stutzle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 11–18. Morgan Kaufmann, San Francisco (2002)
20. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: A classification of hyper-heuristics approaches. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, 2nd edn. Springer (2009)

21. Hamadi, Y., Monfroy, E., Saubion, F.: An Introduction to Autonomous Search. In: Hamadi, Y., et al. (eds.) Autonomous Search. Springer (2012) ISBN 978-3-642-21433-2
22. Cowling, P.I., Kendall, G., Soubeiga, E.: Hyperheuristics: a tool for rapid prototyping in scheduling and optimisation. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002. LNCS, vol. 2279, pp. 1–10. Springer, Heidelberg (2002)
23. Denzinger, J., Fuchs, M., Fuchs, M.: High performance ATP systems by combining several AI methods. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 1997), USA, pp. 102–107 (1997)
24. Crowston, W., Glover, F., Thompson, G., Trawick, J.: Probabilistic and parametric learning combinations of local job shop scheduling rules, Tech. rep., ONR Research Memorandum No. 117, GSIA, Carnegie-Mellon University, Pittsburg (1963)
25. Fisher, H., Thompson, L.: Probabilistic learning combinations of local job-shop scheduling rules, Industrial Scheduling. Prentice Hall (1963)
26. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach for scheduling a sales summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)
27. Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 457–474 (2003)
28. Ross, P.: Hyper-heuristics. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, ch.17, pp. 529–556. Springer, Berlin (2005)
29. Burke, E., Petrovic, S., Qu, R.: Case based heuristic selection for timetabling problems. Journal of Scheduling 9(2), 115–132 (2006)
30. Mitchell, T.: Machine Learning. McGraw-Hill Education, ISE Editions (1997)
31. Alpaydin, E.: Introduction to Machine Learning, Adaptive Computation and Machine Learning. The MIT Press (2004)
32. Panait, L., Luke, S.: Cooperative Multi-Agent Learning: The State of the Art, Autonomous Agents and Multi-Agent Systems, pp. 387–434 (2005)
33. Jansen, T., Wiegand, R.P.: Exploring the explorative advantage of the cooperative coevolutionary (1+1) EA. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 310–321. Springer, Heidelberg (2003)
34. Sandholm, T.W., Crites, R.H.: On multiagent Q-learning in a semi-competitive domain. In: Weiss, G., Sen, S. (eds.) IJCAI-WS 1995. LNCS (LNAI), vol. 1042, pp. 191–205. Springer, Heidelberg (1996)
35. Weinberg, M., Rosenschein, J.: Best-response multiagent learning in non-stationary environments. In: AAMAS 2004 Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (2004)
36. Schmidhuber, J., Zhao, J.: Multi-agent learning with the success-story algorithm. In: Weiss, G. (ed.) ECAI 1996 Workshops. LNCS (LNAI), vol. 1221, pp. 82–93. Springer, Heidelberg (1997)
37. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann Publishers Inc. (1993)
38. Beddoe, G., Petrovic, S., Li, J.: A hybrid metaheuristic case-based reasoning system for nurse rostering. Journal of Schedling 12(2), 99–119 (2009)
39. Schank, R.: Dynamic memory; a theory of reminding and learning in computers and people. Cambridge University Press (1982)
40. Gentner, D.: Structure mapping - a theorical framework for analogy. Cognitive Science 7, 155–170 (1983)

41. Porter, B., Bareiss, R.: PROTOS: An experiment in knowledge acquisition for heuristic. In: Proceedings of the First International Meeting on Advances in Learning (IMAL), Les Arcs, France (1986)
42. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. Artificial Intelligence Communications 7, 39–52 (1994)
43. Petrovic, S., Yang, Y., Dror, M.: Case-based selection of initialisation heuristics for metaheuristic examination timetabling. Expert Syst. Appl. 33, 772–785 (2007)
44. Burke, E.K., MacCarthy, B.L., Petrovic, S., Qu, R.: Knowledge Discovery in a Hyper-Heuristic for Course Timetabling Using Case-Based Reasoning. In: Burke, E., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 276–287. Springer, Heidelberg (2003)
45. Schmidt, G.: Case-based reasoning for production scheduling. International Journal of Production Economics 56-57, 537–546 (1998)
46. Schirmer, A.: Case-based reasoning and improved adaptive search for project scheduling. Naval Research Logistics 47, 201–222 (2000)
47. Coello, J.M.A., Camilo dos Santos, R.: Integrating CBR and heuristic search for learning and reusing solutions in real-time task scheduling. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 89–103. Springer, Heidelberg (1999)
48. MacCarthy, B., Jou, P.: Case-based reasoning in scheduling. In: Proceedings of the Symposium on Advanced Manufacturing Processes, Systems and Techniques (AMPST 1996). MEP Publications Ltd. (1996)
49. Oman, S., Cunningham, P.: Using case retrieval to seed genetic algorithms. International Journal of Computational Intelligence and Applications 1(1), 71–82 (2001)
50. Cunningham, P., Smyth, B.: Case-Based Reasoning in Scheduling: Reusing Solution Components. The International Journal of Production Research 35, 2947–2961 (1997)
51. Pereira, I., Madureira, A., de Moura Oliveira, P.: Multi-apprentice learning for metaheuristics parameter tuning in a Multi Agent Scheduling System. In: 2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 31–36 (2012)
52. OR-Library, `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`
53. Adams, J., Balas, E., Zawack, D.: The shifting bottleneck procedure for job shop scheduling. Management Science 34, 391–401 (1988)
54. Fisher, H., Thompson, G.L.: Probabilistic learning combinations of local job-shop scheduling rules. In: Muth, J.F., Thompson, G.L. (eds.) Industrial Scheduling, pp. 225–251. Prentice Hall (1963)
55. Lawrence, S.: Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement). Carnegie-Mellon University, Pittsburgh (1984)
56. Applegate, D., Cook, W.: A computational study of the job-shop scheduling instance. ORSA Journal on Computing 3, 149–156 (1991)
57. Storer, R.H., Wu, S.D., Vaccari, R.: New search spaces for sequencing instances with application to job shop scheduling. Management Science 38, 1495–1509 (1992)
58. Yamada, T., Nakano, R.: A genetic algorithm applicable to large-scale job-shop instances. In: Manner, R., Manderick, B. (eds.) Parallel Instance Solving from Nature, vol. 2, pp. 281–290. North-Holland (1992)

Vertical Transfer Algorithm for the School Bus Routing Problem

Ocotlán Díaz-Parra, Jorge A. Ruiz-Vanoye,
Ma. de los Ángeles Buenabad-Arias, and Ana Canepa Saenz

Departamento de Ciencias De la Información
Universidad Autónoma del Carmen
Ciudad del Carmen, Campeche, México
ocotlan@diazparra.net

Abstract. In this paper is a solution to the School Bus Routing Problem by the application of a bio-inspired algorithm in the vertical transfer of genetic material to offspring or the inheritance of genes by subsequent generations. The vertical transfer algorithm or Genetic algorithm uses the clusterization population pre-selection operator, tournament selection, crossover-k operator and an intelligent mutation operator called mutation-S. The use of the bio-inspired algorithm to solve SBRP instances show good results about Total Bus Travel Distance and the Number of Buses with the Routes.

Keywords: Transportation, Combinatorial Optimization, Algorithms, School Bus Routing Problem, SBRP, bio-inspired algorithm.

1 Introduction

The School Bus Routing Problem (SBRP) is a significant problem in the management of school bus fleet for the transportation of students, each student must be assigned to a particular bus which must be routed in a efficient manner so as to pick up (or return home) each of these students [29]. The SBRP consist of smaller sub-problems [14]: Bus Stop Selection seeks to select a set of bus stops and assign students to these stops, Bus Route Generation generate a bus route for a single school, School Bell Time Adjustment consider the starting and ending time of schools to maximize the number of routes by a bus and to reduce the number of busses used for a multi-school, Route Scheduling specifies the starting and ending time of each route of a bus for a multi-school.

The characteristics of SBRP [31] are: Number of School (single or multiple), surroundings of service (urban or rural), Problem scope (morning, afternoon, both), Mixed Load (allowed or no allowed), special-educations students (considered or not considered), Fleet mix (homogeneous fleet or heterogeneous fleet), Objectives (number of buses used, total bus travel distance or time, Total students riding distance or time, student walking distance, load balancing, maximum route length, Child´s time loss), Constraints (vehicle capacity, maximum riding time, school time windows, maximum walking time or distance, earliest pick-up time, minimum student number to create a route).

The mathematical model of SBRP [20] is formed by the equations1-8:

$$\min\ z = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{M} C_{ij} X_{ijk} \tag{1}$$

$$\sum_{k=1}^{M}\sum_{j=0}^{n} X_{ijk} = 1 \quad i = 1,2,\ldots,n \tag{2}$$

$$\sum_{k=1}^{M}\sum_{i=0}^{n} X_{ijk} = 1 \quad j = 1,2,\ldots,n \tag{3}$$

$$\sum_{k=1}^{M}\sum_{j=0}^{n} X_{0jk} = \sum_{k=1}^{M}\sum_{i=0}^{n} X_{i0k} = M \tag{4}$$

$$\sum_{j=0}^{n} X_{ijk} = \sum_{j=0}^{n} X_{jik}\ ;\ i = 1,2,\ldots,n;\ k = 1,2,\ldots,M \tag{5}$$

$$\begin{aligned} &U_{ik} + U_{jk} + (n - m + 1) X_{ijk} \le (n - M); \\ &1 \le i, j \le n, i \ne j, k = 1,2,\ldots,M \end{aligned} \tag{6}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} X_{ijk} q_i \le Q;\ k = 1,2,\ldots,M \tag{7}$$

$$\sum_{i=1}^{n}\sum_{j=0}^{n} X_{ijk} t_{ij} \le \tau; k = 1,2,\ldots,M \tag{8}$$

Where: School buses are centrally located and have collect waiting students at n pick-up points and to drive them to school. The number of students that wait in pick-up point i is q_i, ($q_i > 0$, $i = 1, 2, \ldots, n$). The capacity of each bus is limited to Q students ($q_i \le Q$). The objective function to the School Bus Problem is composing of two costs: a) cost incurred by the number of buses used, b) driving cost (fuel, maintenance, drivers salary, and others), subject to operational constraints, Cost a or b have to be minimized.

Newton and Thomas [28] are pioneering in offering solution by considered restrictions of routing for a single school, one bus and 80 bus stops of the SBRP.

Angel et al. [1] propose to group the bus stops by a clustering algorithm of 1500 students and 5 schools in Indiana.

Bennett and Gazis [3] used a modified version of the Savings Algorithm [10] to solve an instance from Toms River, NJ with 256 bus stops and 30 routes.

Newton and Thomas [29] propose an algorithm to solution a problem for one district, by the location of each school, the location of each student, the time period for use of transport by student and the availability of buses.

Pickens and Tyler [32] propose a mathematical model of a school district network, and after defining stop locations, provide an initial set of near-optimal routes.

Bodin and Berman [4] solves an instance from Long Island, NY with 13000 students and 25 schools using a 3-OPT procedure to generate an initial traveling salesman tour with feasible routes.

Gavish and Shlifer [20] propose a Branch and Bound algorithm for solving a class of transportation scheduling problems such as: the Combining Truck Trip problem, the Delivery problem, the School Bus problem, the Assignment of Buses to Schedules, and the Travelling Salesman problem.

Dulac et al. [19] present a comprehensive study of the school bus routing problem in urban surroundings.

Hargroves and Demetsky [23] show a case study of a suburban-rural county in Virginia.

Lindenberg [26] develop a program for minimizing the total number of buses of a given bus fleet with different capacities needed for serving a school district.

Swersey and Ballard [39] gave a set of routes that takes all students from their bus stop to their schools from New Haven, CT with 30-38 buses and 100 routes.

Desrosiers et al. [15] solves the instance from Montréal, CA with 20000 students and 60 schools.

Chen et al. [8] proposes an expert system approach to routing and scheduling school buses for a rural school system.

Bookbinder and Edwards [5] developed a program scheduling problem with a set of student pick-ups and drop-offs for which school-to-school routes. Several problems from the Durham Board of Education are solved.

Chen et al. [9] generates routes to reduce the number of buses required and the fleet traveling distance by an expert system approach. It also allows planner participation in the process. Application to a real-world rural school district is discussed.

Thangiah and Nygard [40] use the GENROUTER system to route school buses for two school districts. The routes obtained by GENROUTER system were superior to those obtained by the CHOOSE school bus routing system and the current routes in use by the two school districts.

Graham [21] describes how the state of North Carolina has been implementing a network of transportation information management systems (TIMS) at the local school district level to improve the efficiency and cost-effectiveness of route scheduling for school buses. The program has contributed to a statewide reduction in mileage exceeding 2.6 million miles annually.

Chou [11] presents the design and applications of a decision support system developed for bus routing, route sequence mapping, and passenger geocoding. The system, built on a geographic information system (GIS), is suitable for municipal transit planning and school bussing.

Bowerman et al. [7] introduces a multi-objective approach to modeling the urban school bus routing problem and describes an algorithm for generating a solution to this problem. First groups students into clusters using a multi-objective districting algorithm and then generates a school bus route and the bus stops for each cluster using a combination of a set covering algorithm and a traveling salesman problem algorithm. Numerical results are reported using test data from Wellington County, Ontario.

Braca et al. [6] propose investigate various issues related to the development of a computerized system to help route and schedule school buses throughout the five boroughs of New York City. They analyze various aspects of the problem including the generation of routes in the Borough of Manhattan and provide a solution requiring far fewer buses than are currently in use. The computerized system, called CATS, incorporating many of the results obtained in this research, is currently being used to route Special Education students.

Pacheco et al. [30] proposes the use of meta-heuristics method based on neighborhood movements to solve the SBRP. Rhoulac et al. [33] show on-board mobile Global Positioning System equipment on School bus routing and scheduling in North Carolina using the transportation information management system (TIMS).

Li and Fu [25] describes a case study (test data from a kindergarten in Hong Kong) of the school bus routing problem, formulated a multi-objective combinatorial optimisation problem (minimizing the total number of buses required, the total travel time spent by pupils at all pick-up points, which is what the school and parents are concerned with most, and the total bus travel time. It also aims at balancing the loads and travel times between buses), and a heuristic algorithm for its solution.

Corberán et al. [12] address the problem of routing school buses in a rural area. They develop a solution procedure that considers each objective separately and search for a set of efficient solutions instead of a single optimum. The solution procedure is based on constructing, improving and then combining solutions within the framework of the evolutionary approach known as scatter search.

Spada et al. [38] propose a modeling framework where the focus is on optimizing the level of service for a given number of buses in the school bus routing and scheduling problem.

Ripplinger [34] show a rural routing heuristic of two parts: constructing the initial route and then improving it by using a fixed tenure Tabu search algorithm. The rural routing heuristic is applied to a randomly generated school district with rural characteristics.

Guo et al. [22] propose a new heuristic optimization algorithm for solving the school bus problem (to minimize the number of buses, to minimize total travel time of buses, to minimize total travel time spent by all children, to balance the loads among buses and to balance the travel time among buses).

Iskander et al. [24] propose an algorithm to solve a school district that has 27 schools, and showed significant improvement over existing schedules and schedules produced with a route first, cluster second algorithm. The algorithm clusters the stops, develops routes by solving modified traveling salesman problems, and assigns routes to buses to produce the final schedules.

Schittekat et al. [36] propose a solution by the GRASP method. Schittekat et al. [37] develop an integer programming formulation for small instances of the school bus routing problem.

Bektaş and Elmastaş [2] propose an exact solution approach for solving a real-life school bus routing problem (SBRP) for transporting the students of an elementary school throughout central Ankara, Turkey. The problem is modeled as a capacitated and distance constrained open vehicle routing problem and an associated integer linear program is presented.

Thangiah et al. [41] presents a heuristic to solve a complex rural school bus routing problem for student transportation in Pennsylvania using digitized road networks obtained from the U. S. Census Bureau that can lead to cost savings for both State and local governments. The rural school district between pickup points, depots and schools, consisting of 4200 road segments. Feasible solutions to the complex rural school bus routing problem consisting of 13 depots, 5 schools, 71 pickup points and 583 students.

Park and Kim [31] provide a comprehensive review of the school bus routing problem.

Díaz-Parra et al. [18] propose a set of test instances of the School Bus Routing Problem, called SBRPLIB (School Bus Routing Problem Library).

Ruiz-Vanoye and Díaz-Parra [43] show similarities between Meta-heuristics Algorithms and the Science of Life (examines the structure, function, growth, origin, evolution, distribution and classification of all living things): Meta-heuristics based on gene transfer, Meta-heuristics based on interactions among individual insects and Meta-heuristics based on biological aspects of alive beings. The Meta-heuristics based on gene transfer could be Vertical Transfer Algorithm or Genetic Algorithm (natural evolution of genes in an organic population or Vertical Gene Transfer) and Transgenic Algorithm (transfers of genetic material to another cell that is not descending or Horizontal Gene Transefer) [43].

The transfer of genetic material to offspring or the inheritance of genes by subsequent generations is an essential basis of the evolutionary process. The most common form of gene transfer for higher organisms is sexual reproduction.

In the case of higher plants, genetic information is passed along to the next generation by pollination; this is called Vertical Gene Transfer (VGT). Vertical Gene Transfer (VGT) occurs when an organism receives genetic material from its ancestor, e.g., from its parent or a species from which it evolved [44]. This mechanism was used in computer science by Holland [45] to define Genetic Algorithms (GA). GAs are heuristic adaptive that solve optimisation problems or NP-Complete problems by simulating the natural evolution of genes in an organic population.

This paper proposes a Vertical Transfer Algorithm or Genetic Algorithm to solve the School Bus Routing Problem with the characteristics of Single School, urban service, Problem scope (Morning), Mixed Load (not allowed), Special-educations students (not considered), Homogeneous fleet, and the Objectives (Number of buses used and Total Bus travel distance). The paper is organized as follows. Firstly, it describes Vertical Transfer Algorithm to solve the School Bus Routing Problem and afterwards, presents the results, discussion and conclusions.

Table 1. Related Works of SBRP

Research	Description
Newton and Thomas [28]	An algorithm to solution a problem for one district.
Pickens and Tyler [32]	A mathematically model of a school district network.
Bodin and Berman [4]	Solves an instance from Long Island, NY with 13000 students and 25 schools using a 3-OPT procedure.
Gavish and Shlifer [20]	A Branch and Bound algorithm for solving the School Bus problem.
Hargroves and Demetsky [23]	A case study of a suburban-rural county in Virginia.
Lindenberg [26]	A program for minimizing the total number of buses of a given bus fleet with different capacities needed for serving a school district.
Swersey and Ballard [39]	A set of routes that takes all students from their bus stop to their schools from New Haven, CT with 30-38 buses and 100 routes.
Desrosiers et al. [15]	Solves the instance from Montréal, CA with 20000 students and 60 schools.
Chen et al.[8]	An expert system approach to routing and scheduling school buses for a rural school system.
Bookbinder and Edwards [5]	Several problems from the Durham Board of Education are solved.
Chen et al. [9]	Generates routes to reduce the number of buses required and the fleet traveling distance by an expert system approach.
Thangiah and Nygard [40]	The GENROUTER system to route school buses for two school districts.
Graham [21]	Describes how the state of North Carolina has been implementing a network of transportation information management systems (TIMS) at the local school district level
Chou [11]	The design and applications of a decision support system developed for bus routing, route sequence mapping, and passenger geocoding.
Bowerman et al. [7]	A multi-objective approach to modeling the urban school bus routing problem and describes an algorithm for generating a solution to this problem.
Braca et al. [6]	The development of a computerized system to help route and schedule school buses throughout the five boroughs of New York City.
Pacheco [30]	Meta-heuristics method based on neighborhood movements to solve the SBRP.
Rhoulac et al.[33]	Show on-board mobile Global Positioning System equipment on School bus routing and scheduling in North Carolina using the transportation information management system (TIMS).
Li and Fu [25]	Describes a case study (test data from a kindergarten in Hong Kong) of the school bus routing problem
Spada et al. [38]	A modeling framework where the focus is on optimizing the level of service for a given number of buses in the school bus routing and scheduling problem.
Iskander et al. [24]	Propose an algorithm to solve a school district that has 27 schools, and showed significant improvement over existing schedules and schedules produced with a route first, cluster second algorithm.
Bektaş and Elmastaş [2]	Propose an exact solution approach for solving a real-life SBRP for transporting the students of an elementary school throughout central Ankara, Turkey.
Thangiah et al. [41]	Presents a heuristic to solve a complex rural SBRP for student transportation in Pennsylvania
Park and Kim [31]	Provide a comprehensive review of the school bus routing problem.
Díaz-Parra et al. [18]	Propose a set of test instances of the School Bus Routing Problem, called SBRPLIB (School Bus Routing Problem Library).
Kim and Soh [46]	They implement the simulation model through the template model in spread-sheet platform to find operational directions of school bus managing policy of a case for Wonkwang University
Park et al. [47]	A new mixed load improvement algorithm to solve a real world SBRP.

2 Vertical Transfer Algorithm for the School Bus Routing Problem

The vertical transfer algorithm for solving the School Bus Routing Problem is a genetic algorithm with a selection operator of tournament, crossover-k operator, and a mutation operator based on mutation-S operator. The Genetic Algorithm is considered as bio-inspired algorithm [35] because realizes the transfer of genetic material to offspring or the inheritance of genes by subsequent generations (called Vertical Gene Transfer or VGT).

In the Figure 1 is the Vertical Transfer Algorithm for solving the SBRP.

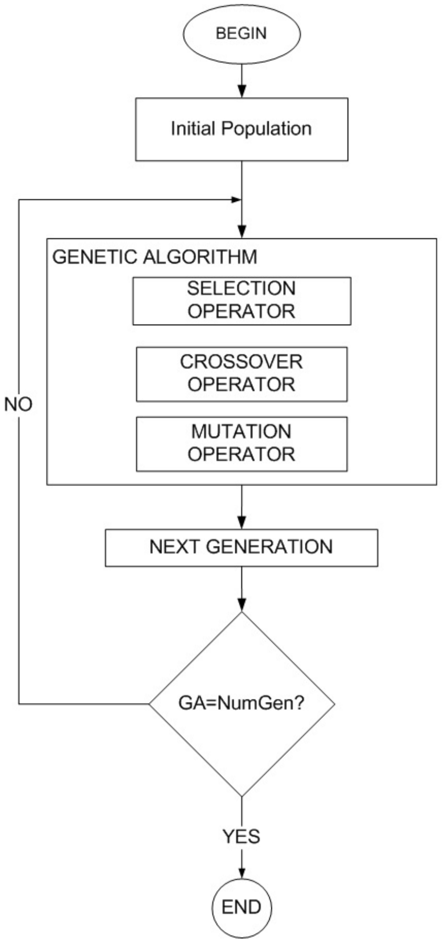


Fig. 1. Vertical Transfer Algorithm

The Code of the Vertical Transfer Algorithm or Genetic algorithm to solve SBRP is:

Input: Vehicle Capacity, Bus Stop, *X* coord., *Y* Coord., Student Number (Demand), Bus Stop time window (earliest pick-up time, Due pick-up time, Maximum Riding Time, Maximum Walking Time).

Output: Total Bus travel distance, Number of Buses and the routes.

0. Begin

1. Generate not-random initial population (one individual) of n chromosomes, suitable solution for the problem, by the use of the clusterisation population pre-selection operator.

2. New population. Create a new population by repeating following steps until the new population is complete:

a) Generate random population of n chromosomes based on the not-random initial population or the population of the new generation, interchanging of random way a pair of genes of the individuals.

b) Fitness. Evaluate the fitness $f(x)$ of each chromosome x in the population.

c) Selection operator. Select two parent chromosomes from a population according to their fitness (tournament operator).

d) Crossover operator. The crossover-k operator consists of finding two points randomly in the individual (*rand1* and *rand2*) and to search in the individual the corresponding genes to make the crossover.

e) Mutation operator. Mutation-S consists of detecting the genes with major distances in the matrix of Euclidean distances and to compare with the genes involved in the individual that minor generates distances to change, verifies the restriction of time and vehicle capacity.

3. Accepting. Place new offspring in a new population.

4. Replace. Use new generated population for a further run of algorithm.

5. Test. If the end condition is satisfied, stop, and return the best solution in current population.

6. Loop. Go to step 2.

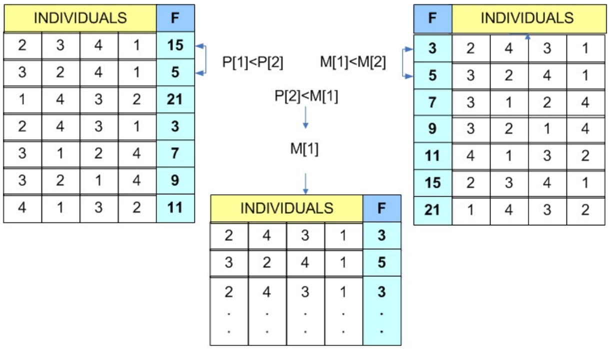
7. End

2.1 Initialisation Phase

Clusterisation population pre-selection operator [17] is used only for generating the initial population (an individual with n chromosomes, set of genes or vehicles) of an intelligent way for evolutionary and genetic algorithms; and for generating the new population (by repeating steps until the new population is complete, see next sections). The initial population and the new population are generating by a data-mining technique called by k-means algorithm [27], which classifies the groups of data with similar characteristics. We cluster the geographical location (*X* coord. and *Y* Coord.) of the School Bus Routing Problem. The initialization of the k-means algorithms was obtained of the characteristic instance called vehicles [17].

2.2 Selection Operator

The Tournament selection operator [13] consists of randomly taking two individuals from the population and to generate a random number r (between zero and one). If $r < k$, where k is a parameter, selects the best one of the individuals on the contrary selects worse, the both individual ones are given back to the initials so that they can be selected again. We use the Tournament Selection to take two individuals from the population of the School Bus Routing Problem. The figure 2 shows the use of the tournament selection operator on populations of individual of the School Bus Routing Problem.



INDIVIDUALS				F
2	3	4	1	15
3	2	4	1	5
1	4	3	2	21
2	4	3	1	3
3	1	2	4	7
3	2	1	4	9
4	1	3	2	11

F	INDIVIDUALS			
3	2	4	3	1
5	3	2	4	1
7	3	1	2	4
9	3	2	1	4
11	4	1	3	2
15	2	3	4	1
21	1	4	3	2

INDIVIDUALS				F
2	4	3	1	3
3	2	4	1	5
2	4	3	1	3
.	.	.	.	.
.	.	.	.	.

Fig. 2. Tournament Selection operator

2.3 Crossover Operator

The crossover-k operator [16] consists of finding two points randomly in the individual (rand1 and rand2) and to search in the individual (of the Vehicle Routing Problem with time windows) the corresponding genes to make the crossover. We use the crossover-k operator to find two points randomly in the individual and to search in the individual of the School Bus Routing Problem:

a) The crossover-k operator generates two individuals from two individuals.

b) The operator takes two individuals randomly from the population and generates two random numbers, and then the search of the nodes corresponding to the random numbers is realized in individual 1.

c) Once identified the nodes to cross, the operator search the nodes in the individual 2.

d) Since the positions of the corresponding nodes in both individual have been identified, the crossing is realized of nodes in individual 1 (the first individual), to generate the second individual, the crossover is realized in the individual 2.

e) The crossover is realized as long as the nodes to cross are different.

In the Figure 3 is the mechanism of operation of the crossover-k operator.

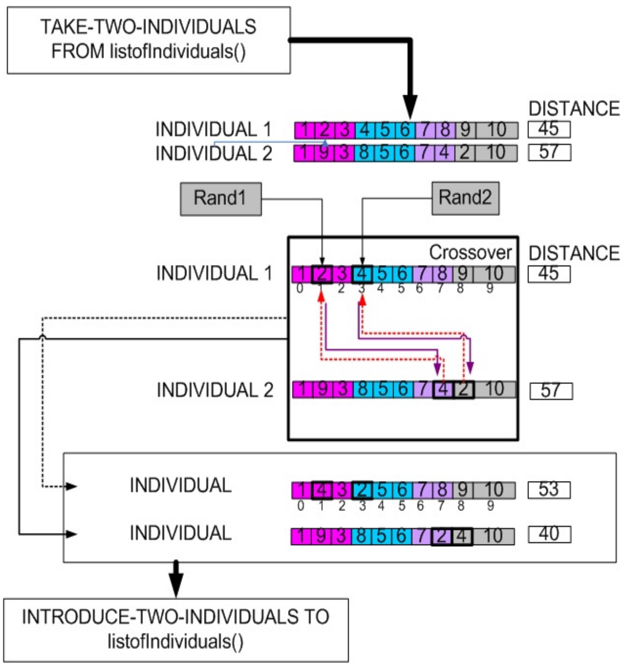
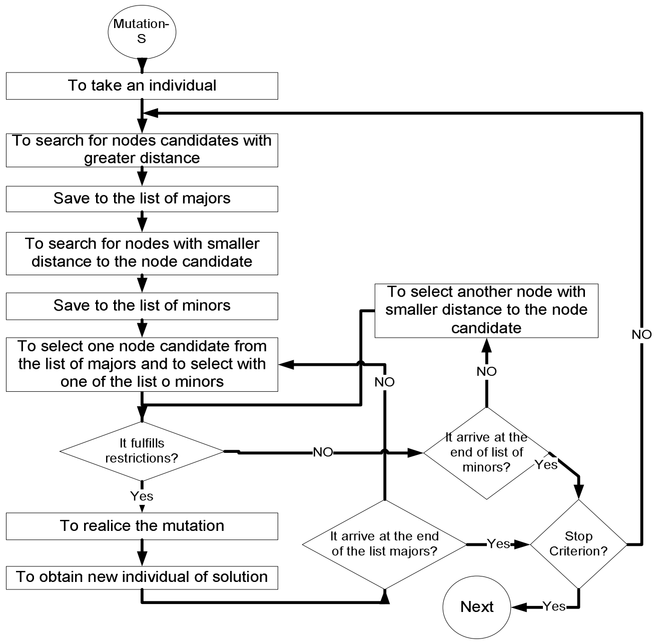


Fig. 3. Crossover-k operator

2.4 Mutation Operator

Díaz-Parra and Cruz-Chávez [16] propose a Genetic Algorithm to solve the Vehicle Routing Problem with Time Windows, using an intelligent mutation operator (mutation-S) for local search. We use the intelligent mutation operator for the School Bus Routing Problem.

Mutation-S consists of detecting what of all the genes that conform the individual is the one that involves major distances. On the basis of the gene of greater distance the one realizes a search in the matrix of Euclidean distances with each one of the genes involved in the individual that minor generates distances is the gene candidate to change, next to identify the gene candidate, verifies the reductions of area of time window and vehicle capacity, if the then restrictions are not violated comes to realize the mutation.



Mutation-S
To take an individual
To search for nodes candidates with greater distance
Save to the list of majors
To search for nodes with smaller distance to the node candidate
Save to the list of minors
To select one node candidate from the list of majors and to select with one of the list o minors
It fulfills restrictions?
Yes
To realice the mutation
To obtain new individual of solution
NO
It arrive at the end of list of minors?
NO
To select another node with smaller distance to the node candidate
Yes
NO
It arrive at the end of the list majors?
Yes
Stop Criterion?
NO
Yes
Next

Fig. 4. Crossover-k operator

The temporal complexity of the Vertical Transfer Algorithm for solves SBRP instances are in the equations 9 to 12.

$$\begin{aligned}T_{sec}(n)=c_1+\sum_{i=0}^{n}(c_2+\sum_{j=0}^{n}(c_3+\sum_{k=0}^{n}c_4)+\sum_{l=0}^{n_1}(c_5+\sum_{m=0}^{n}c_6)+\sum_{n=0}^{C_1}(c_7+\sum_{o=0}^{n}c_8+\\ \sum_{p=0}^{C_2}(c_9+\sum_{q=0}^{n}c_{10}+\sum_{r=0}^{n}c_{11}+\sum_{s=o}^{n}c_{12}+\sum_{t=0}^{n}c_{13}+\sum_{u=0}^{n}c_{14}+\sum_{v=0}^{n}c_{15})))=\end{aligned} \tag{9}$$

$$\begin{aligned}T_{sec}(n)=c_1+c_2n+c_3n^2+c_4n^3+c_5n^2+c_6n^3+\\ C_1c_7n+C_1c_8n^2+C_1C_2c_9n^2+C_1C_2c_{10}n^3+\\ C_1C_2c_{11}n^3+C_1C_2c_{12}n^3+C_1C_2c_{13}n^3+C_1C_2c_{14}n^3\\ +C_1C_2c_{15}n^3)))=\end{aligned} \tag{10}$$

$$\begin{aligned}T_{sec}(n)=c_1+c_2n+(c_3+c_5+c_8+C_1C_2c_9)n^2+(c_4+c_6+\\ C_1C_2(c_{10}+c_{11}+c_{12}+c_{13}+c_{14}+c_{15}))n^3=\end{aligned} \tag{11}$$

$$T_{sec}(n)=c_1+c_2n+c_{16}n^2+c_{17}n^3 \tag{12}$$

Where: *n* represents the size of the instance, *C* represents each operation of the algorithm, for example: *C1* represents the generation's number and *C2* represents the stop criteria of the neighborhood. The temporal complexity of the bio-inspired algorithm for solve SBRP in the worst of the cases is: $T(n) \in \mathrm{O}(n^3)$.

3 Experimentation and Results

The experimentation was carried on HP TouchSmart 300-1020 with AMD Athlon II 235e Dual-Core Processor and memory of 4 GB and the algorithm was development using Visual C++ v.6. The instances of SBRP were obtained from School Bus Routing Problem Library-SBRPLIB (a depository of test instances of the School Bus Routing Problem) [18]. We use the instance set S1 of the SBRPLIB. The instance set S1 contains 50 instances with Number of School: single, Surroundings of Service: Urban, Problem scope: Morning, Fleet mix: Homogeneous, vehicle capacity: 40, Bus Stop: 200. The urban means routes inside the city, morning means 7:00-9:00, homogeneous fleet is the vehicle capacity equal in all the fleet; and the instance set UAEM1, the instance set UAEM1 contains 1 instance with number of school: single, surrondings of service: Urban, Problem scope: Morning, fleet mix: Homogeneous, vehicle capacity: 40, Bus Stop: 21.

Experiments are repeated 30 times for each instance. In table 2 are the parameters or characterization of the SBRP instances.

Table 2. SBRP Instances

NS	SS	PS	ML	SEE	F	VN	VC
STWb	STWdue						
BS	XCO	YCO	SN	EPT	DPT	MRT	MWT
0	x_0	Y_0	SN_0	EPT_0	DPT_0	MRT_0	MWT_0
…	…	…	…	…	…	…	…
n	x_n	Y_n	SN_n	EPT_n	DPT_n	MRT_n	MWT_n

Where NS: Number of School (single or multiple), SS: Surroundings of Service (urban or rural), PS: Problem scope (morning, afternoon, both), ML: Mixed Load (allowed or no allowed), SEE: special-educations students (considered or not considered), F: Fleet mix (homogeneous fleet or heterogeneous fleet), VN: Vehicle Number, VC: Vehicle Capacity, STWb: School Time Windows Begin, SWTdue: School Time Windows Due, BS: Bus Stop, XCO: X Coord., YCO: Y Coord., SN: Student Number, MRT: Maximum Riding time, EPT: earliest pick-up time, DPT: Due pick-up time, MWT: Maximum Walking Time or distance.

In Table 3 are the results of the Vertical Transfer Algorithm. The input of the algorithm is an initial population from which realized a clustering to obtain a list of individuals (where I: instances, TB: Total Bus Travel Distance, NB: Number of Buses, GA: Genetic Algorithm). The k-Means algorithm for clustering (data-mining techniques), introduced by MacQueen [27], is one of the clustering algorithms more well-known which classifies the groups of data with similar characteristics; these groups know themselves like clusters. The parameters provided to the k-means procedure in the genetic algorithm are: XCO = X Coordinate and YCO = Y Coordinate obtained from the S1 instance set of the SBRPLIB [18].

Table 3. Results of the Vertical Transfer Algorithm

I	GA		I	GA	
	TB	NB		TB	NB
SBRP-S1-1	10185.19	35	SBRP-S1-26	11018.24	35
SBRP-S1-2	10965.97	36	SBRP-S1-27	11597.19	36
SBRP-S1-3	11266.22	36	SBRP-S1-28	11515.74	38
SBRP-S1-4	15214.01	35	SBRP-S1-29	11490.97	36
SBRP-S1-5	12851.75	36	SBRP-S1-30	11365.43	33
SBRP-S1-6	12582.34	36	SBRP-S1-31	11090.65	36
SBRP-S1-7	12494.22	36	SBRP-S1-32	12272.99	35
SBRP-S1-8	9818.49	36	SBRP-S1-33	11890.97	36
SBRP-S1-9	9818.49	36	SBRP-S1-34	13112.94	38
SBRP-S1-10	11280.01	36	SBRP-S1-35	11491.31	37
SBRP-S1-11	10279.03	34	SBRP-S1-36	10138.13	31
SBRP-S1-12	11428.98	33	SBRP-S1-37	10108.62	33
SBRP-S1-13	14031.38	35	SBRP-S1-38	10006.01	36
SBRP-S1-14	10035.33	34	SBRP-S1-39	11744.12	37
SBRP-S1-15	10216.76	37	SBRP-S1-40	11200.10	34
SBRP-S1-16	11165.36	36	SBRP-S1-41	11102.05	36
SBRP-S1-17	10302.56	38	SBRP-S1-42	11381.58	36
SBRP-S1-18	11005.69	34	SBRP-S1-43	12074.00	34
SBRP-S1-19	10558.33	37	SBRP-S1-44	12513.76	36
SBRP-S1-20	11542.48	36	SBRP-S1-45	11537.06	34
SBRP-S1-21	10958.58	36	SBRP-S1-46	11565.18	36
SBRP-S1-23	12692.01	34	SBRP-S1-47	11222.52	35
SBRP-S1-24	11521.35	34	SBRP-S1-48	11080.48	35
SBRP-S1-25	10025.53	37	SBRP-S1-49	10903.73	33
SBRP-S1-26	11018.24	35	SBRP-S1-50	10342.50	33

In the Figure 5 is the solution of the instance SBRP-S1-1.sbrp.

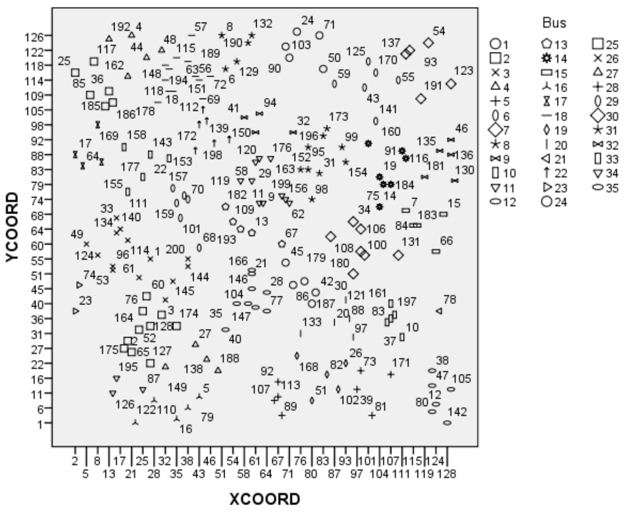


Fig. 5. Instance SBRP-S1-1

In Table 4 are the results of the instance SBRP-S1-1 with the Vertical Transfer Algorithm.

Table 4. Results of Instance SBRP-S1-1

Total Bus Travel Distance	Number of Buses
10185.198323	35
10198.961481	35
10225.412947	35
10298.665642	35
10382.977067	35
10533.203659	36
10573.120426	36
10674.057590	36
10747.046257	36
10818.807889	36

We use the instance set UAEM-1 corresponding with one school (Autonomous University of Morelos State or UAEM) of the Cuernavaca city, Surroundings of Service: Urban, Problem scope: Morning, Fleet mix: Homogeneous, vehicle capacity: 40, Bus Stop: 22. The urban means routes inside the city, morning means 7:00-9:00, homogeneous fleet is the vehicle capacity equal in all the fleet. Autonomous University of Morelos State is a public institution of higher education larger, coverage, and more important in the state Mexico's Morelos, with headquarters in the capital, the city of Cuernavaca. The institution has 27 academic units, 5 centers and 2 research units located in 3 campuses campus installed in various municipalities in the state. A survey (30 surveys peer faculty or 660 surveys) to the students of the 22 university faculties in order to obtain real information of university transport and analyzing their behavior in order to make a proposal to transport college students benefit from the UAEM. The representative sample taken from the student population is comprised of 30% of students on north campus (Chamilpa) of the UAEM, the university has a total of 11,504 senior students spread over three campuses, North Campus, east campus and south campus. The figure 7 contains the bus stations necessary to transport some students of the University.

Fig. 6. University Transport of UAEM

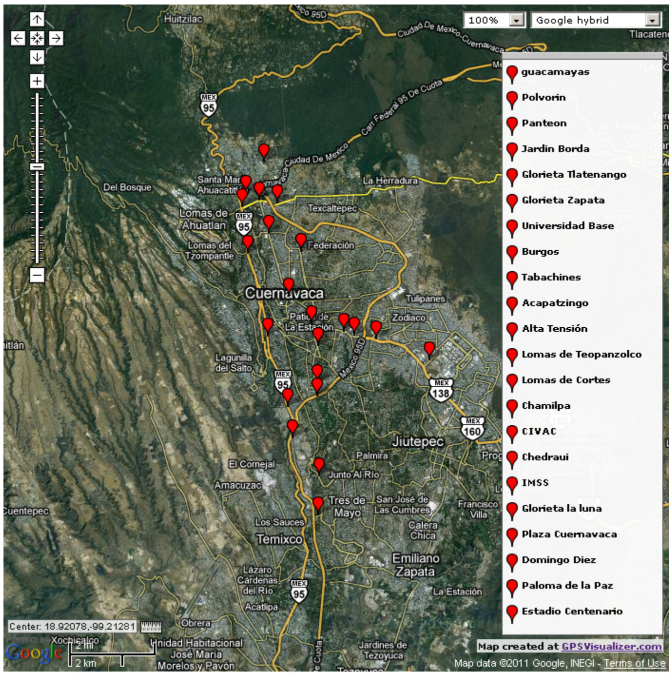


Fig. 7. Cuernavaca and Bus Stop locations of UAEM

In the Figures 8 and 9 are the solutions of the instance UAEM-1.sbrp for 3 and 4 vehicles.

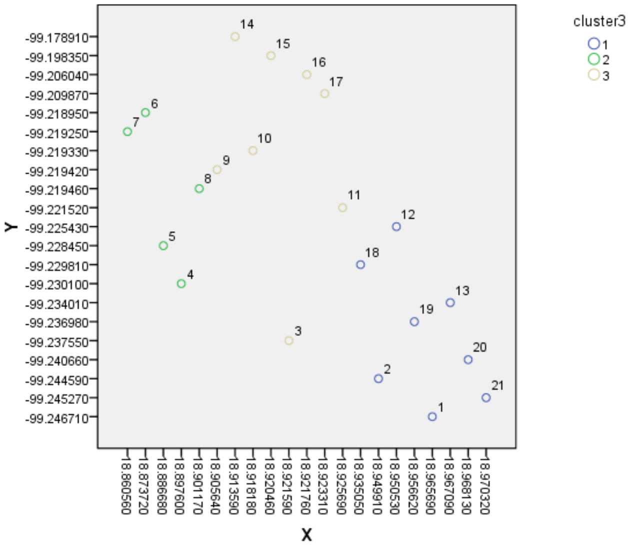


Fig. 8. Instance UAEM-1 with K = 3

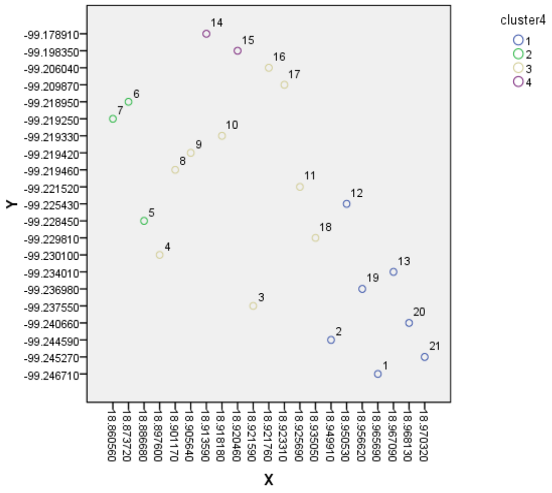


Fig. 9. Instance UAEM-1 with K = 4

In Table 5 are the total bus travel distance of the vertical transfer algorithm used to solve the SBRP-UAEM1, where the parameters are Individuals: 1000, Generations: 20, and Neighbors: 10. And in the table 6 is the best route of UAEM-1.

Table 5. Results of Instance UAEM-1

Total Bus Travel Distance	**Number of Buses**
1717.639358	3
1818.539493	3
1818.556720	3
1919.586404	3
1919.631402	3
1919.498600	4
1919.6325360	4
1919.6925580	4
2020.549677	4
2020.618819	4
2020.782089	4
2020.898991	4
2741.742000	4
2020.898991	4
2741.742000	4

Table 6. Best Route for UAEM-1 Instance

Vehicles	Routes
1	Guacamayas, Polvorin, Panteón, Jardín Borda, Glorieta Tlaltenango, Glorieta Zapata, Universidad Base.
2	Burgos, Tabachines, Acapatzingo, Alta Tensión, Lomas de Teopanzolco, Lomas de Cortes, Chamilpa, Universidad Base.
3	CIVAC, Chedraui, IMSS, Glorieta la Luna, Plaza Cuernavaca, Domingo Diez, Paloma de la Paz, Estadio Centenario, Universidad Base.

4 Conclusions

School bus scheduling is important because it reduces costs to the universities or schools and brings added value to the students to have a quality transport. The algorithm contained in this article provides solutions to set of instances S1 and UAEM-1 of SBRPLIB and is the first algorithm that solves this type of set of instances. Future work will use a new bio-inspired algorithm in aspects of biology to solve transportation problems.

Acknowledgements. This work was supported by SEP-PROMEP (Mexico) through grant PROMEP/103.5/12/8060 and the Autonomous University of Carmen.

References

[1] Angel, R.D., Caudle, W.L., Noonan, R., Whinston, A.: Computer-Assisted School Bus Scheduling. Management Science B 18, 279–288 (1972)

[2] Bektaş, T., Elmastaş, S.: Solving school bus routing problems through integer programming. Journal of the Operational Research Society 58(12), 1599–1604 (2007)

[3] Bennet, B., Gazis, D.: School Bus Routing by computer. Transportation Research 6, 317–326 (1972)

[4] Bodin, L.D., Berman, L.: Routing and scheduling of school buses by computer. Transportation Science 13(2), 113–129 (1979)

[5] Bookbinder, H.J., Edwards, H.S.: School-bus routing for program scheduling. Computers and Operations Research 17(1), 79–94 (1990)

[6] Braca, J., Bramel, J., Posner, B., Simchi-Levi, D.: Computerized approach to the New York City school bus routing problem. IIE Transactions 29(8), 693–702 (1997)

[7] Bowerman, R., Hall, B., Calamai, P.: A multi-objective optimization approach to urban school bus routing: Formulation and solution method. Transportation Research Part A 29(2), 107–123 (1995)

[8] Chen, D.-S., Kallsen, H.A., Snider, R.C.: School bus routing and scheduling: An expert system approach. Computers and Industrial Engineering 15(1-4), 179–183 (1988)

[9] Chen, D.-S., Kallsen, H.A., Chen, H.-C., Tseng, V.-C.: A bus routing system for rural school districts. Computers and Industrial Engineering 19(1-4), 322–325 (1990)

[10] Clarke, G., Wright, J.W.: Scheduling of Vehicles from a Central Depot to a number of delivery points. Operations Research 12, 568–581 (1964)

[11] Chou, Y.-H.: Automatic bus routing and passenger geocoding with a geographic information system. In: Proceedings of the 6th Conference of Vehicle Navigation and Information Systems (VNIS), pp. 352–359 (1995)
[12] Corberán, A., Fernández, E., Laguna, M., Martí, R.: Heuristic solutions to the problem of routing school buses with multiple objectives. Journal of the Operational Research Society 53(4), 427–435 (2002)
[13] Wetzel, A.: Evaluation of the effectiveness of genetic algorithms in combinational optimization. University of Pittsburgh, Pittsburgh (1983) (unpublished)
[14] Desrosiers, J., Ferland, J., Rousseau, J.-M., Lapalme, G., Chapleau, L.: An overview of a school busing system. In: International Conference on Transportation, Scientific Management of Transport Systems, New Delhi, vol. IX, pp. 235–243 (1981)
[15] Desrosiers, J., Ferland, J.A., Rousseau, J.-M., Lapalme, G., Chapleau, L.: TRANSCOL: A multi-period school bus routing and scheduling systems. Management Sciences 22, 47–71 (1986)
[16] Díaz-Parra, O., Cruz-Chávez, M.A.: Evolutionary Algorithm with Intelligent Mutation Operator that solves the Vehicle Routing Problem of Clustered Classification with Time Windows. Polish Journal of Environmental Studies 17(4C), 91–95 (2008)
[17] Díaz-Parra, O., Ruiz-Vanoye, J.A., Zavala-Díaz, J.C.: Population pre-selection operators used for generating a non-random initial population to solve vehicle routing problem with time windows. Scientific Research and Essays 5(22), 3529–3537 (2010)
[18] Díaz-Parra, O., Ruiz-Vanoye, J.A., Zavala-Díaz, J.C.: School Bus Routing Problem Library-SBRPLIB. International Journal of Combinatorial Optimization Problems and Informatics 2(1), 23–26 (2011)
[19] Dulac, G., Ferland, J.A., Forgues, P.A.: School bus routes generator in urban surroundings. Computers and Operations Research 7(3), 199–213 (1980)
[20] Gavish, B., Shlifer, E.: An approach for solving a class of transportation scheduling problems. European Journal of Operational Research 3(2), 122–134 (1979)
[21] Graham, D.S.: A GIS for bus routing saves money, worry in North Carolina. Geo. Info. Systems 3(5), 39–43 (1993)
[22] Guo, Q., Li, L., Guo, Y.: Routing optimization for school bus problem. Journal of Southwest Jiaotong University 41(4), 486–490 (2006)
[23] Hargroves, B.T., Demetsky, M.J.: A computer assisted school bus routing strategy: A case study. Socio-Economic Planning Sciences 15(6), 341–345 (1981)
[24] Iskander, W., Jaraiedi, M., Emami, F.: A practical approach for school bus routing and scheduling. In: IIE Annual Conference and Exposition, Orlando, FL (2006)
[25] Li, L., Fu, Z.: The school bus routing problem: A case study. Journal of the Operational Research Society 53(5), 552–558 (2002)
[26] Lindenberg, W.: EDP Program System for Minimizing the Costs of School Bus Traffic. Angewandte Informatik, Applied Informatics 25(4), 158–165 (1983)
[27] MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Symposium on Math, Statistics, and Probability, Berkeley, CA, vol. 1, pp. 281–297. University of California Press (1967)
[28] Newton, R.M., Thomas, W.H.: Design of school Bus Routes by computer. Socio-Economic Planning Science 3(1), 75–85 (1969)
[29] Newton, R.M., Thomas, W.H.: Bus routing in a multi-school system. Computers & Operations Research 1(2), 213–222 (1974)
[30] Pacheco Bonrostro, J.A., Aragón, A., Delgado Quiralte, C.: Design of algorithms for the problem of the school transport. Application in the Burgos province. Questiió: Quaderns d'Estadística, Sistemes, Informatica i Investigació Operativa 24(1), 55–82 (2000)

[31] Park, J., Kim, B.-I.: The school bus routing problem: A review. European Journal of Operational Research 202(2), 311–319 (2010)
[32] Pickens, P.R., Tyler, J.M.: Transportation planning for urban school systems. ASCE Transp. Eng. J. 100(TE2), 461–473 (1974)
[33] Rhoulac, T.D., Rouphail, N., Tsai, J.C.: Using global positioning system to improve school bus routing and scheduling. Transportation Research Record 1768, 242–249 (2001)
[34] Ripplinger, D.: Rural school vehicle routing problem. Transportation Research Record 1922, 105–110 (2005)
[35] Ruiz-Vanoye, J.A., Díaz-Parra, O.: Similarities between Meta-heuristics Algorithms and the Science of Life. Central European Journal of Operations Research 19(4), 445–466 (2010)
[36] Schittekat, P., Sevaux, M., Sörense, K., Springael, J.: A metaheuristic for the School Bus Routing Problem. In: 22nd European Conference on Operational Research EURO XXII (2007)
[37] Schittekat, P., Sevaux, M., Sörensen, K.: A mathematical formulation for a school bus routing problem. In: Proceedings of the International Conference on Service Systems and Service Management (ICSSSM), vol. 2, pp. 1552–1557 (2007b)
[38] Spada, M., Bierlaire, M., Leibling, T.: Decision-aid methodology for the school bus routing and scheduling problem. In: Proceedings of the 3rd Swiss Transport Research Conference, Monte Verita-Ascona, pp. 19–21 (2003)
[39] Swersey, A.J., Ballard, W.: Scheduling School Buses. Management Science 30(7), 844–853 (1984)
[40] Thangiah, S.R., Nygard, K.E.: School bus routing using genetic algorithms. In: Proceedings of SPIE - the International Society for Optical Engineering, vol. 1707, pp. 387–398 (1992)
[41] Thangiah, S.R., Fergany, A., Wilson, B., Pitluga, A., Mennell, W.: School bus routing in rural school districts. Lecture Notes in Economics and Mathematical Systems 600, 209–232 (2008)
[42] Díaz-Parra, O., Ruiz-Vanoye, J.A., Buenabad-Arias, Á., Cocón, F.: A Vertical Transfer Algorithm for School Bus Routing Problem. In: Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC 2012), Mexico City, November 5-9, pp. 66–71 (2012)
[43] Ruiz-Vanoye, J.A., Díaz-Parra, O.: Similarities between Meta-heuristics Algorithms and the Science of Life. Central European Journal of Operations Research 19(4), 445–466 (2011)
[44] Lederberg, J., Tatum, E.L.: Novel genotypes in mixed cultures of biochemical mutants of bacteria. Cold Spring Harbor Symposia of Quantitative Biology 11 (1946)
[45] Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
[46] Kim, J.H., Soh, S.: Designing hub-and-spoke school bus transportation network: a case study of wonkwang university. Promet-Traffic & Transportation 24(5), 389–394 (2012)
[47] Park, J., Tae, H., Kim, B.I.: A post-improvement procedure for the mixed load school bus routing problem. European Journal of Operational Research 217(1), 204–213 (2012)

An Efficient Craziness Based Particle Swarm Optimization Technique for Optimal IIR Filter Design

S.K. Saha[1], R. Kar[1], D. Mandal[1], and S.P. Ghoshal[2]

[1] Department of Electronics and Communication Engineering
[2] Department of Electrical Engineering,
National Institute of Technology, Durgapur, West Bengal, India
{namus.ahas,rajibkarece,durbadal.bittu,
spghoshalnitdgp}@gmail.com

Abstract. In this paper an improved version of Particle Swarm Optimization (PSO) called Craziness based PSO (CRPSO) is considered as an efficient optimization tool for designing digital Infinite Impulse Response (IIR) filters. Apart from gaining better control on cognitive and social components of conventional PSO, the CRPSO dictates better performance due to incorporation of craziness parameter in the velocity equation of PSO. This modification in the velocity equation not only ensures the faster searching in the multidimensional search space but also the solution produced is very close to the global optimal solution. The effectiveness of this algorithm is justified with a comparative study of some well established algorithms, namely, Real coded Genetic Algorithm (RGA) and conventional Particle Swarm Optimization (PSO) with a superior CRPSO based outcome for the designed 8th order IIR low pass (LP), high pass (HP), band pass (BP) and band stop (BS) filters. Simulation results affirm that the proposed CRPSO algorithm outperforms its counterparts not only in terms of quality output, i.e., sharpness at cut-off, pass band ripple and stop band attenuation but also in convergence speed with assured stability.

1 Introduction

Signal carries information, but this information is getting contaminated with noise which is picked up mostly by electro magnetic means. So, at the receiving end to extract the information signal processing is executed on noise corrupted signal. Depending on nature of signal and point of application signal processing may be analog, digital or mixed in practice. Application of digital signal processing (DSP) has increased many folds as the production of DSP in bulk is easier as the basic operation is confined into mainly addition, multiplication and recalling of previous data. In digital filter design minimum number of discrete components are required that immunes the performance of designed filter from thermal drift.

Digital filters are broadly classified into two main categories namely; finite impulse response (FIR) filter and infinite impulse response (IIR) filter [1-2]. The output of FIR filter depends on present and past values of input, so the name non-recursive is aptly suited to this filter. On the other hand, the output of IIR filter depends not only on previous inputs, but also on previous outputs with impulse responses continuing forever in time at least theoretically, so the name 'recursive' is aptly suited to this filter; anyway, a large memory is required to store the previous outputs for the recursive IIR filter.

Hence, due to these aspects FIR filter realization is easier with the requirement of less memory space and design complexity. Ensured stability and linear phase response over a wide frequency range are the additional advantages. On the other hand, IIR filter distinctly meets the supplied specifications of sharp transition width, lower pass band ripple and higher stop band attenuation with ensured lower order compared to FIR filter. As a consequence, a properly designed IIR filter can meet the magnitude response close to ideal and more finely as compared to a FIR filter. Due to these challenging features with wide field of applications, performances of IIR filters designed with various optimization algorithms are compared to find out the effectiveness of algorithms and the best optimal IIR filter with assured stability.

In the conventional design approach, IIR filters of various types (Butterworth, Chebyshev and Elliptic etc.) can be implemented with two methods. In the first case, frequency sampling technique is adopted for Least Square Error [3] and Remez Exchange [4] process. In the second method, filter coefficients and minimum order are calculated for a prototype low pass filter in analog domain which is then transformed to digital domain with bilinear transformation. This frequency mapping works well at low frequency, but in high frequency domain this method is liable to frequency warping [5].

IIR filter design is a highly challenging optimization problem. So far, gradient based classical algorithms such as steepest descent and quasi Newton algorithms have been aptly used for the design of IIR filters [6-7]. In general, these algorithms are very fast and efficient to obtain the optimum solution of the objective function for a unimodal problem. But the error surface (typically the mean square error between the desired response and estimated filter output) of IIR filter is multimodal and hence superior evolutionary optimization techniques are required to find out better near global solution.

The shortfalls of classical optimization techniques for handling the multimodal optimization problem are as follows:

- Requirement of continuous and differentiable error fitness function (cost or objective function),
- Usually converges to the local optimum solution or revisits the same sub-optimal solution,
- Incapable to search the large problem space,
- Requirement of the piecewise linear cost approximation (linear programming),

Highly sensitive to starting points when the number of solution variables is increased and as a result the solution space is also increased.

So, it can be concluded that classical search techniques are only suitable for handling differentiable unimodal objective function with constricted search space. But the error surface of IIR filter is usually multimodal and non-differentiable. So the various evolutionary heuristic search algorithms are applied for filter optimization problems, which are as follows: Genetic Algorithm (GA) is developed with the inspiration of the Darwin's "Survival of the Fittest" strategy [8-9]; Simulated Annealing (SA) is designed from the thermodynamic effects [10]; Artificial Immune Systems (AIS) mimics the biological immune systems [11]; Ant Colony Optimization (ACO) simulates the ants' food searching behaviour [12]; Bee Colony Optimization mimics the honey collecting behaviour of the bee swarm [13]; Cats Swarm Optimization(CSO) is based upon the behaviour of cats for tracing and seeking of an object [14]; and PSO and its variants simulate the behaviour of bird flocking or fish schooling [15-21].

Ecology based Predator-prey model as an evolutionary optimization technique is discussed in [22], where each prey is considered as a possible solution in search space which is chased by a predator in predefined region; Searching behaviour of human being is mimicked for the development of Seeker Optimization Algorithm (SOA) [23]; In Bacteria Foraging Optimization (BFO) technique food searching behaviour of E. Coli bacteria is mimicked [24].

Naturally, it is a vast area of research continuously being explored. In this paper, the capability of global searching and near optimum result finding features of GA, PSO and CRPSO are investigated thoroughly for solving 8th order IIR filter design problems. GA is a probabilistic heuristic search optimization technique developed by Holland [25]. The features such as multi-objective, coded variable and natural selection made this technique distinct and suitable for finding the near global solution of filter coefficients.

Particle Swarm Optimization (PSO) is swarm intelligence based algorithm developed by Eberhart *et al.* [26-27]. Several attempts have been taken to design digital filter with basic PSO and its modified versions [15-21], [28-29]. The main attraction of PSO is its simplicity in computation and a few steps are required in the algorithm.

The limitations of the conventional PSO are premature convergence and stagnation problem [30-31]. To overcome these problems an improved version of PSO called CRPSO is suggested by the authors for the design of 8th order digital IIR low pass (LP), high pass (HP), band pass (BP) and band stop (BS) filters.

The paper is organized as follows: Basic structure of IIR filter along with the error fitness function is described in section 2. Different evolutionary algorithms namely, RGA, PSO and CRPSO are discussed in section 3. In section 4, comprehensive and demonstrative sets of data and illustrations are analyzed to make a floor of comparative study of performances among different algorithms. Finally, section 5 concludes the paper.

2 IIR Filter Design Formulation

This section discusses the design strategy of IIR filter based on all concerned algorithms. The input-output relation is governed by the following difference equation [2].

$$y(p)+\sum_{k=1}^{n} a_k y(p-k)=\sum_{k=0}^{m} b_k x(p-k) \tag{1}$$

where $x(p)$ and $y(p)$ are the filter's input and output, respectively, and $n(\geq m)$ is the filter's order. With the assumption of coefficient, $a_0 = 1$ the transfer function of the IIR filter is expressed as:

$$H(z)=\frac{\sum_{k=0}^{m} b_k z^{-k}}{1+\sum_{k=1}^{n} a_k z^{-k}} \tag{2}$$

Let $z = e^{j\Omega}$. Then, the frequency response of the IIR filter becomes

$$H(\Omega)=\frac{\sum_{k=0}^{m} b_k e^{-jk\Omega}}{1+\sum_{k=1}^{n} a_k e^{-jk\Omega}} \tag{3}$$

$$H(\Omega)=\frac{Y(\Omega)}{X(\Omega)}=\frac{b_0+b_1e^{-j\Omega}+b_2e^{-j2\Omega}+...+b_me^{-jm\Omega}}{1+a_1e^{-j\Omega}+a_2e^{-j2\Omega}+...+b_ne^{-jn\Omega}} \tag{4}$$

where $\Omega = 2\pi\left(\frac{f}{f_s}\right)$ in [0, π] is the digital frequency; f is the analog frequency and f_s is the sampling frequency. Different fitness functions are used for IIR filter optimization problems [32-34]. The commonly used approach to IIR filter design is to represent the problem as an optimization problem with the mean square error (MSE) as the error fitness function [34] expressed in (5).

$$J(\omega)=\frac{1}{N_s}[(d(p)-y(p))^2] \tag{5}$$

where N_s is the number of samples used for the computation of the error fitness function; $d(p)$ and $y(p)$ are the filter's desired and actual responses, respectively. The difference $e(p)=d(p)-y(p)$ is the error between the desired and the actual filter responses. The design goal is to minimize the MSE $J(\omega)$ with proper adjustment of coefficient vector ω represented as:

$$\omega = [a_0 a_1 \ldots a_n b_0 b_1 \ldots b_m]^T. \tag{6}$$

In this paper, a novel error fitness function given in (7) is adopted in order to achieve higher stop band attenuation and to have moderate control on the transition width. Using (7), it is found that the proposed filter design approach results in considerable improvement in stop band attenuation over other optimization techniques.

$$J_1(\omega) = \sum_{\omega} abs\left[abs\left(\left|H_d(\omega)\right| - 1\right) - \delta_p\right] + \sum_{\omega}\left[abs\left(\left|H_d(w)\right| - \delta_s\right)\right] \tag{7}$$

For the first term of (7), $\omega \in$ pass band including a portion of the transition band and for the second term of (7), $\omega \in$ stop band including the rest portion of the transition band. The portions of the transition band chosen depend on pass band edge and stop band edge frequencies.

The error fitness function given in (7) represents the generalized fitness function to be minimized using the evolutionary algorithms RGA, conventional PSO and the proposed CRPSO individually. Each algorithm tries to minimize this error fitness J_1 and thus optimizes the filter performance. Unlike other error fitness functions as given in [32-34] which consider only the maximum errors, J_1 involves summation of all absolute errors for the whole frequency band, and hence, minimization of J_1 yields much higher stop band attenuation and lesser pass band ripples.

3 Evolutionary Algorithms Employed

3.1 Real Coded Genetic Algorithm (RGA)

Standard Genetic Algorithm (also known as real coded GA) is mainly a probabilistic search technique, based on the principles of natural selection and evolution built upon the Darwin's "Survival of the Fittest" strategy [25]. Each encoded chromosome that constitutes the population is a solution to the filter designing optimization problem. These solutions may be good or bad, but are tested rigorously through the genetic operations such as crossover and mutation to evolve a global optimal or near global optimal solution of the problem at hand. Chromosomes are constructed over some particular alphabet {0, 1}, so that chromosomes' values are uniquely mapped onto the real decision variable domain. Each chromosome is evaluated by a function known as fitness function, which is usually the fitness function or objective function of the corresponding optimization problem. Each chromosome has a probability of selection and has to take part in the genetic operation based upon the Roulette's wheel strategy. In the genetic operations, crossover and mutation bring the variation in alleles of gene in the chromosome population along with the alleviation of trapping to local optimal solution.

Steps of RGA as implemented for the optimization of coefficient vector ω are as follows [35-36]:

Step 1: Initialize the real coded chromosome strings (ω) of n_p = 120 population, each consisting of equal number of numerator and denominator filter coefficients b_k and a_k , respectively; total coefficients = (n+1)*2 for nth order filter to be designed; minimum and maximum values of filter coefficients, hmin = -2, hmax = 2; number of samples=128; $\delta_p = 0.001$, $\delta_s = 0.0001$; maximum iteration cycles= 400, n=8.

Step 2: Decoding of the strings and evaluation of error fitness $J_1(\omega)$ according to (7).

Step 3: Selection of elite strings in order of increasing error fitness values from the minimum value.

Step 4: Copying the elite strings over the non selected strings.

Step 5: Crossover and mutation generate offspring.

Step 6: Genetic cycle updating.

Step 7: The iteration stops when maximum number of cycles is reached. The grand minimum error and its corresponding chromosome string or the desired solution having (n+1)*2 number of coefficients are finally obtained.

3.2 Particle Swarm Optimization (PSO)

PSO is flexible, robust, population based stochastic search algorithm with attractive features of simplicity in implementation and ability to quickly converge to a reasonably good solution. Additionally, it has the capability to handle larger search space and non-differential objective function, unlike traditional optimization methods. Eberhart *et al.* [26-27] developed PSO algorithm to simulate random movements of bird flocking or fish schooling.

The algorithm starts with the random initialization of a swarm of individuals, which are known as particles within the multidimensional problem search space, in which each particle tries to move toward the optimum solution, where next movement is influenced by the previously acquired knowledge of particle best and global best positions once achieved by individual and the entire swarm, respectively. The features incorporated within this simulation are velocity matching of individuals with the nearest neighbour, elimination of ancillary variables and inclusion of multidimensional search and acceleration by distance. Instead of the presence of direct recombination operators, acceleration and position modification supplement the recombination process in PSO. Due to the aforementioned advantages and simplicity, PSO has been applied to different fields of practical optimization problems.

To some extent, IIR filter design with PSO is already reported in [15-21], [28-29]. A brief idea about the algorithm for a D-dimensional search space with n_p particles that constitutes the flock is presented here. Each i^{th} particle is described by a position vector as $S_i = (s_{i1}, s_{i2}, ..., s_{iD})^T$ and velocity is expressed by $V_i = (v_{i1}, v_{i2}, ..., v_{iD})^T$.

The best position that the i^{th} particle has reached previously *pbest*$_i$ = $(p_{i1}, p_{i2}, ..., p_{iD})^T$, and group best is expressed as *gbest* = $(p_{g1}, p_{g2}, ..., p_{gD})^T$.

The maximum and minimum velocities are $V_{\max}$, $V_{\min}$, respectively.

$$V_{\max} = (v_{\max 1}, v_{\max 2}, ..., v_{\max D})^T \text{ and } V_{\min} = (v_{\min 1}, v_{\min 2}, ..., v_{\min D})^T .$$

The positive constants C_1, C_2 are related with accelerations and $rand_1, rand_2$ lie in the range [0, 1]. The inertia weight w is a constant chosen carefully to obtain fast convergence to optimum result. k denotes the iteration number.

The basic steps of the PSO algorithm are as follows [19-21]:

Step1: Initialize the real coded particles (ω) of n_p = 25 population, each consisting of equal number of numerator and denominator filter coefficients b_k and a_k, respectively; total coefficients D = (n+1)*2 for equal number numerator and denominator coefficients with nth order filter to be designed; minimum and maximum values of filter coefficients, hmin = -2, hmax = 2; number of samples=128; $\delta_p = 0.001$, $\delta_s = 0.0001$; maximum iteration cycles= 100 ; n= 8.

Step 2: Compute the error fitness value for the current position S_i of each particle

Step 3: Each particle can remember its best position $(pbest)$ which is known as cognitive information and that would be updated with each iteration.

Step 4: Each particle can also remember the best position the swarm has ever attained $(gbest)$ and is called social information and would be updated in each iteration.

Step 5: Velocity and position of each particle are modified according to (8) and (9), respectively [26].

$$V_i^{(k+1)} = w * V_i^{(k)} + C_1 * rand_1 * \{pbest_i^{(k)} - S_i^{(k)}\} + C_2 * rand_2 * \{gbest_i^{(k)} - S_i^{(k)}\} \quad (8)$$

where
$$\begin{aligned} V_i &= V_{\max} \; for V_i > V_{\max} \\ &= V_{\min} \; for \; V_i < V_{\min} \end{aligned}$$

$$S_i^{(k+1)} = S_i^{(k)} + V_i^{(k+1)} \quad (9)$$

Step 6: The iteration stops when maximum number of cycles is reached. The grand minimum error fitness and its corresponding particle or the desired solution having (n+1)*2 number of coefficients are finally obtained.

3.3 Craziness Based Particle Swarm Optimization (CRPSO) Technique

The global search ability of above discussed conventional PSO is improved with the help of the following modifications. This modified PSO is termed as craziness based particle swarm optimization (CRPSO).

The velocity in this case can be expressed as follows [37]:

$$\begin{aligned} V_i^{(k+1)} &= r_2 * sign(r_3) * V_i^k \\ &+ (1-r_2) * C_1 * r_1 * \left\{ pbest_i^{(k)} - S_i^{(k)} \right\} + (1-r_2) * C_2 * (1-r_1) * \left\{ gbest^{(k)} - S_i^{(k)} \right\} \end{aligned} \tag{10}$$

where r_1, r_2 and r_3 are the random parameters uniformly taken from the interval [0, 1] and $sign(r_3)$ is a function defined as:

$$\begin{aligned} sign(r_3) &= -1 \quad \text{where } r_3 \le 0.05 \\ &= 1 \quad \text{where } r_3 > 0.05 \end{aligned} \tag{11}$$

The two random parameters $rand_1$ and $rand_2$ of (8) are independent. If both are large, both the personal and social experiences are over used and the particle is driven too far away from the local optimum. If both are small, both the personal and social experiences are not used fully and the convergence speed of the technique is reduced. So, instead of taking independent $rand_1$ and $rand_2$, one single random number r_1 is chosen so that when r_1 is large, $(1-r_1)$ is small and vice versa. Moreover, to control the balance between global and local searches, another random parameter r_2 is introduced. For birds' flocking for food, there could be some rare cases that after the position of the particle is changed according to (9), a bird may not, due to inertia, fly towards a region at which it thinks is most promising for food. Instead, it may be leading toward a region which is in opposite direction of what it should fly in order to reach the expected promising regions. So, in the step that follows, the direction of the bird's velocity should be reversed in order for it to fly back to the promising region. $sign(r_3)$ is introduced for this purpose. In birds' flocking or fish schooling, a bird or a fish often changes directions suddenly. This is described by using a ''craziness'' factor and is modelled in the technique by using a craziness variable. A craziness operator is introduced in the proposed technique to ensure that the particle would have a predefined craziness probability to maintain the diversity of the particles. Consequently, before updating its position the velocity of the particle is crazed by,

$$V_i^{(k+1)} = V_i^{(k+1)} + P(r_4) * sign(r_4) * v_i^{craziness} \tag{12}$$

where r_4 is a random parameter which is chosen uniformly within the interval [0, 1]; $v^{craziness}$ is a random parameter which is uniformly chosen from the interval $[v_i^{\min}, v_i^{\max}]$; and $p(r_4)$ and $sign(r_4)$ are defined, respectively, as:

$$\begin{aligned} P(r_4) &= 1 \quad \text{when} \quad r_4 \le P_{cr} \\ &= 0 \quad \text{when} \quad r_4 > P_{cr} \end{aligned} \tag{13}$$

$$\begin{aligned} sign(r_4) &= -1 \quad \text{when} \quad r_4 \ge 0.5 \\ &= 1 \quad \text{when} \quad r_4 < 0.5 \end{aligned} \tag{14}$$

where P_{cr} is a predefined probability of craziness.

The steps of CRPSO algorithm are as follows:

Step 1: Population is initialized for a swarm of n_p vectors, in which each vector represents a solution of filter coefficient values.

Step 2: Computation of initial cost values of the total population, n_P.

Step 3: Computation of population based minimum cost value, i.e., the group best solution vector (*gbest*) and computation of the personal best solution vectors (*pbest*).

Step 4: Updating the velocities as per (10) and (12); updating the particle vectors as per (9) and checking against the limits of the filter coefficients; finally, computation of the updated cost values of the particle vectors and population based minimum cost value.

Step 5: Updating the *pbest* vectors, *gbest* vector; replace the updated particle vectors as initial particle vectors for step 4.

Step 6: Iteration continues from step 4 till the maximum iteration cycles or the convergence of minimum cost values are reached; finally, *gbest* is the vector of optimal IIR filter coefficients.

The justifications of choosing the value of different CRPSO parameters are as follows:

Reversal of the direction of bird's velocity should rarely occur; to achieve this, $r_3 \le 0.05$ (a very low value) is chosen when $sign(r_3)$ will be -1 to reverse the direction. If P_{cr} is chosen less or, equal to 0.3, the random number r_4 will have more probability to become more than P_{cr}, in that case, craziness factor $P(r_4)$ will be zero in most cases, which is actually desirable, otherwise heavy unnecessary oscillations will occur in the convergence curve near the end of the maximum iteration cycles as referred to (9). $v^{craziness}$ is chosen very small (=0.0001) as shown in Table 2. $r_4 \ge 0.5$ or, <0.5 is chosen to introduce equal probability of direction reversal of $v^{craziness}$ as referred to (12).

The design objective in this paper is to obtain the optimal combination of the IIR LP, HP, BP and BS filter coefficients, so as to acquire the maximum stop band attenuation with the least transition width. Here lies the author's contribution that this design objective has been attained by the proposed CRPSO technique. The values of the parameters used for RGA, PSO and CRPSO techniques are given in Table 2.

4 Simulation Results and Discussions

Extensive simulation study has been performed for comparison of optimization performances of three algorithms namely, RGA, PSO, and CRPSO, respectively, for the 8th order IIR LP, HP, BP and BS filter optimization problems. The design specifications followed for all algorithms are given in Table 1.

The values of the control parameters of RGA, PSO, and CRPSO are given in Table 2. Each algorithm is run for several times to get the best solution and the best results are reported in this paper. All optimization programs are run in MATLAB 7.5 version on core (TM) 2 duo processor, 3.00 GHz with 2 GB RAM.

Three aspects of the algorithms are investigated in this work namely, their accuracy, speed of convergence and stability. Figures 1, 4, 7 and 10 show the comparative gain plots in dB for the designed 8th order IIR LP, HP, BP and BS filters obtained for different algorithms. Normalized gain plots are shown in Figures 2, 5, 8 and 11 for the comparative study of 8th order IIR LP, HP, BP and BS filters. The best optimized numerator coefficients (b_k) and denominator coefficients (a_k) obtained after completion of predefined iteration cycles are reported in Tables 3, 6, 9 and 12. The values of statistical parameters for stop band attenuation in dB for 8th order IIR LP, HP, BP and BS filters designed using RGA, PSO, and CRPSO, respectively, are presented in Tables 4, 7, 10 and 13. Tables 5, 8, 11 and 14 show the maximum pass band ripple (normalized), maximum, minimum, average stop band ripple (normalized), and the transition widths for 8^{th} order IIR LP, HP, BP and BS filters designed using RGA, PSO and CRPSO, respectively. From the above tables and figures it can be explored that the proposed 8th order IIR filter designed with CRPSO attains the highest stop band attenuation in all cases with comparatively good figures for the rest of the parameters, such as stop band and pass band ripples, transition width etc. Figures 5, 8, 11 and 14 show the pole-zero plots for all 8th order IIR filters concerned with this paper for CRPSO based technique. These figures demonstrate the existence of poles within the unit circle which ensures the bounded input bounded output (BIBO) stability condition for the designed IIR filters.

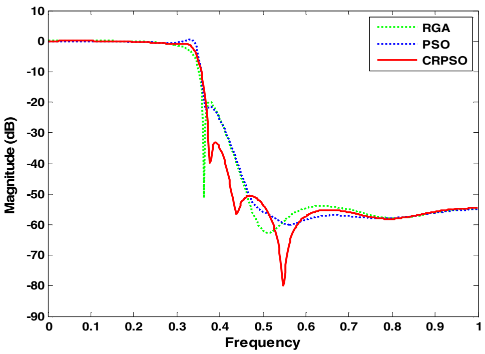


Fig. 1. Gain plots in dB for 8th order IIR LP filter using RGA, PSO and CRPSO

Table 1. Design Specifications of IIR LP, HP, BP and BS Filters

Filter Type	Pass band ripple (δ_p)	Stop band ripple (δ_s)	Pass band normalized edge frequency (ω_p)	Stop band normalized edge frequency (ω_s)
LP [19]	0.001	0.0001	0.35	0.40
HP	0.1	0.01	0.35	0.30
BP	0.1	0.01	0.35 and 0.65	0.3 and 0.7
BS	0.1	0.01	0.25 and 0.75	0.3 and 0.7

Table 2. Control Parameters of RGA, PSO and CRPSO

Parameters	RGA	PSO	CRPSO
Population size	120	25	25
Iteration cycles	400	100	100
Crossover rate	1	-	-
Crossover	Two Point Crossover	-	-
Mutation rate	0.01	-	-
Mutation	Gaussian Mutation	-	-
Selection	Roulette	-	-
Selection probability	1/3	-	-
C1, C2	-	2.05, 2.05	2.05, 2.05
$v_i^{\min}$, $v_i^{\max}$	-	0.01, 1.0	0.01, 1.0
wmax, wmin	-	1.0, 0.4	-
p_{cr}	-	-	0.3
$v^{craziness}$	-	-	0.0001

Table 3. Optimized Coefficients and Performance Comparison of Concerned Algorithms for 8th Order IIR LP Filter

Algorithms	Num. Coeff. (b_k)	Den. Coeff. (a_k)	Max. Stop Band Attenuation (dB)
RGA	0.0167 0.0059 0.0434 0.0234 0.0451 0.0302 0.0277 0.0120 0.0092	0.9996 -3.5213 7.1631 -9.4231 8.7904 -5.7905 2.6429 -0.7583 0.1066	20.000
PSO	0.0165 0.0060 0.0423 0.0237 0.0454 0.0286 0.0275 0.0122 0.0073	0.9996 -3.5201 7.1638 -9.4233 8.7894 -5.7906 2.6430 -0.7593 0.1072	21.5683
CRPSO	0.0169 0.0054 0.0424 0.0228 0.0456 0.0285 0.0275 0.0115 0.0092	0.9990 -3.5209 7.1621 -9.4221 8.7896 -5.7908 2.6431 0.7587 0.1072	33.1170

Table 4. Statistical Results for Stop Band Attenuation (dB) for 8th Order IIR LP Filter

Algorithm	Maximum	Mean	Variance	Standard Deviation
RGA	20.0000	42.9281	263.0129	16.2177
PSO	21.5683	44.5499	264.6049	16.2667
CRPSO	33.1170	48.3590	80.5940	8.9774

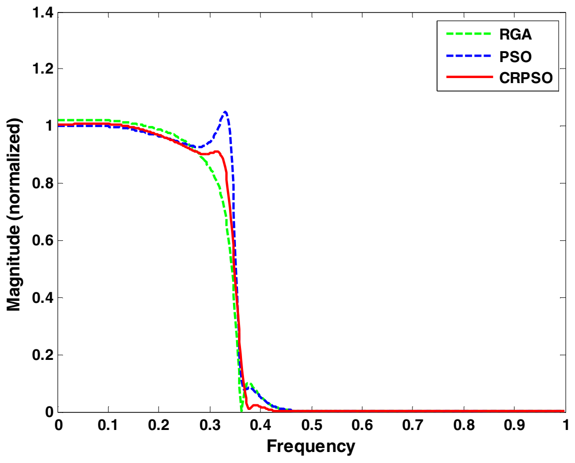


Fig. 2. Normalized gain plots for 8th order IIR LP filter using RGA, PSO and CRPSO

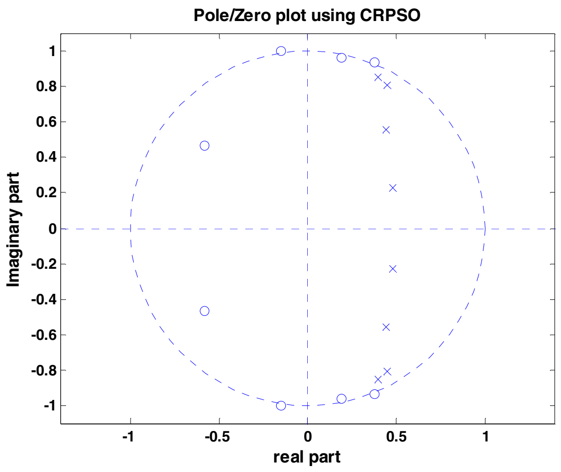


Fig. 3. Pole-zero plot of 8th order IIR LP filter using CRPSO

Table 5. Qualitatively Analyzed Results for 8th Order IIR LP Filter

Algorithm	Maximum Pass band ripple (normalized)	Stop band ripple (normalized)			Transition Width
		Maximum	Minimum	Average	
RGA	0.0214	0.1000	7.3286×10^{-4}	5.0366×10^{-2}	0.0341
PSO	0.0500	0.0835	1.0000×10^{-3}	4.2250×10^{-2}	0.0216
CRPSO	0.0086	0.0221	1.0000×10^{-4}	1.1100×10^{-2}	0.0370

Table 6. Optimized Coefficients and Performance Comparison of Concerned Algorithms for 8th order IIR HP filter

Algorithms	Num. Coeff. (b_k)	Den. Coeff. (a_k)	Max. Stop Band Attenuation (dB)
	0.1250 -0.7092 1.9588	0.9999 -2.1875 3.8221	
RGA	-3.3672 3.9090 -3.1264	-3.6220 2.9095 -1.3332	46.2199
	1.6821 -0.5585 0.0881	0.5678 -0.0861 0.0285	
	0.1252 -0.7091 1.9587	1.0001 -2.1874 3.8222	
PSO	-3.3671 3.9091 -3.1263	-3.6220 2.9096 -1.3333	47.7018
	1.6821 -0.5584 0.0881	0.5678 -0.0861 0.0285	
	0.1252 -0.7091 1.9587	1.0000 -2.1874 3.8223	
CRPSO	-3.3672 3.9090 -3.1263	-3.6220 2.9094 -1.3334	49.9710
	1.6820 -0.5584 0.0883	0.5679 -0.0861 0.0284	

Table 7. Statistical Results for Stop Band Attenuation (dB) for 8th Order IIR HP Filter

Algorithm	Maximum	Mean	Variance	Standard Deviation
RGA	46.2199	49.8589	13.2467	2.6391
PSO	47.7018	50.7807	9.4796	3.0789
CRPSO	49.9710	53.1421	3.8708	1.9674

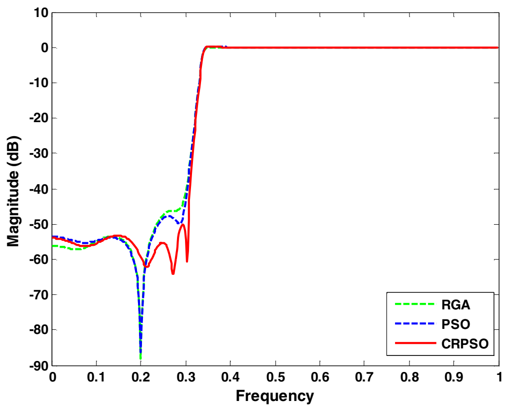


Fig. 4. Gain plots in dB for 8th order IIR HP filter using RGA, PSO and CRPSO

Table 8. Qualitatively Analyzed Results for 8th Order IIR HP Filter

Algorithm	Maximum Pass Band Ripple (normalized)	Stop Band Ripple (normalized)			Transition Width
		Maximum	Minimum	Average	
RGA	0.0146	0.48863×10^{-2}	0.39587×10^{-4}	0.24629×10^{-2}	0.0598
PSO	0.0186	0.41201×10^{-2}	0.47667×10^{-4}	0.20839×10^{-2}	0.0500
CRPSO	0.0356	0.31726×10^{-2}	6.2291×10^{-4}	0.18978×10^{-2}	0.0349

Table 9. Optimized Coefficients and Performance Comparison of Concerned Algorithms for 8th order IIR BP filter

Algorithms	Num. Coeff. (b_k)	Den. Coeff. (a_k)	Max. Stop Band Attenuation (dB)
RGA	0.1369 -0.0069 -0.0200 -0.0043 0.1897 0.0069 -0.0338 -0.0056 0.1253	0.9971 -0.0075 1.5866 -0.0094 1.7020 0.0000 0.8246 -0.0025 0.2247	18.2445
PSO	0.1274 0.0071 -0.0209 0.008 0.1857 0.0001 -0.0292 -0.0052 0.1299	0.9927 -0.002 1.5940 0.0029 1.6978 -0.0002 0.8079 -0.0034 0.2058	20.1389
CRPSO	0.1082 -0.0078 -0.0233 0.0018 0.1561 -0.0033 -0.0273 -0.0015 0.1037	1.0001 -0.0062 1.6899 0.0028 1.7556 -0.0023 0.8516 -0.0078 0.2038	22.7295

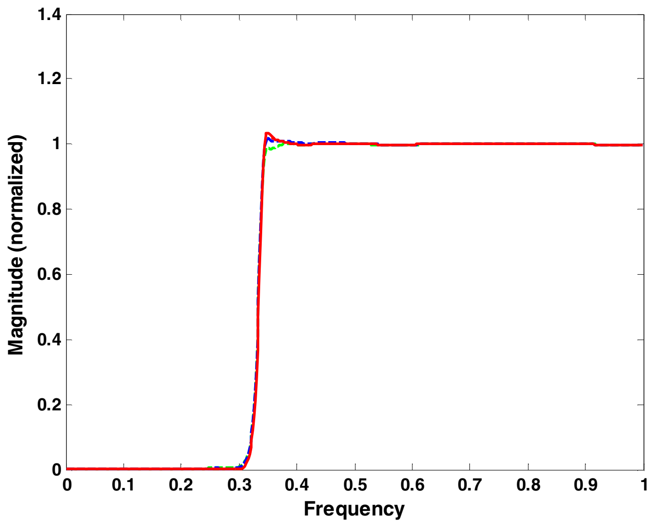


Fig. 5. Normalized gain plots for 8th order IIR HP filter using RGA, PSO and CRPSO

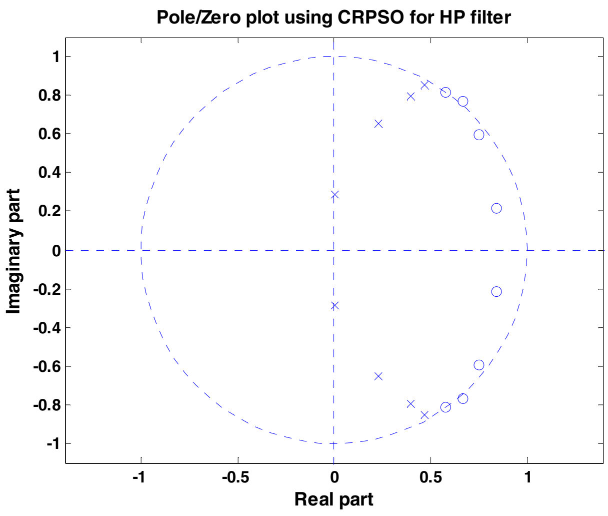


Fig. 6. Pole-zero plot of 8th order IIR HP filter using CRPSO

Table 10. Statistical Results for Stop Band Attenuation (dB) for 8th Order IIR BP Filter

Algorithm	Maximum	Mean	Variance	Standard Deviation
RGA	18.2445	20.3032	4.2382	2.0587
PSO	20.1389	21.4826	1.8054	1.3437
CRPSO	22.7295	24.4450	1.1011	1.0493

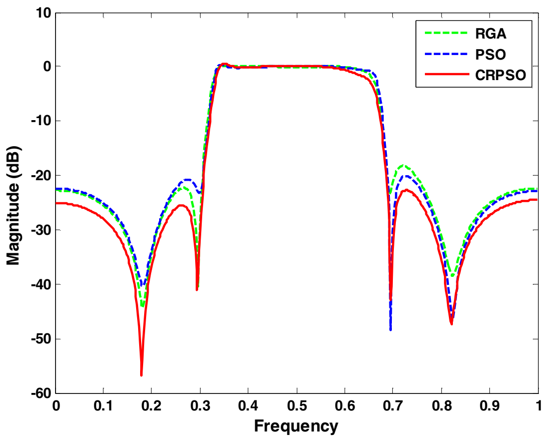


Fig. 7. Gain plots in dB for 8th order IIR BP filter using RGA, PSO and CRPSO

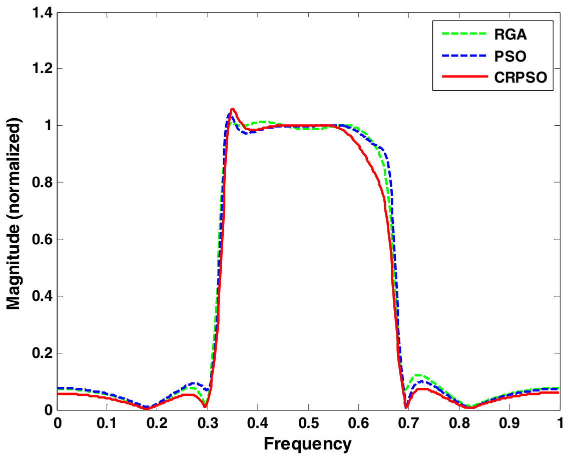


Fig. 8. Normalized gain plots for 8th order IIR BP filter using RGA, PSO and CRPSO

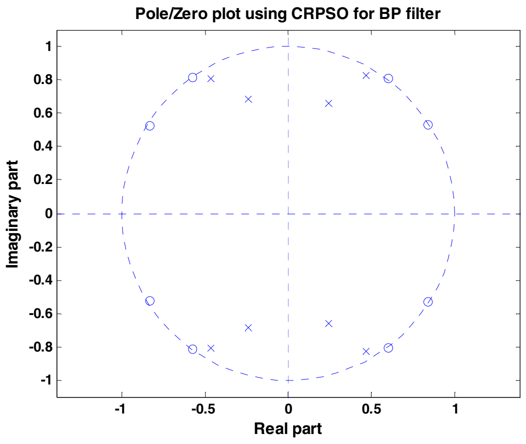


Fig. 9. Pole-zero plot of 8th order IIR BP filter using CRPSO

Table 11. Qualitatively Analyzed Results for 8th Order IIR BP Filter

Algorithm	Maximum Pass Band Ripple (normalized)	Stop band ripple (normalized)			Transition Width
		Maximum	Minimum	Average	
RGA	0.0134	12.24×10^{-2}	12.0000×10^{-3}	6.7200×10^{-2}	0.0311
PSO	0.0399	9.84×10^{-2}	3.7771×10^{-3}	5.1089×10^{-2}	0.0277
CRPSO	0.0578	7.30×10^{-2}	1.4313×10^{-3}	3.7200×10^{-2}	0.0409

Table 12. Optimized Coefficients and Performance Comparison of Concerned Algorithms for 8th order IIR BS filter

Algorithms	Num. Coeff.(b_k)	Den. Coeff. (a_k)	Max. Stop Band Attenuation (dB)
RGA	0.2269 -0.0189 0.5039 0.0170 0.6409 -0.0136 0.4866 0.0093 0.2189	1.0190 -0.0067 0.0968 0.0109 0.8671 0.0180 -0.0322 0.0177 0.1182	17.4734
PSO	0.2142 -0.0058 0.4833 -0.0008 0.6503 0.0097 0.4976 0.0041 0.2091	1.0073 -0.0069 0.0980 -0.0077 0.8902 -0.0073 -0.0198 -0.0048 0.1089	21.9740
CRPSO	0.2144 -0.0083 0.4817 -0.0055 0.6589 0.0001 0.4841 0.0050 0.2162	0.9959 -0.0061 0.0894 0.0040 0.8909 0.0038 -0.0273 -0.0003 0.1095	23.8659

Table 13. Statistical Results for Stop Band Attenuation (dB) for 8th Order IIR BS Filter

Algorithm	Maximum	Mean	Variance	Standard Deviation
RGA	17.4734	21.0867	13.0559	3.6133
PSO	21.9740	24.1658	4.8038	2.1918
CRPSO	23.8659	24.8641	0.5169	0.7190

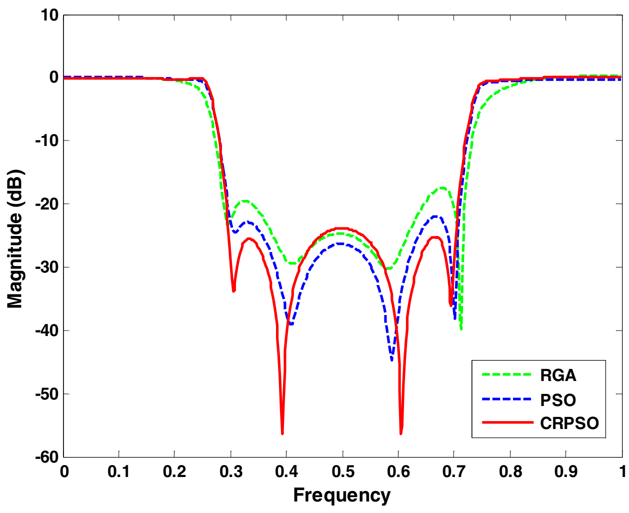


Fig. 10. Gain plots in dB for 8th order IIR BS filter using RGA, PSO and CRPSO

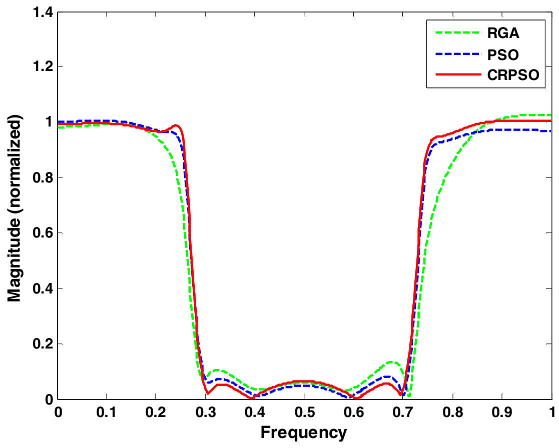


Fig. 11. Normalized gain plots for 8th order IIR BS filter using RGA, PSO and CRPSO

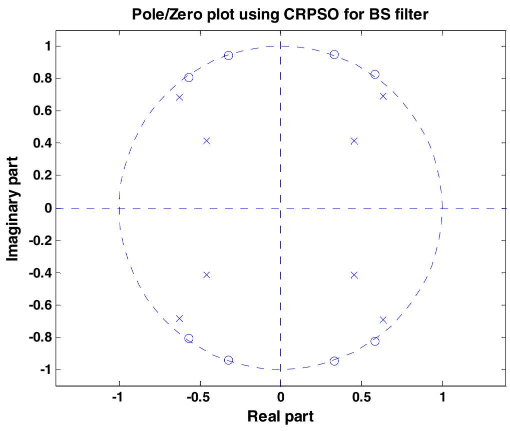


Fig. 12. Pole-zero plot of 8th order IIR BS filter using CRPSO

Table 14. Qualitatively Analyzed Results for 8th Order IIR BS Filter

Algorithm	**Maximum Pass Band Ripple (normalized)**	**Stop Band Ripple (normalized)**			**Transition Width**
		Maximum	**Minimum**	**Average**	
RGA	0.0268	13.38×10^{-2}	30.6000×10^{-3}	8.2200×10^{-2}	0.0535
PSO	0.0303	7.97×10^{-2}	5.8373×10^{-3}	4.2769×10^{-2}	0.0377
CRPSO	0.0344	6.41×10^{-2}	1.4978×10^{-3}	3.2799×10^{-2}	0.0410

Table 15. Comparison of Performance Criteria among algorithms published in relevant literatures

Reference	Proposed Algorithm	Filter Type	Order	Stop Band Attenuation (dB)	Max. Pass Band Ripple	Max. Stop Band Ripple	Transition Width
Luitel *et al.* [32]	DE-PSO	LP	9th	25	0.257	0.259	NR*
Luitel *et al.* [33]	PSO-QI	LP	9th	27	0.808	0.793	NR*
Karaboga *et al.* [34]	GA	LP	9th	14	NR*	NR*	NR*
Gao *et al.* [38]	DC	LP	6th	29	NR*	NR*	NR*
		HP	6th	42	NR*	NR*	NR*
Xue *et al.* [39]	GA	LP	7th	15	NR*	NR*	NR*
Present paper	CRPSO	LP	8th	33.1170	0.0086	0.00221	0.0370
		HP	8th	49.9710	0.0356	0.31726 e-2	0.0349
		BP	8th	22.7295	0.0578	7.30 e-2	0.0409
		BS [21]	8th	23.8659	0.0344	6.41 e-2	0.0410

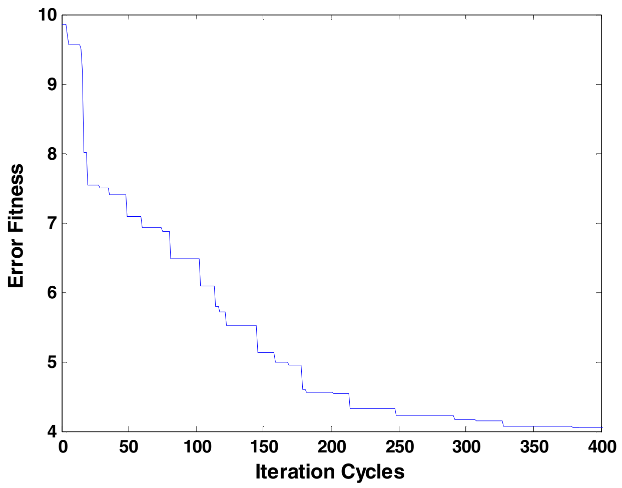


Fig. 13. Convergence profiles for RGA for 8th order IIR BS filter

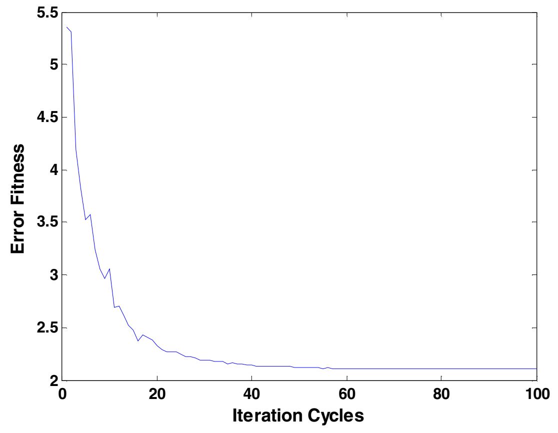


Fig. 14. Convergence profiles for PSO for 8th order IIR BS

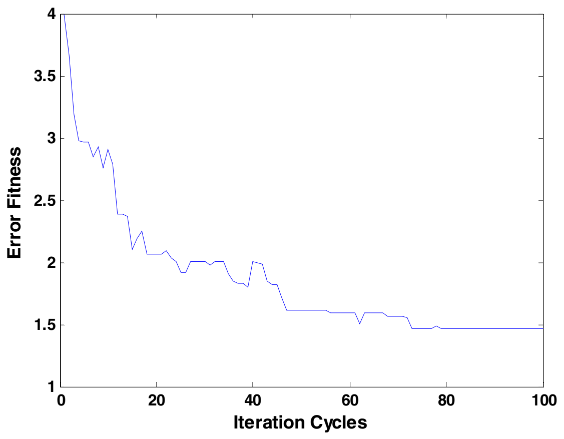


Fig. 15. Convergence profiles for CRPSO for 8th order IIR BS

Comparative study of results in terms of order, maximum attenuation, transition width, pass band and stop band ripples of IIR filters designed with different approaches adopted in different published literatures are reported in Table 15. Luitel *et al.* [32] proposed DE-PSO algorithm for the design of 9th order LP filter and maximum stop band attenuation, pass band and stop band ripples of approximately 25 dB, 0.257 and 0.259, respectively. Again, in [33], Luitel *et al.* proposed PSO-QI algorithm for the design of 9th order LP filter with the values of maximum stop band attenuation, pass band and stop band ripples of 27 dB, 0.808 and 0.793, respectively. Karaboga *et al.* proposed GA for the design of 9th order LP filter and the maximum value of stop band attenuation as 14 dB was reported in [34]. Gao *et al.* proposed differential cultural algorithm for the design of 6th order LP and HP filters in [38]. Maximum stop band attenuations of 29 dB and 42 dB for LP and HP filters, respectively, were reported there. Xue *et al.* also proposed GA for the design of 7th order LP filter and maximum stop band attenuation of 15 dB was reported in [39]. In the present paper CRPSO is proposed for the design of 8th order LP, HP, BP and BS filters. With this optimization technique, values of maximum stop band attenuation are 33.1170 dB, 49.9710 dB, 22.7295 dB and 23.8659 dB; maximum pass band ripple are 0.0086, 0.0356, 0.0578 and 0.0344; maximum stop band ripple are 0.221×10^{2}, 0.31726×10^{-2}, 7.30×10^{-2} and 6.41×10^{-2} and transition widths are 0.0370, 0.0349, 0.0409 and 0.0410 are obtained for LP, HP, BP and BS filters, respectively. So, CRPSO yields consistently higher stop band attenuation, lower stop band ripples with moderate control on the transition width and pass band ripples.

4.1 Comparison of Effectiveness and Convergence Profiles of RGA, PSO and CRPSO

Figures 13-15 depict the convergences of error fitness values obtained by RGA, PSO, and CRPSO for the 8th order IIR BS filter. Similar plots can also be obtained for the rest of the filters, which are not shown here.

As shown in Figures 13-15, RGA, PSO and CRPSO take 379, 85 and 79 iteration cycles to reach the error value of 4.043, 2.105 and 1.461, respectively, from which it can be concluded the CRPSO based approach finds the near sub-optimal solution of filter coefficients most fleetly among others with ensured grand minimum error value. With consideration of above facts and Figures 13-15, it can be easily inferred that the proposed CRPSO based optimization technique not only obtains the lowest error fitness value but also fast enough to achieve that. With a view to the above fact, it may finally be concluded that the performance of the CRPSO is the best among the three mentioned algorithms. All optimization programs are run in MATLAB 7.5 version on core (TM) 2 duo processor, 3.00 GHz with 2 GB RAM.

5 Conclusions

In this paper, a stochastic optimization algorithm, CRPSO, is applied to the optimal design of 8th order low pass, high pass, band pass and band stop IIR digital filters. The proposed filter design algorithm, CRPSO, is based upon the PSO in which pitfalls of conventional PSO have been judiciously managed with the perspective of closely mimicking the behaviour of fish in a school. The optimal filters thus obtained meet the stability criterion and show the best attenuation characteristics with reasonably good transition widths. The CRPSO algorithm converges very fast to the best quality optimal solution and reaches the lowest minimum error fitness value in the shortest number of iteration cycles. Statistically analysed results obtained for the CRPSO also justify the potential of the proposed algorithm for the realization of digital IIR filters.

References

1. Oppenheim, A.V., Buck, R.W.: Discrete-Time Signal Processing. Prentice-Hall, Englewood Cliffs (1999)
2. Proakis, J.G., Manolakis, D.G.: Digital Signal Processing. Prentice-Hall, Englewood Cliffs (1996)
3. Lang, M.C.: Least-squares design of IIR filters with prescribed magnitude and phase responses and pole radius constraint. IEEE Trans. on Signal Processing 48(11), 3109–3121 (2000)
4. Jackson, L.B., Lemay, G.J.: A simple Remez exchange algorithm to design IIR filters with zeros on the unit circle. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, USA, vol. 3, pp. 1333–1336 (1990)
5. Hussain, Z.M., Sadik, A.Z., O'Shea, P.: Digital Signal Processing-An Introduction with MATLAB Applications. Springer (2011)
6. Antoniou, A.: Digital Signal Processing: Signals, Systems and Filters. McGraw Hill (2005)
7. Lu, W.S., Antoniou, A.: Design of digital filters and filter banks by optimization: a state of the art review. In: Proc. European Signal Processing Conf., Finland, vol. 1, pp. 351–354 (2000)
8. Tsai, J.T., Chou, J.H., Liu, T.K.: Optimal design of digital IIR filters by using hybrid Taguchi genetic algorithm. IEEE Trans. on Industrial Electronics 53(3), 867–879 (2006)
9. Yu, Y., Xinjie, Y.: Cooperative co-evolutionary genetic algorithm for digital IIR filter design. IEEE Trans. on Industrial Electronics 54(3), 1311–1318 (2007)
10. Chen, S., Istepanian, R., Luk, B.L.: Digital IIR filter design using adaptive simulated annealing. Digital Signal Processing 11(3), 241–251 (2001)
11. Kalinli, A., Karaboga, N.: Artificial immune algorithm for IIR filter design. Engineering Applications of Artificial Intelligence 18(8), 919–929 (2005)
12. Karaboga, N., Kalinli, A., Karaboga, D.: Designing digital IIR filters using ant colony optimization algorithm. Engineering Applications of Artificial Intelligence 17(3), 301–309 (2004)
13. Karaboga, N.: A new design method based on artificial bee colony algorithm for digital IIR filters. Journal of Franklin Institute 346(4), 328–348 (2009)
14. Panda, G., Pradhan, P.M., Majhi, B.: IIR system identification using cats swarm optimization. Expert Systems with Applications 38(10), 12671–12683 (2011)
15. Chen, S., Luk, B.L.: Digital IIR filter design using particle swarm optimization. Int. J. Modelling, Identification and Control 9(4), 327–335 (2010)
16. Das, S., Konar, A.: A swarm intelligence approach to the synthesis of two-dimensional IIR filters. Engineering Applications of Artificial Intelligence 20(8), 1086–1096 (2007)
17. Fang, W., Sun, J., Xu, W.: A new mutated quantum-behaved particle swarm optimization for digital IIR filter design. EURASIP Journal on Advances in Signal Processing, 1–7 (2009)
18. Pan, S.T., Chang, C.Y.: Particle swarm optimization on D-stable IIR filter design. In: IEEE World Congress on Intelligent Control Automation, Taipei, pp. 621–626 (2011)
19. Saha, S.K., Kar, R., Mandal, D., Ghoshal, S.P.: IIR filter design with craziness based particle swarm optimization technique. World Academy of Science, Engineering and Technology 60, 1628–1635 (2011)
20. Saha, S.K., Chaudhuri, A., Kar, R., Mandal, D., Ghoshal, S.P.: Optimization of IIR high pass filter using craziness based particle swarm optimization technique. In: Proc. IEEE Symposium on Humanities, Science and Engineering Research, pp. 401–406 (2012)
21. Saha, S.K., Rakshit, J., Kar, R., Mandal, D., Ghoshal, S.P.: Optimal digital stable IIR band stop filter optimization using craziness based particle swarm optimization technique. In: Proc. IEEE World Congress on Information and Communication Technology, pp. 774–779 (2012)

22. Singh, B., Dhillon, J.S., Brar, Y.S.: Predator prey optimization method for the design of IIR filter. WSEAS Transactions on Signal Processing 9(2), 51–62 (2013)
23. Saha, S.K., Kar, R., Mandal, D., Ghoshal, S.P.: Seeker optimization algorithm: application to the design of linear phase finite impulse response filter. IET Signal Process 6(8), 763–771 (2012)
24. Saha, S.K., Kar, R., Mandal, D., Ghoshal, S.P.: Bacteria foraging optimization algorithm for optimal FIR filter design. International Journal of Bio-Inspired Computation 5(1), 52–66 (2013)
25. Holland, J.H.: Adaptation in Natural and Artificial Systems. Univ. Michigan Press, Ann Arbor (1975)
26. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE Int. Conf. on Neural Network, vol. 4, pp. 1942–1948 (1995)
27. Eberhart, R.C., Shi, Y.: Comparison between genetic algorithm and particle swarm optimization. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (eds.) EP 1998. LNCS, vol. 1447, pp. 611–616. Springer, Heidelberg (1998)
28. Sun, J., Fang, W., Xu, W.: A quantum-behaved particle swarm optimization with diversity-guided mutation for the design of two-dimensional IIR digital filters. IEEE Trans. on Circuits and Systems-II 57(2), 141–145 (2010)
29. Gao, Y., Li, Y., Qian, H.: The design of IIR digital filter based on chaos particle swarm optimization algorithm. In: IEEE Int. Conf. on Genetic and Evolutionary Computing, Hubei, pp. 303–306 (2008)
30. Ling, S.H., Iu, H.H.C., Leung, F.H.F., Chan, K.Y.: Improved hybrid particle swarm optimized wavelet neural network for modelling the development of fluid dispensing for electronic packaging. IEEE Trans. Ind. Electron. 55(9), 3447–3460 (2008)
31. Biswal, B., Dash, P.K., Panigrahi, B.K.: Power quality disturbance classification using fuzzy c-means algorithm and adaptive particle swarm optimization. IEEE Trans. Ind. Electron. 56(1), 212–220 (2009)
32. Luitel, B., Venayagamoorthy, G.K.: Differential evolution particle swarm optimization for digital filter design. In: IEEE Congress on Evolutionary Computation, pp. 3954–3961 (2008)
33. Luitel, B., Venayagamoorthy, G.K.: Particle swarm optimization with quantum infusion for the design of digital filters. In: IEEE Swarm Intelligence Symposium, USA, pp. 1–8 (2008)
34. Karaboga, N., Cetinkaya, B.: Design of minimum phase digital IIR filters by using genetic algorithm. In: Proc. IEEE 6th Nordic Signal Processing Symposium, Finland, pp. 29–32 (2004)
35. Saha, S.K., Kar, R., Mandal, D., Ghoshal, S.P.: Optimal IIR filter design using novel particle swarm optimization technique. International Journal of Circuits, Systems and Signal Processing, Springer 6(2), 151–162 (2012)
36. Saha, S.K., Kar, R., Mandal, D., Ghoshal, S.P.: Gravitational search algorithm: application to the optimal digital IIR filter design. Journal of King Saud University–Engineering Sciences (2013), doi: http://dx.doi.org/10.1016/j.jksues.2012.12.003
37. Mondal, S., Ghoshal, S.P., Kar, R., Mandal, D.: Craziness based Particle Swarm Optimization Algorithm for FIR Band Stop Filter Design. Journal of Swarm and Evolutionary Optimization 7, 58–64 (2012)
38. Gao, H., Diao, M.: Differential cultural algorithm for digital filters design. In: IEEE 2nd Int. Conf. on Computer Modeling and Simulation, pp. 459–463 (2010)
39. Xue, L., Rongchun, Z., Qing, W.: Optimizing the design of IIR filter via genetic algorithm. In: IEEE Int. Conf. on Neural Networks and Signal Processing, China, pp. 476–479 (2003)

Effect of Spatial Structure on the Evolution of Cooperation in the N-Choice Iterated Prisoner's Dilemma

Xiaoyang Wang[1], Yang Yi[2], Huiyou Chang[3], and Yibin Lin[3]

[1] School of Business, Sun Yat-sen University
Guangzhou, Guangdong, China
wxy_lele@163.com

[2] School of Information Science and Technology, Sun Yat-sen University
Guangzhou, Guangdong, China
issyy@mail.sysu.edu.cn

[3] School of Software, Sun Yat-sen University
Guangzhou, Guangdong, China
huiyou.chang@gmail.com,
yiftd@sina.com

Abstract. The evolution of cooperation is an enduring conundrum in biology and the social sciences. The prisoner's dilemma game has emerged as the most promising mathematical metaphors to study cooperation. Mechanisms promoting the evolution of cooperation in two-player, two-strategy spatial iterated prisoner's dilemma (IPD) games have been discussed in great detail over the past decades. Understanding the effects of repeated interactions in *n*-choice spatial IPD game is a formidable challenge. In this paper, the simulations are conducted with four different types of neighbourhood structures, and agents update their strategies by probabilistically imitating the strategies of better performing neighbours. During the evolution each agent can modify his own strategy and/or personal feature via a particle swarm optimization approach in order to improve his utility. The particle swarm optimization (PSO) approach is a bionic method which can simulate the interactions among agents in a realistic way. The results show that the evolutionary stability of cooperation does emerge in *n*-choice spatial IPD game, and the consideration of social cohesion in PSO approach promotes the evolution of cooperation. In addition, the neighbourhood structures and cost-to-benefit ratio increase the capability of cooperation and prevent the invading of defectors.

Keywords: evolution of cooperation, *N*-choice spatial iterated prisoner's dilemma (IPD), particle swarm optimization (PSO).

1 Introduction

Cooperation is a key force in evolution, exists in all scales of organization from unicellular organisms to complex modern human society [9, 27]. The emergence and stabilization of cooperative behavior has become a core problem in biology, economics, mathematics, computer science and sociology [20-22, 28]. Evolutionary game theory has proven to be one of the most fruitful approaches to investigate this problem, using evolutionary models based on so-called social dilemmas [29]. An enormous body of studies, however, has concentrated on the iterated prisoner's dilemma (IPD) [7, 8], which was proposed by Axelrod in the early 1980s. Those works mainly focused on how to evolve cooperative behavior from a population of agents involved in this nonzero-sum game. Axelrod applied evolution in order to determine whether it was possible to obtain successful strategies of mutual cooperation. In his original human-centered competitions, the tit-for-tat (TFT) strategy, based on reciprocal cooperation and submitted by Rapport, proved to be the most successful strategy.

Early studies of the IPD using the co-evolutionary approach consider only two choices, i.e., cooperation and defection for each player in the game. Over the years, various mechanisms have been proposed to help explain and understand cooperative phenomena. One of the possible mechanisms accounting for the promotion of cooperation is the consideration of spatial structure (reciprocity) [24]. The presence of structure means that each individual does not interact with each other, but with a small subset of the population, which constitutes his/her neighborhood and is arranged according to an underlying network of relationships. This idea was very successfully introduced by Nowak and May in their seminal paper [30], they considered a spatial version of the Prisoner's Dilemma game and showed that spatial structure enables both cooperators and defectors to persist indefinitely (see [24] for a review).

A key characteristic of spatial reciprocity is the fact that short-range interactions restrict the number of choices which can affect the behavior of a given player at a particular site. The spatial structure allows cooperators to form clusters, in which the benefits of mutual cooperation can out weight the losses against defectors. Therefore, the general understanding has been that cooperative behavior differs when the spatial structure is limited. However, several studies have shown that it is actually possible to evolve and sustain cooperation in IPD game with n-choice, i.e., multiple levels of cooperation. Darwen and Yao [10-12] were among the first who studied the n-choice IPD a co-evolutionary learning environment, and their model showed more realistic behavioral interactions between agents than conventional two-choice IPD game. They demonstrated approaches to improve classical co-evolutionary learning and increase higher generalization performance in n-choice IPD game. In addition to the co-evolutionary algorithm, Chong [1-5] investigated the co-evolutionary learning of n-choice IPD game with indirect reciprocity. Their investigation has extended the IPD to bridge the gap between model and real-world dilemmas and suggested why and how the indirect reciprocity can promote the cooperation. To date, the effect of spatial reciprocity on IPD game with multiple choices has received much less attention [31]. The importance of considering the n-choice IPD game should not be underestimated, as they may be essential in economic and biology applications.

In this paper, we extend this line of research by studying the effects of spatial reciprocity on the evolution of cooperation in the n-choice IPD game. Agents in the population are mapped into four different types of two-dimensional lattice for comparison, that is, the regular-connected network, the Moore network, the Von-Neumann network and the small world network. At each time step, the focal agent participates in a game instance with other agents drawn from its local neighborhood. The number of choices and the payoff matrix defines an agent's strategy, which is used to select an action. A bionic method, the PSO algorithm with synchronous updating is used to evolve the strategies over time. In this paper, the behavior performances of players with different environmental factors in regular-connected network are first examined, such as the number of choices, the size of population et al. Comprehensive numerical simulations across a range of parameter settings by using PSO algorithm is to check whether cooperation can still be maintained in a regular-connected network, and the results are used for the comparison of the cases when the agents move in complex spatial structures. Secondly, the effect of spatial structure on the evolution of cooperation is evaluated in a well-mixed population, and examines the ability of cooperation promoting. Thirdly, the influence of the cost-to-benefit ratio is tested in this n-choice IPD game.

The rest of this paper is organized as follows. An overview of the core IPD problem and relevant historic related work is presented in Section 2. A summary of the co-evolutionary model is given in Section 3, along with a description of spatial structure used to evolve cooperation. Section 4 explains the experimental procedure followed for this study, and the results are analyzed. Section 5 concludes this paper by summarizing some of the major experimental findings.

2 Overview of the Iterated Prisoner's Dilemma

2.1 Conventional IPD Game

In the conventional IPD game, each player has two choices: cooperation and defection. A player would receive payoffs as the payoff matrix set when his opponent makes his choice in IPD game [2]. A reward (R) is given when both players choose to cooperate, whereas punishment (P) will be given if both of them choose to defect. In the situation where one player defects and the other player cooperates, the one who defects is awarded a tempting reward (T) but the one who cooperates will be given the sucker's punishment (S). Accordingly, a dilemma will always exist when given the rules $T>R>P>S$ and $2R>T+S$.

Table 1 illustrates the game in terms of costs and benefits to the players. A cooperative act results in a benefit b to the opposing player and a cost c to the cooperator, where $b>c>0$. Defector incurs neither costs nor benefits in some previous studies [26], which means $k=0$. However, in order to ensure there exists no negative payoffs in the matrix, let $k=c$ in this paper. Under this situation, if the opponent cooperates, a player gets the reward $R= k+(b-c)$ if he/she also cooperates, but can get $T= k+b$ by defecting. If the opponent defects, a player gets the lowest payoff $S=k-c$ for being cooperative and $P=k$ for being defect. As the definition of prisoner's dilemma game, the defection is the better choice regardless of what the opponent plays.

Table 1. Payoff matrix of prisoner's dilemma game

B \ A	**Cooperate**	**Defect**
Cooperate	*k+(b-c)*	*k-c*
Defect	*k+b*	*k*

2.2 *N*-Choice IPD Game

In the studies of Chong and Yao[1,2], the n-choice model is based on a simple interpolation of the conventional IPD. Specifically, the n-choice IPD is linearly interpolated from the 2-choice IPD using equation (1) [10, 11, 2-5]:

$$P_A = x - yC_A + zC_B, \quad -1 \leq C_A, C_B < 1 \tag{1}$$

Where p_A is the payoff of player A, given that c_A and c_B are the cooperation levels of the choices that agents A and B make respectively. Note that the payoff matrix must satisfy the following conditions in a multiple choices IPD game:

1) for $c_A < c_A'$ and constant c_B: $p_A(c_A, c_B) > p_A(c_A', c_B)$;
2) for $c_A \leq c_A'$ and $c_B < c_B'$: $p_A(c_A, c_B) < p_A(c_A', c_B')$;
3) for $c_A < c_A'$ and $c_B < c_B'$: $p_A(c_A', c_B') > \left(\frac{1}{2}\right)(p_A(c_A, c_B') + p_A(c_A', c_B))$.

The above three conditions are similar to those for the conventional IPD game. The four corners are the same payoffs for the 2-choice IPD and that any 2×2 sub-matrix of the nxn matrix is itself a 2-choice IPD. The variables x, y and z in equation (1) can be calculated when the payoff matrix of 2-choice IPD game is fixed. Once given the payoff equation and the three conditions above, an n-choice IPD can be formulated [2, 4]. However, only the number of choices is fixed where the line between full cooperation and full defection is symmetrical distribution of the zero. For example, consider the case of an IPD game of 4 choices (Table 2 shows) which is based on the conventional IPD

Table 2. Payoff matrix of four-choice IPD game

A \ B	+1	$+\frac{1}{3}$	$-\frac{1}{3}$	-1
+1	4	$2\frac{2}{3}$	$1\frac{1}{3}$	0
$+\frac{1}{3}$	$4\frac{1}{3}$	3	$1\frac{2}{3}$	$\frac{1}{3}$
$-\frac{1}{3}$	$4\frac{2}{3}$	$3\frac{1}{3}$	2	$\frac{2}{3}$
-1	5	$3\frac{2}{3}$	$2\frac{1}{3}$	1

game (Table 1 shows). If in Table 1, T=5, R=4, P=1, S=0, equation (2) is the matched equation for the four-choice IPD payoff matrix.

$$P_A = 2.5 - 0.5C_A + 2C_B, \quad -1 \le C_A, C_B < 1 \tag{2}$$

To normalize the range of cost and benefit, we define $r=\frac{c}{b}$ (b, c are illustrated in 2.1) as cost-to-benefit ratio. Payoffs of agents in *four*-choice IPD game can be calculated based on Table 1, Equation (1) and cost-to-benefit ratio. For example, in a *four*-choice payoff matrix as Table 2 shows, where $r=\frac{1}{4}$, $c=k=1$, $b=4$, $x=2.5$, $y=0.5$, $z=2$. If $r=\frac{1}{2}$, $c=k=1$, we can calculate that $b=2$, $x=1.5$, $y=0.5$, $z=1$.

3 The Model

3.1 Strategy Update

For strategy update, we use PSO algorithm to evolve the pool of agents' strategies. The PSO technique was introduced by Kennedy and Eberhart [18]. Inspired by the flocking behaviour of birds, PSO has been applied successfully to function the optimization, game learning, data clustering, and image analysis and neural networks training [14-17]. PSO involves "flying" a swarm (or population) of n-dimensional particles, and through a problem space, each possible solution to the optimization problem need to search a single optimum or multiple optima. Each particle has its own velocity, a memory of the best position it has obtained thus far (referred to as its personal best position), and knowledge of the best solution found by other particles (referred to as the global best solution).

In the PSO algorithm, each particle adjusts its position in a direction toward its own personal best position in a direction toward its own personal best position and the global best position. The velocity of the particle is calculated using:

$$V_{id}^{k+1} = \omega V_{id}^{k} + c_1 rand(0,1)\left(P_{id}^{k} - X_{id}^{k}\right) + c_2 rand(0,1)\left(P_{gd}^{k} - X_{id}^{k}\right) \tag{3}$$

where for particle i, the position vector can be represented by $X_i = (X_{i1}, X_{i2}, \dots, X_{id})$, V_{id} presents the velocity of the ith particle on the specific d-dimension. ω is the inertia weight, c_1 and c_2 are acceleration coefficients. P_{id}^{k} is the best position of particle i on dimension d at iteration k, and P_{gd}^{k} is the dth dimensional best position in the whole particle swarm at iteration k.

For particle i, its position x_{id} is changed according to:

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \tag{4}$$

To prevent velocities from growing too fast, velocity clamping is usually used to restrict step sizes. That is, if $\left|V_{id}^{k+1}\right| > V_{max}^{d}$ then

$$V_{id}^{k+1} = sign(V_{id})V_{max}^{d} \tag{5}$$

Various studies [16]-[18] have shown that under certain conditions convergence is guaranteed to a stable equilibrium point. These conditions include

$$\omega > \frac{1}{2}(c_1 + c_2) - 1 \tag{6}$$

$$1 > \omega > 0 \tag{7}$$

The P_{gd}^k and P_{id}^k are given by

$$P_{gd} = arg\, max\{\frac{1}{n}\sum_{i,j=1}^{n} \sum\nolimits_{\forall C_i, C_j \in C}(x - yC_i + zC_j)\} \tag{8}$$

$$P_{id} = arg\, max\, \{(x - yC_i + zC_j)\} \tag{9}$$

$$\forall C_i, C_j \in C \in [-1,1], i, j = 1,2 \dots n, i \neq j$$

Where C_i, C_j are strategies in the payoff matrix, and are generated symmetrical of the zero once the level of choice is fixed. -1 means full defection while +1 means full cooperation.

Combining with the prisoner's dilemma framework, each particle in the swarm is associated with a behaviour (or "strategy"), and the purpose of the behaviours is to maximize the payoff of individual or the group. The strategy is used to compute its position in the next step. An interpretation of multiple levels of choice between cooperation and defection is employed to the PSO approach.

3.2 The Spatial Environment for Strategy Co-evolution

In spatial evolutionary environment, the players (or agents) of a population are distributed on a regular grid and interact with other players in its neighbourhood. In our model, the players are mapped to network nodes (vertices) and the edges (or links) which dictate the interaction topology. Each agent participates in an interactive game with l-1 other agent drawn from its local neighborhood. Figure 1 depicts the four examples of neighborhood structure for interaction, and single agent is located in the node (vertex) of the grid world. In Figure 1(a) and (b), the two examples show that agent can only interact with the agents in dotted box. The regular-network (Figure 1 (d)) is two-dimensional and agents are connected by the edges. As for the small-world network, we use a version similar to the one introduced by Watts and Strogatz [25]. From the two-dimensional regular-network substrate we rewire each link with probability ρ (Figure 1 (c)). In this paper, we allow neither self or repeated links nor disconnected graphs to ensure that the individuals in the population are highly clustered and have relatively short path length. When ρ=0, it's essentially a regular-network. As ρ=1, it is defined as full-connected network.

The calculation of the neighborhood best particle depends on the spatial neighborhood structure agents used. Various spatial neighborhood structures have been defined [23], of which the four neighborhood structures of Figure 1 are used in this study. The neighborhood structures extend the 1-D lattice from the local best position

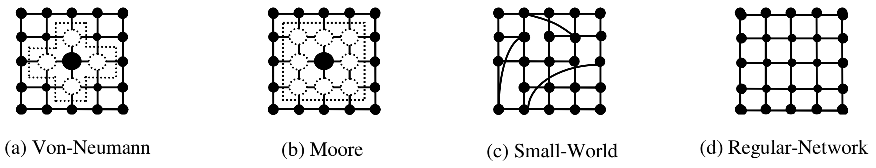


Fig. 1. Examples of neighborhood structures

into a two-dimensional lattice in variable space, allowing more intricate information sharing behavior. Studies have shown that the neighborhood structure outperforms other network structures on several benchmark problems [13, 23, 26].

In general, the neighborhood best position is calculated as

$$P_{gd}^{k+1} \in \{NE_i | f(P_{gd}^k) = \max\{f(x), \forall x \in NE_i\}\} \tag{10}$$

with the neighbourhood defined as

$$NE_i = \{P_{gd}^k(i-l), \dots, P_{gd}^k(i-1), P_{gd}^k(i), P_{gd}^k(i+1), \dots, P_{gd}^k(i+l)\} \tag{11}$$

for neighbourhood size of l.

Every agent is initialized with a random strategy, represented by a real number. The individuals play the game over a fixed number of generations, each consisting of a number of repeated interactions. The utility (fitness) of each agent is determined by summing its payoffs in the game against the group numbers. At the end of each generation, all agents are presented with an opportunity to update their strategies according to the payoffs received.

3.3 Training Algorithm

The biomimetic methods [5, 6, 21] have been widely used in the iterated prisoner's dilemma game. The complete PSO based co-evolutionary algorithm is listed in Figure 2. The algorithm in Figure 2 can be interpreted as follows: A player corresponds to a single particle in the swarm. The collections of player complete in the n-choice IPD game. Each player has its own "strategy" which is represented by the particle's current position. By recording the personal best positions, particles have to compete against previously located best solutions. The neighborhood best solutions are also recorded for particles to choose their strategies.

4 Simulation Experiments and Results

Extensive computational simulations have been carried out to investigate the population dynamics of the games played. The experiments in the first part are conducted to analyse the general performance of strategy co-evolution. The second part of the experiments mainly examines the influence of spatial structure to the cooperation

```
Let
$N$ be the number of players in a prisoner's dilemma game;
$L$ be the number of levels in a payoff matrix;
$PNum$ be the size of particle swarm;
$S$ be the spatial structure of the IPD game;
$k$ and $K$ be the current generation and maximum number of iterations, respectively;
$x_{id}^{k}$ and $v_{id}^{k}$ be the position and velocity of the $i$th particle at iteration $k$ on the specific $d$-dimension, respectively;
$v_{si\max}^{d}$ be the $d$th dimensional maximum velocity of $i$th particle and its neighborhoods;
$fitness(i)$ be the fitness function of the $i$th particle;
$p_{sigd}^{k}$ be the $d$th dimensional global best position of the $i$th particle and its neighborhoods ;
$p_{id}^{k}$ be the $d$th dimensional history best position of the $i$th particle so far during $k$ iteration;
Step 1(Initialization): For each particle $i$ and dimension $d$
    Step 1.1: Initialize $N$, $L$, $PNum$, $K$, .
    Step 1.2: Initialize $x_{id}^{k}$ with an integer between -1 and 1 according to the payoff matrix randomly.
    Step 1.3: Initialize $v_{id}^{k}$ with a real number between $-v_{si\max}^{d}$ and $v_{si\max}^{d}$ randomly.
    Step 1.4: Calculate $fitness(i)$.
    Step1.5: Initialize $p_{id}^{k}$ with a copy of $x_{id}^{k}$ .
    Step 1.6: Initialize $p_{sigd}^{k}$ with the best $p_{id}^{k}$ among the $PNum$ particles.
Step 2: Repeat until $k>K$.
    Step 2.1: Determine $N$, $L$, $PNum$ of the game.
    Step 2.2: Random choose $N$ players and for each player in the game do
        Step 2.2.1: Compare performance against current personal best position, if need to update then use Eq.(9).
        Step 2.2.2: Compare performance against neighborhoods' best position, if need to update then use Eq.(10).
        Step 2.2.3: Update velocity according to Eq.(3).
        Step 2.2.4: Update position according to Eq.(4).
        Step 2.2.5: Update $fitness(i)$.
    Step 2.3: Choose other $N$ players from $PNum$ population and play game.
Step 3: Output the fitness, position of each player in the population.
```

Fig. 2. The PSO algorithm for multiple choices IPD game in spatial environment

learning in the population. We compare the proportions of full cooperators (+1) and the average cooperation in both the spatial structures and regular network. The third part of the experiments is conducted to compare the proportions of full cooperators by varying the cost-to-benefit ratio *r* in all the spatial structures.

4.1 General Performance of Strategy Co-evolution by Using PSO Approach in Regular Network

The experiments of this part are testing the co-evolutionary process of behaviour performances in *n*-choice IPD game with regular connected network. The following parameters are used in the PSO: *PNum* from 8 to 64, *L* from 4 to 32. In order to keep the uniform distribution of choices during the initialization process, each choice is set to players with the same probability. For example, if *PNum*=8 and *L*=4, then each player has 25% probability to get one of the four choices. Each iteration two players are chosen for the game, and the average payoff is calculated between two players.

Figure 3 shows the average frequencies for the choices that are played at each generation of the PSO approach. Because of the limitation of the figure, if the number of choices is higher than four, some specific choices are shown in Figure 3. From Figure 3, it can be found that the frequency of full cooperation (+1) is significantly higher than other choices, indicating that there are more full cooperative plays by the evolving strategies. If the *PNum* is fixed, comparing Figure 3(a) and (b), (c) and (d), (e) and (f), as the number of choices increases, players choose several levels of choices instead of full cooperation (+1). This is quite different from the classical IPD game with two choices [27], where the mutual cooperation runs quite consistently and quickly. It illustrates that more choices indeed made cooperation more difficult to evolve.

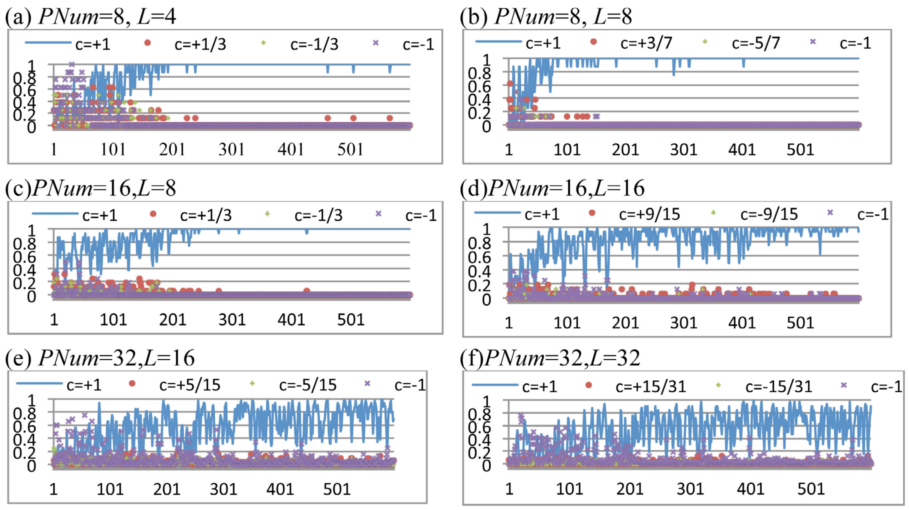


Fig. 3. The probability of each choice is calculated during the iteration. The figure can't show each choice in detail if L>4, however, some specific points are show in this figure.

Figure 4 shows the plots of average payoff of each generation. The average payoffs in Figure 4 reflect the changes of the behaviour performances. As the increase of population and choice number, more middle levels of choice are chosen. This experimental finding is rather similar to some phenomena in our human society, first is full cooperation or full defection is easier to emerge in a group of people with small population size than in a larger one; the second is more choices for people leads to more complex choosing diversity.

Figure 3 and Figure 4 show the situation that each player has the same opportunity to get a choice during the initialization process. Figure 5 and Figure 6 show the situation that the chance is not uniformly for each player in the initialization process. In order to show the comparisons clearly, the behaviour performances of L=16 are shown in Figure 5, and the Figure 6 plots the average payoff. From Figure 5, it can be seen that no matter how the choices are distributed during the initialization process, mutual full cooperation is always the most popular choice. As the number of population increases, players choose other choices instead of mutual cooperation. The results of cooperation evolution by using PSO approach can be clearly seen from the data in Table 3 and Table 4. The means of average payoff are above 2.5.

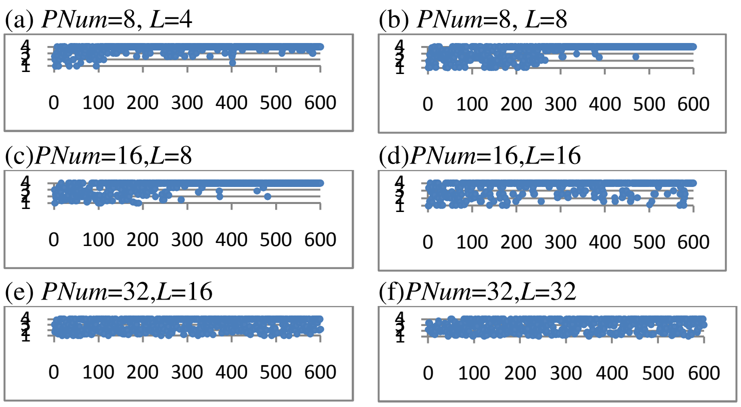


Fig. 4. The average payoff of each player in the multiple choices IPD game

Table 3. Comparison results from the experiments that used the *n*-choice IPD with 600 iterations. "Mean" indicates the average payoff of each agent during the iteration. "Max" indicates the highest average payoff of agent during the iteration. "Min" indicates the lowest average payoff of agent during the iteration.

Environment	Mean±StdDev	Max	Min
PNum=8, *L*=4	3.76±0.33	4	1
PNum=8, *L*=8	3.50±0.76	4	1
PNum=16,*L*=8	3.54±0.76	4	1
PNum=16,*L*=16	3.51±0.80	4	1
PNum=32,*L*=16	2.98±1.22	4	1
PNum=32,*L*=32	2.68±1.62	4	1

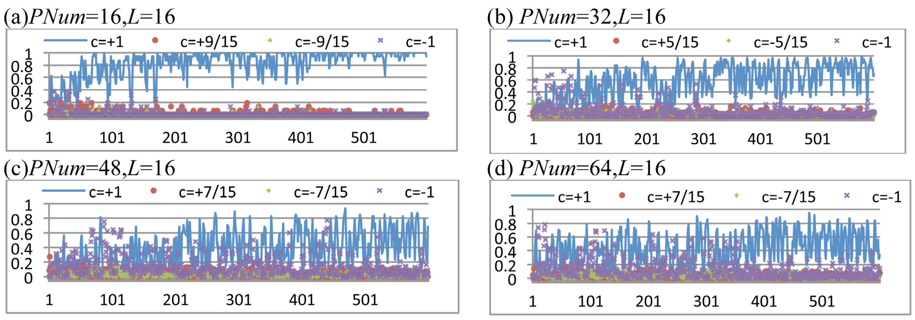


Fig. 5. The probability of each choice is calculated during the iteration. The figure can't show 16 choices in detail, so some specific points are show in this figure.

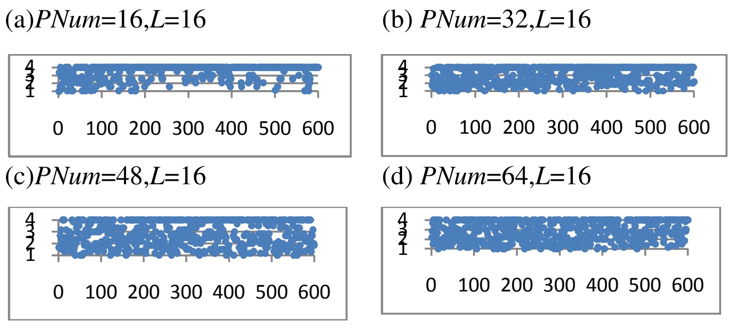


Fig. 6. The average payoff of each player in the IPD game with 16 levels of choice

Table 4. Comparison results from the experiments that used in the *n*-choice IPD with 600 iterations. "Mean" indicates the average payoff of each agent during the iteration. "Max" indicates the highest average payoff of agent during the iteration. "Min" indicates the lowest average payoff of agent during the iteration.

Environment	Mean±StdDev	Max	Min
PNum=16,*L*=16	3.51±0.80	4	1
PNum=32,*L*=16	2.98±1.22	4	1
PNum=48,*L*=16	2.62±1.33	4	1
PNum=64,*L*=16	2.68±1.32	4	1

4.2 Evolution of Cooperation by Using PSO Approach in Spatial Structure Environment

In this section, all experiments are performed on spatial network consisting 10×10=100 agents, randomly initialised with 25% of each four choice at the start of the game. Every agent played iteratively against one another within its social groups for 700 generations. At the end of each generation, PSO algorithm will give each agent an opportunity to update the strategies. Agents interact with each other in the Von-Neumann and Moore network based on Figure 1(a) and (b). The small-world network is generated by setting the random rewire probability ρ=0.5.

The main goal of experiments in this part is to test the co-evolutionary process of behaviour performances in four-choice IPD game by using PSO algorithm. The payoff matrix used in this part is shown in Table 2. Figure 7 shows the average frequencies of the choices that are played at each generation in the regular network by using the PSO approach. From Figure 7, it can be found that the frequency of full cooperation (+1) is significantly higher than other choices, indicating that there are more full cooperative plays by the evolving strategies. The other choices are still exist with relatively low frequency.

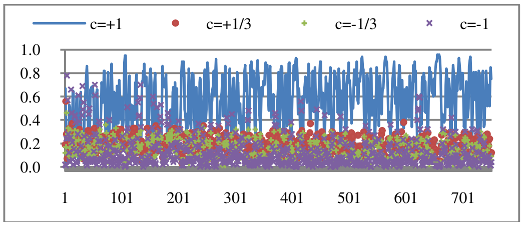


Fig. 7. The probability of each choice in regular network during the iteration

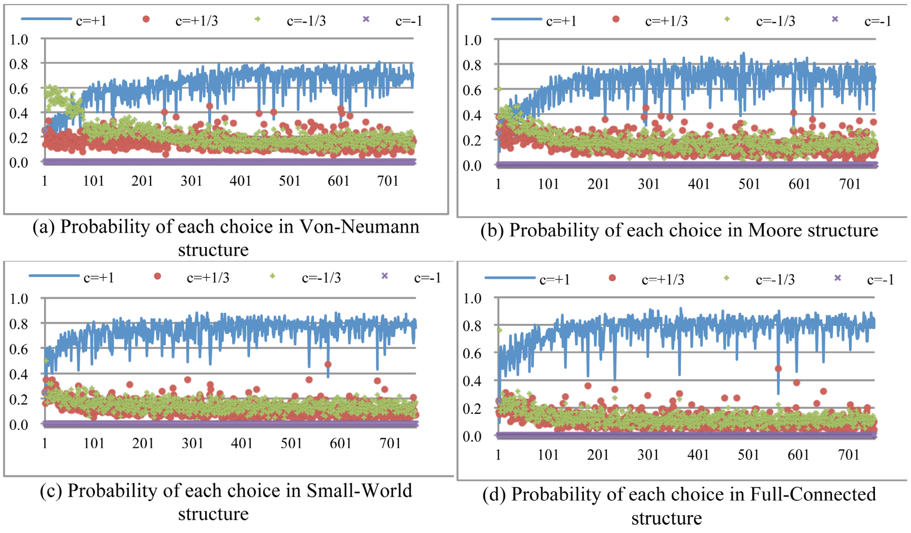


Fig. 8. Probability of each choice in different spatial environment during the iteration

Figure 8 shows the average frequencies of the choices that are played at each generation in four kind of spatial structures by using the PSO approach. Compared with Figure 7, it can be found that full cooperation (+1) is the highly dominated choice and the spatial environment is conducive for cooperation to emerge. The result thus far suggest that the iterated interactions of population in spatial environment can indeed promote cooperation, however, the behavioral outcome is correlated with the structures of the neighborhood. To further ascertain this, Figure 9 shows the average frequency of full cooperation in four spatial structures. From the figure, it is clear that the full-connected structure has increased more cooperation than others. Players in the random rewired network, such as the small-world and full-connected network are easier to cooperate than in the regular networks, such as Von-Neumann and Moore networks.

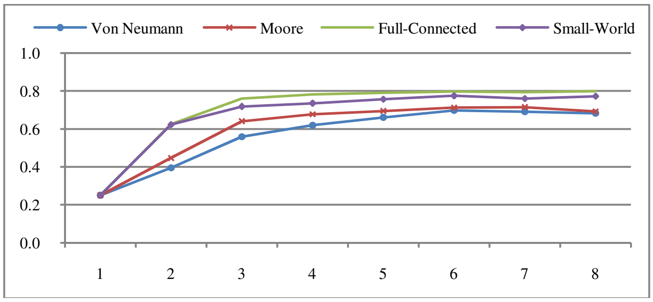


Fig. 9. The probability of full cooperation (+1) in each spatial structure

4.3 Levels of Cooperation with Varying the Cost-to-Benefit Ratio

In the previous sets of experiment, we compare the proportions of full cooperation with and without spatial environment by using PSO algorithm. In this part of experiment, the cost-to-benefit ratios, r is observed to find the influence to the evolution of cooperation. Payoffs of agents are calculated based on Table 3, Equation (1) and the cost-to-benefit ratios.

In order to see the proportion of full cooperation across different cost-to-benefit ratios within the immediate neighborhood, a close examination of the frequency of full cooperation in Figure 10 with r from 0 to 1 and plots of the average cooperation ratio in Figure 11. A close observation of Figure 10 and Figure 11 reveals that agents in the four-choice IPD population can maintain high levels of cooperation only for small r that less than 0.5. As r increases, the proportion of full cooperators drops regardless of the spatial structures. When the value of r is sufficiently high, defectors would dominate the agent population. For a four-choice IPD game with r >0.5, it is extremely challenging to evolve cooperative behavior. The average proportion of full cooperators is under 50% and becomes very low especially in the spatial structure is Von-Neumann and Moore network, as can be seen in Figure 10 and 11. On the other hand, when the value of r is small it would be worth to cooperate. For the values of $r < 0.5$, the population may settle into a mixed state where the cooperators resist the invasion of defectors as.

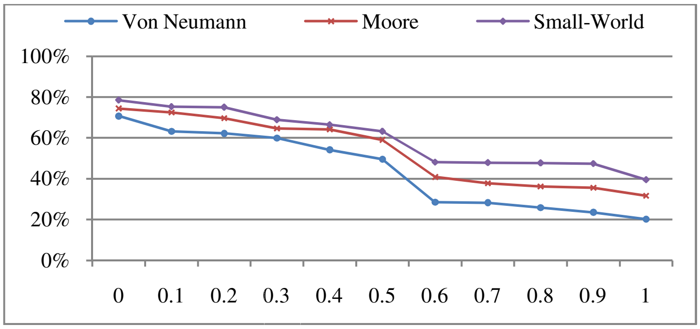


Fig. 10. The probability of full cooperation (+1) of *four*-choice IPD game in each spatial structure

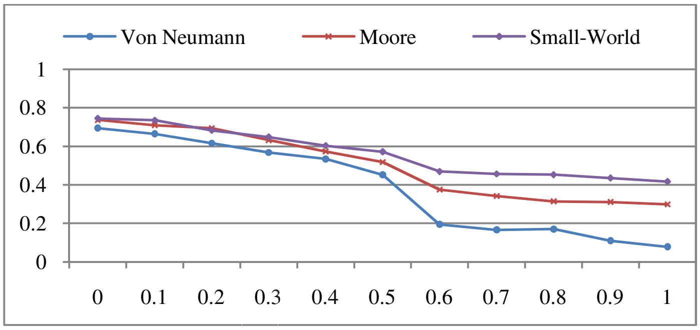


Fig. 11. The average cooperation of *four*-choice IPD game in each spatial structure

In small-world network environment, cooperators can lead to isolated clusters to defend the defectors. Even with a not very large re-wiring probability, it's possible that an agent could have all its links re-wired resulting in random longer range interactions and thus more general clustering in the population than other spatial structures. In other words, minor re-wiring in the population may be beneficial when the game becomes more challenging. The simulation results suggest that a small re-wiring probability could be advantageous when the evolution of cooperation is difficult.

5 Conclusion

In this paper, a new interpretation for n-choice IPD game in spatial environment is proposed. Detailed simulations across a range of spatial structures and cost-to-benefit ratios clearly showed that cooperation can be maintained, or sometimes even enhanced. The findings are consistent with previous studies along this line of research (e.g., Refs. [19, 23]). Significantly, this research has taken the current understanding of spatial evolutionary in IPD game one step further by considering the multiple choices game. This work may be helpful in understanding cooperative phenomena in systems where efficient collective actions are required.

Future work will include comparisons against existing game theory and IPD approaches, an investigation of the effects of using different payoff matrices to influence cooperative behaviour, competing against more neighbours of the spatial environment. As the theoretical study of the PSO techniques is extremely necessary and important, more detailed studies of the combination of evolved strategies and better PSO algorithms in the co-evolutionary training technique should be carried out.

Acknowledgement. This work is supported by two research programs of China. One is the National High Technology Research and Development Program of China (863 Program) under Grant no. 2012AA101701 and the other is the Key Projects in the National Science & Technology Pillar Program during the Twelfth Five-year Plan Period under Grant no. 2012BAK17B08.

References

1. Chong, S.Y., Yao, X.: Behavioral Diversity, Choices and Noise in the Iterated Prisoner's Dilemma. IEEE Transactions on Evolutionary Computation 9(6), 540–551 (2005)
2. Chong, S.Y., Yao, X.: Multiple Choices and Reputation in Multiagent Transactions. IEEE Transactions on Evolutionary Computation 11(6), 689–711 (2007)
3. Chong, S.Y., Tiño, P., Yao, X.: Measuring Generalization Performance in Coevolutionary Learning. IEEE Transactions on Evolutionary Computation 12(4), 479–505 (2008)
4. Chong, S.Y., Tiño, P., Yao, X.: Relationship Between Generalization and Diversityin Coevolutionary Learning. IEEE Transactions on Computational Intelligence and AI in Games 1(3), 214–232 (2009)
5. Chong, S.Y., Tiño, P., Ku, D.C., Yao, X.: Improving Generalization Performance in Co-Evolutionary Learning. IEEE Transactions on Evolutionary Computation 16(1), 70–85 (2012)
6. Ishibuchi, H., Takahashi, K., Hoshino, K., Maeda, J., Nojima, Y.: Effects of configuration of agents with different strategy representations on the evolution of cooperative behaviour in a spatial IPD game. In: IEEE Conference on Computational Intelligence and Games (2011)
7. Axelrod, R.: The evolution of cooperation. Basic Books, New York (1984)
8. Axelrod, R., Davis, L.: The evolution of strategies in the iterated prisoner's dilemma, Artificial Life II, pp. 32–41. Morgan Kaufmann, San Mateo (1987)
9. David, B.F.: On the relationship between the duration of an encounter and the evolution of cooperation in the iterated prisoner's dilemma. Evolution of Computation 3(3), 349–363 (1996)
10. Darwen, P.J., Yao, X.: Coevolution in iterated prisoner's dilemma with intermediate levels of cooperative: Application to missile defense. International Journal of Computational Intelligence and Applications 2(1), 83–107 (2002)
11. Darwen, P.J., Yao, X.: Why more choices cause less cooperation in iterated prisoner's dilemma. In: Proceedings of the Congress on Evolutionary Computation, pp. 987–994. IEEE Press, Seoul (2001)
12. Darwen, P.J., Yao, X.: Co-evolution in iterated prisoner's dilemma with intermediate levels of cooperative: Application to missile defense. International Journal of Computational Intelligence and Applications 2(1), 83–107 (2002)
13. Ishibuchi, H., Namikawa, N.: Evolution of iterated prisoner's dilemma game strategies in structured demes under random pairing in game playing. IEEE Transactions on Evolutionary Computation 9(6), 552–561 (2005)
14. Zheng, Y., Ma, L., Qian, I.: On the convergence analysis and parameter selection in particle swarm optimization. In: Processing of International Conference of Machine Learning Cybern., pp. 1802–1807 (2003)
15. Franken, N., Engelbrecht, A.P.: Comparing PSO structures to learn the game of checkers from zero knowledge. In: The 2003 Congress on Evolutionary Computation, pp. 234–241 (2003)
16. Franken, N., Engelbrecht, A.P.: Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma. IEEE Transactions on Evolutionary Computation 9(6), 562–579 (2005)
17. Di Chio, C., Di Chio, P., Giacobini, M.: An evolutionary game-theoretical approach to particle swarm optimization. In: Giacobini, M., et al. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 575–584. Springer, Heidelberg (2008)

18. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. IEEE International Conference of Neural Network, vol. 4, pp. 1942–1948 (1995)
19. Chen, S.Y., Wang, Z.J.: Acceleration Strategies in Generalized Belief Propagation. IEEE Transactions on Industrial Informatics 8(1), 41–48 (2012)
20. Bonaventura, M., Kaye, A., Rachel, B., Rebecca, G., Kate, H., Natasha, W.: Human friendship favours cooperation in the Iterated Prisoner's Dilemma. Behaviour 143(11), 1383–1395 (2006)
21. Moriyama, K.: Utility based Q-learning to facilitate cooperation in Prisoner's Dilemma games. Web Intelligence and Agent Systems: An International Journal 7, 233–242 (2009)
22. Chen, B., Zhang, B., Zhu, W.D.: Combined trust model based on evidence theory in iterated prisoner's dilemma game. International Journal of Systems Science 42(1), 63–80 (2011)
23. Chiong, R., Kirley, M.: Effects of Iterated Interactions in Multi-player Spatial Evolutionary Games. IEEE Transactions on Evolutionary Computation, doi:10.1109/TEVC.2011.2167682
24. Nowark, M.A.: Five rules of the evolution of cooperation. Science 314, 1560–1563 (2006)
25. Watts, D., Stogatz, S.H.: Collective dynamics of small-world networks. Nature 393, 440–442 (1998)
26. Chiong, R., Kirley, M.: Iterated N-Player Games on Small-World Networks. In: GECCO 2011 (2011)
27. Li, Y.X., Jin, X.G., Su, X.C., Kong, F.S., Peng, C.B.: Cooperation and charity in spatial public goods game under different strategy update rules. Physica A 389, 1090–1098 (2009)
28. Chiong, R., Kirley, M.: Co-evolutionary learning in the N-player iterated prisoner's dilemma with a structured environment. In: Korb, K., Randall, M., Hendtlass, T. (eds.) ACAL 2009. LNCS, vol. 5865, pp. 32–42. Springer, Heidelberg (2009)
29. Chiong, R., Kirley, M.: Imitation vs evolution: Analysing the effects of strategy update mechanisms in N-player social dilemmas. In: IEEE Congress on Evolutionary Computation 2010 (2010)
30. Nowark, M.A., May, R.M.: Evolutionary games and spatial chaos. Nature 359, 826–829 (1992)
31. Wang, X., Yi, Y., Chang, H., Lin, Y.: Co-evolutionary learning of N-choice IPD Game with PSO Algorithm. In: 4th World Congress on Nature and Biologically Inspired Computing, Mexico, pp. 92–97 (2012)

Extracting, Identifying and Visualisation of the Content, Users and Authors in Software Projects

Ivan Polášek and Marek Uhlár

Faculty of Informatics and Information Technology
Slovak University of Technology in Bratislava
Slovakia
{polasek,uhlar}@fiit.stuba.sk

Abstract. The paper proposes a method for extracting, identifying and visualisation of topics, code tiers, users and authors in software projects. In addition to standard information retrieval techniques, we use *AST* for source code and *WordNet* ontology to enrich document vectors extracted from parsed code, *LSI* to reduce its dimensionality and the swarm intelligence in the bee behaviour inspired algorithms to cluster documents contained in it. We extract topics from the identified clusters and visualise them in 3D graphs. Developers within and outside the teams can receive and utilize visualized information from the code and apply them to their projects. This new level of aggregated 3D visualization improves refactoring, source code reusing, implementing new features and exchanging knowledge.

Keywords: Software Project, Visualisation, Source Code, WordNet Ontology, Topic Identification and Extraction, Latent Semantic Indexing, Bee Behaviour Inspired Algorithms, AST, Swarm Intelligence, Authorship.

1 Introduction

This paper describes our approach to information and knowledge mining, which aims to support collaborative programming and helps software developers in medium and large teams to understand complicated code structures and extensive content as well as to identify source code authors and concrete people working with existing modules. Accordingly newcomers as well as other colleagues can reference to real source code authors and users more efficiently. The goal of our research is to provide a new perspective on software projects for participants to find and reuse particular parts of the source code. Subsequently applied graphs demonstrate the structure of the software project and the main topics contained in the source code in a natural way for them. This can help newcomers to quickly adapt to the project, but also senior developers can benefit from these methods. Determination of the source code topics can be used for purposes of identifying domain expertise of developers. It is also possible to support program comprehension by identifying common topics in source code. The contribution of our method is to find new level of aggregated visualization for the identified and interconnected information from the source code for collaborative development.

2 Related Works

Our approach is based on the combination of swarm intelligence, *LSI* (*Latent Semantic Indexing*) and the standard techniques of information retrieval for parsing software project.

We were inspired by method proposed by Kuhn et al., (2007). The steps of mentioned method are creation of term-document matrix, using *LSI* to reduce its dimensionality and afterwards aggregating it with standard hierarchical average-linkage clustering algorithm. *LSI* is used as inverted index to extract topics (top relevant terms) from identified clusters [1].

The results are displayed with correlation matrix (Fig. 1). It has documents on sides. The similarity between two documents d_i and d_j is $a_{(i,j)}$.

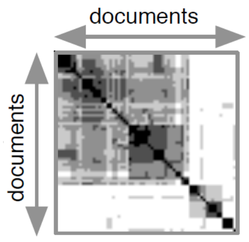


Fig. 1. Correlation Matrix [1]

In our work we try to evaluate the suitability of swarm intelligence bee inspired algorithms on text document clustering applied in many fields of research [2], [3], [4], [18]. We use *Flower Pollination Algorithm* (*FPA*) [5] and *Artificial Bee Colony Algorithm* (*ABC*) [6]. Main advantages of *FPA* in clustering are the inutility of input parameters such as number of clusters and initial cluster centers.

3 Our Approach

Based on the mentioned method [1] we created our own approach. We added term-document matrix enrichment via identifying relations between documents and *WordNet*. Our idea is to improve the topic identification and replace clustering with the basic hierarchical average-linkage by clustering algorithms inspired by swarm intelligence bee behaviour which are intensively researched nowadays and were proven suitable for clustering [6]. Also, we proposed visualisation for identified clusters (topics) as a coloured structure in software project.

Fig. 2 is the model of our approach, where simply all steps except *parsing*, *clustering* and *visualising* are optional and their purpose is to improve the results or speed of the process. The system is modular, so it is easy to replace either the clustering module or another one with a different component (using different algorithm) for further research.

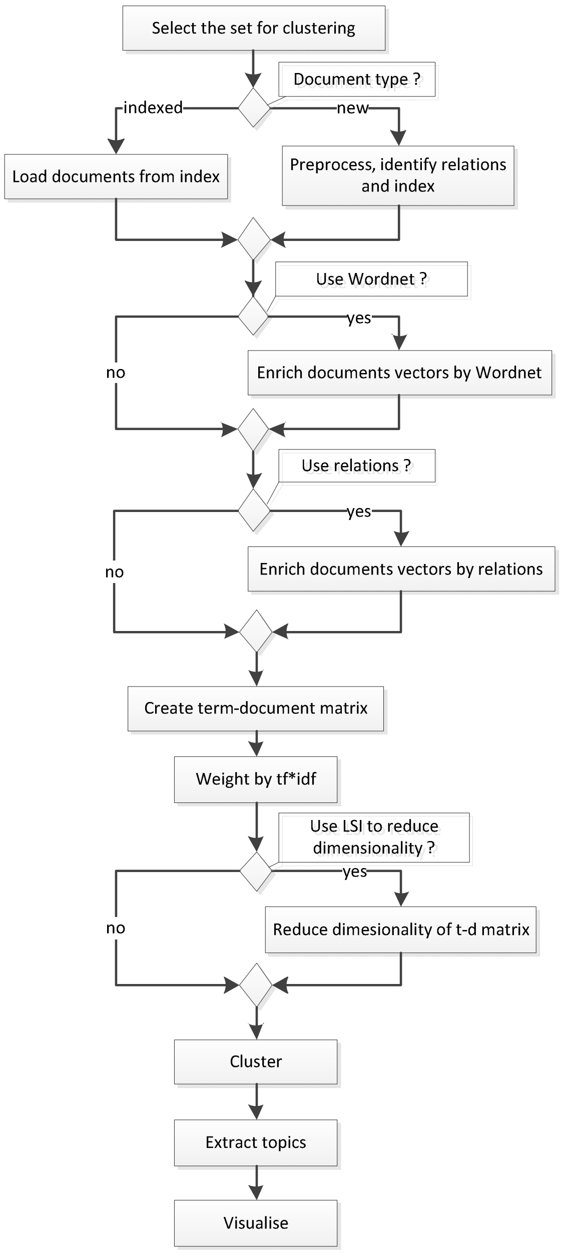

Select the set for clustering
Document type ?
indexed
new
Load documents from index
Preprocess, identify relations and index
Use Wordnet ?
yes
no
Enrich documents vectors by Wordnet
Use relations ?
yes
no
Enrich documents vectors by relations
Create term-document matrix
Weight by tf*idf
Use LSI to reduce dimensionality ?
yes
no
Reduce dimesionality of t-d matrix
Cluster
Extract topics
Visualise

Fig. 2. Method steps

First step of our method is parsing of the source code contained in software project. This can be done on three levels of granularity:

1. Source files - one document is equal to one source file. It can contain more classes.
2. Classes - one document is equal to one class or interface.
3. Methods - one document is equal to one method.

Currently, our prototype is able to work with projects created in .NET framework, C# language. We start by parsing the solution files (*.sln) which involves information about projects contained in solution. Afterwards, we parse individual project files (*.csproj) which contain information about individual files.

3.1 Parsing and Indexing

We index all extracted documents in the open source information retrieval software library *Lucene*. We use these pre-processing steps to reduce noise in the indexed documents:

1. Splitting identifier names by camel case naming convention. (CamelCase -> Camel Case)
2. Splitting identifier names by under score naming convention. (Under_Score -> Under Score)
3. Words with less than three characters are removed. (is, &&, ||)
4. All words are converted to lower case.
5. Removal of stop words – words that do not contribute to semantic meaning. (English stop words – and etc., C# keywords – private, static, while, switch).

These procedures contribute to better topic identification in particular phases of our method. Afterwards, we identify relations between indexed documents. We will use them in the second step of enriching the *term-document matrix*. For this task we use open source library *NRefactory*. Through it we are able to extract *AST* (*Abstract Syntax Tree*), from which we can identify relations between extracted documents.

After identification of relations we create term vector for each document (source file, class, method etc.). Finally, we create term-document matrix containing term frequencies in a particular document. In Fig. 3 we can see a simple example.

$$\begin{array}{rccccl} & d_1 & d_2 & d_3 & \ldots \\ \text{matrix} & 1 & 2 & 0 & \\ \text{database} & 0 & 7 & 6 & \bullet\ \bullet\ \bullet \\ \text{editor} & 1 & 1 & 1 & \\ \vdots & & \bullet & & \\ & & \bullet & & \\ & & \bullet & & \end{array}$$

Fig. 3. Term-document matrix example

We normalize the matrix with standard *tf*idf* algorithm with formula:

$$tfidf(d,t) = tf(d,t) * log_2\left(\frac{|D|}{1+df(t)}\right)$$

where ***d*** is a document from a collection we are working on, ***t*** is a concrete term from this collection and ***tf(d,t)*** is the frequency of the term *t* in the document *d*. **|*D*|** is the number of documents in our collection and ***df(t)*** is the number of documents in the collection which contains the term *t*.

3.2 Using Relations in Source Code to Enrich the Document Vectors

We use relations identified in the previous step to enrich the document term vectors. The type of identified relations depends on the selected granularity level we are working with (source file, class, method).

Source Files

On this level, we are enriching the term vector of each document (source file) in our collection with terms extracted from name of solution, project and eventually the directory it belongs to. Enrichment is done by adding the extracted terms to each document's term vector with particular weight.

We have estimated the weighting coefficient for these enrichments (Table 1) where ***L*** is *Label* and ***EV*** is *Estimated value*.

Table 1. Weighting coefficients

L	Description	EV
W_S	Terms extracted from solution. Solution also contains project and files with different topics. Terms are usually very general. → Weight coefficient is very low.	0,1
W_P	Terms extracted from project names are usually very close to the topic contained in the project's source files → we set the coefficient higher	0,5
W_D	Directory names are very specific but they can also be general and from the topic contained in the files (e.g. Tools) → the coefficient is not as high as for the project.	0,2

The adding is done after *tf*idf* weighting and it can enhance the identification of the topic contained in the documents and extend the vocabulary.

Classes and Interfaces

On this level, we use relations identified through *NRefactory* in the previous step. We are able to identify *aggregation*, *generalization*, *nested classes* and *partial classes*.

For partial classes we offer the possibility to work with them as one class. There is a switch in the prototype to turn this behaviour off.

In case of generalization we add parent class terms into child class term vector with coefficient 0.5. But we are also adding term in reverse direction (child -> parent), but with lower coefficient (0.2). Terms in child class could broaden the topic in parent class.

Furthermore, for aggregation we add terms from referenced class into a referring one. Terms contained in the name of referenced class could be not explanatory enough. We recommend very low coefficient (0.1).

Methods

We work with two types of relations on this level of granularity:

- Override – hiding of method from parent class with our own implementation.
- Overload – methods with the same name but with different parameters.

We are not estimating the indexes. They will be specified by further testing or manually by domain expert.

3.3 Using WordNet Ontology to Enrich the Term-Document Matrix

One of the problems in the semantic clustering of source code identified by [1] is the size of present vocabulary. It is considerably smaller than in the common text documents. As a consequence of this feature the topic extraction was not adequate in some cases.

The authors provide an example of false identification in the source file, in which the main topic was obviously ‘text editor’. However the extracted topic was out of the context because the vocabulary simply does not contain general terms ‘text’ and ‘editor’. The authors propose a solution of using a general English ontology (e.g. WordNet) to improve the results.

WordNet is the ontology (lexical database) containing English nouns, verbs, names, adjectives and adverbs organized in synsets. Synset is a group of semantically equivalent elements. The name used for synset in WordNet is concept. Concepts are interconnected by semantic relations (hypernyms, hyponyms, coordinated terms, holonyms, meronyms, etc.).

We use WordNet to enrich the term-document matrix. Our method is designed on the basis of research performed by [8]. Authors identify three steps to enrich the term-document matrix:

1. Mapping of terms to concepts.
2. Selecting a strategy how to join terms from matrix with identified concepts.
3. Selecting a strategy of choosing the concept.

Only noun terms to concepts are mapped in the current prototype. As the strategy of joining terms with concept we choose adding the concepts to terms. We want to extend the vocabulary as much as possible because of the problems described in the preceding text.

Currently we use functionality included in WordNet for selecting the concept. It is able to order the concepts found for particular term by internal ranking system which is mainly based on using frequency in English language. Concepts identified by this method can be completely out of context. Therefore we plan to implement the context based selection strategy. Steps involved in this strategy as described by [9]:

1. Define the semantic vicinity of the concept (all hypernyms and hyponyms up to level 1) as $V(c) = \{b \in C | c$ is hypernym or hyponym of b$\}$
2. Determine all terms, which could be a concept from vicinity of c by $U(c) = \bigcup_{b \in V(c)} Ref_c^{-1}(b)$.
3. Find most relevant concept on basis of context by $dis(d,t) = first\{c \in Ref_c(t) \mid c\ s\ max.\ tf(d, U(c))\}$.
4. Set the term weights by $cf(d,c) = tf(d, \{t \in T \mid dis(d,t) = c\})$.

Additionally, the identified concepts could be out of the context due to the generality of the WordNet. We plan to use domain ontology to overcome this obstacle in our further research.

3.4 Reducing Term-Document Matrix Dimensions

The vocabulary of software project is sparse but the dimensionality of term-document matrix especially in large projects can be high. Therefore, *LSI* (*Latent Semantic Indexing*) is used for possible improvement in topic identification and extraction [1]. This method is able to dramatically reduce term dimension of term-document matrix preserving the similarity between documents.

It is a huge advantage for clustering algorithm in the next step in terms of speed. Especially for bee inspired algorithms which we use and which are sensible to dimensionality of clustered data.

LSI is based on a mathematical method *SVD* (*Singular Value Decomposition*) originally used in signal processing to reduce noise preserving the original signal. In our case the noise in term-document matrix is synonyms and polysemy. So this technique not only reduces the dimensionality of the matrix, but also removes the noise occurring in natural language. It transforms set of documents and occurrences of terms to vector space. (Fig. 4)

The technique is really powerful. For example, during our testing we reduced the term dimensionality of matrix from 1580 to 77 keeping clustering results at almost the same quality. On the other hand, it is really resource consuming. In our testing it took almost 2 minutes to produce the reduced matrix for relatively small software project (180 classes - documents). Also, the matrix has to be recomputed every time any changes are done on the original matrix.

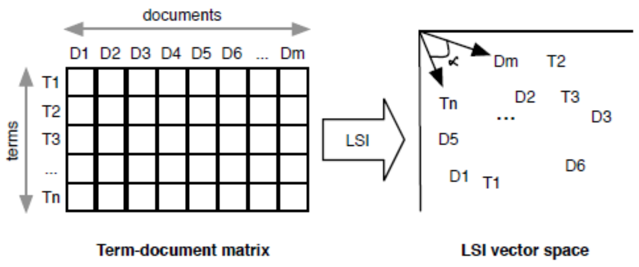


Fig. 4. LSI. [1]

Therefore, we implement this feature as optional in our prototype. It is possible to work completely without it.

3.5 Clustering the Term-Document Matrix with Bee Behaviour Inspired Algorithms

In the next step of our method we cluster the reduced matrix to extract the documents containing similar topics.

Originally, we started our research to explore the possible use of swarm intelligence bee behaviour inspired algorithms for source code or text document clustering in the area of software development. Consequently, we pick up two algorithms from this area and use them in our method.

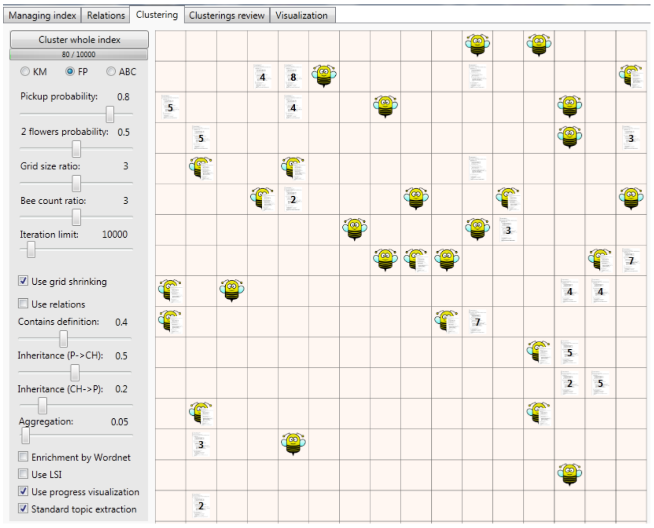
We chose *FPA* (*Flower pollination*) [5] (Fig. 5 shows visualisation of clustering progress in our laboratory tests) because it needs no input parameters such as number of expected clusters and initial centroids. Therefore we use it instead of the classical heuristic (m x n) / t used in text document clustering to identify the number of expected clusters. Also, it is able to cluster the dataset very well. But its results are not as good as some of the more advanced techniques for text clustering such as *SVM* or *C4.5* (Fig. 21).

We applied *ABC* (*Artificial Bee Colony*) algorithm [6] for optimization of clustering results. It needs initial inputs as a number of expected clusters and centroids and is very competitive in this domain.

We used the output of *FPA* (number of clusters, cluster centers) as the input for *ABC*. We modified *FPA* algorithm by replacing the Euclidean distance with cosine similarity metric for better results. Also, we proposed a new feature called “grid shrinking”. As the number of identified clusters is decreasing, we decrease the size of the grid. So the movement of bees on it is more effective. This feature is switchable and parameterizable via GUI.

3.6 Extracting Topics

After clustering we need to label the identified clusters by contained topics - the set of most relevant terms for a particular cluster.



KM [✓ FP] ABC
Pickup probability: 0.8
2 flowers probability: 0.5
Grid size ratio: 3
Bee count ratio: 3
Iteration limit: 10000
[✓] Use grid shrinking
[ ] Use relations
Contains definition: 0.4
Inheritance (P->CH): 0.5
Inheritance (CH->P): 0.2
Aggregation: 0.05
[ ] Enrichment by Wordnet
[ ] Use LSI
[✓] Use progress visualization
[✓] Standard topic extraction

Fig. 5. Visualisation of clustering via Flower Pollination Algorithm

If we have used *LSI*, then we can apply *LSI index* to retrieve the top-n list of the most relevant terms [1] and search the index by documents that are a part of the clusters and their result is a set of terms. In their case they take precisely top 7 terms.

In case we did not use *LSI*, we propose our own method of extracting top terms. We filter the documents in clusters to the ones having cosine similarity with centroid larger than 0.8 (it is in range 0-1). From this set we take the globally top 10 terms determined by *tf*idf* and use them as the topic label for the particular cluster. Alternatively, we can simply take the top n terms from each centroid cluster.

It is also possible to use one of the advanced techniques [7].

3.7 Visualising Results

The last step of our method is to visualise our results. Through previous steps, we have saved the information about indexed documents, relations between them, identified clusters and topics in the database. In this step, we visualise the information in a 3D graph (Fig. 6) with code name *Tent graph* to display the project structure in a uniform and well-arranged way.

The top node represents solution and subnodes are its particular projects. The leaves are classes and interfaces. The graph is implemented in modular way and it is able to show code authors and users, patterns and antipatterns in the code, finished and uncompleted modules, etc.

On granularity level of classes it displays identified clusters in a particular colour. Therefore, the classes and interfaces from particular cluster are tinted with the same colour. The intensity of it represents the similarity between the particular document and the centroid of its cluster. After clicking on the document all other documents from its cluster are highlighted and the topic (set of terms) is displayed in right top corner (Fig. 19). The user can comfortably click on the cluster and highlight only important classes (Fig. 19). Also the list of identified topics is present when no class is selected. Classes and interfaces for particular project are organized in distribution matrix.

With granularity level of methods we use the graph with the code name *Manhattan* (Fig. 20). You can access it by double clicking on a particular class/interface in the *Tent graph*. It displays all methods contained in a class like skyscrapers. The methods from particular cluster are tinted with the same colour. The intensity of it represents the similarity between the particular document and the centroid of its cluster. The height of the skyscraper represents the length of a particular method.

4 Additional Project Information

Currently we are working on expanding the capabilities of our methods to provide more comprehensive knowledge about the source code and the projects. This chapter contains methods for supplemental information [17] we recognize together with topic identification. We will observe and analyze the synergy of this methods and their information about users, authors, tiers and topics (e.g., which author has knowledge about particular expert domain, which author use *copy-paste technique*, which author has better pattern/antipattern ratio, etc.).

4.1 Tier Recognition

Tier recognition is used to automatically identify domains in three-tier based systems. It allows, for example, to estimate developer's orientation on a given tier or to simplify navigation in source code trough clustering.

In our approach we determine tiers in two ways.

Standard Types

Standard types are types in well-known frameworks that represent building blocks of many source codes. The aim of this method is to search for places in given source codes where these types are used to determine their tier using knowledge of relationships between used standard types and code tiers.

Table 2 presents a fragment of a lookup table that represents relationships of some standard .Net namespaces to three code tiers. The table is constructed manually in our work, but we are planning to use automated crawling techniques and clustering algorithms in the future.

Fig. 7 shows steps to determine tier association for given type declarations. First, used standard types are extracted and their namespaces are identified. Using a predefined lookup table, tier ratios of the extracted namespaces are determined.

If a namespace is not found in the lookup table, its parent namespace is searched for, and so on up to the root namespace. If not even the root namespace is found, the given namespace is ignored.

Table 2. A fragment of a lookup table, which maps .Net namespaces to tier ratios

Namespace	Presentation tier	Application tier	Data tier
System.ComponentModel	0.4	0.3	0.3
System.Data	0.05	0.05	0.9
System.DirectoryServices	0.15	0.7	0.15
System.Drawing	1.0	0.0	0.0
System.Globalization	0.6	0.1	0.3
System.IdentityModel	0.2	0.8	0.0
…	…	…	…

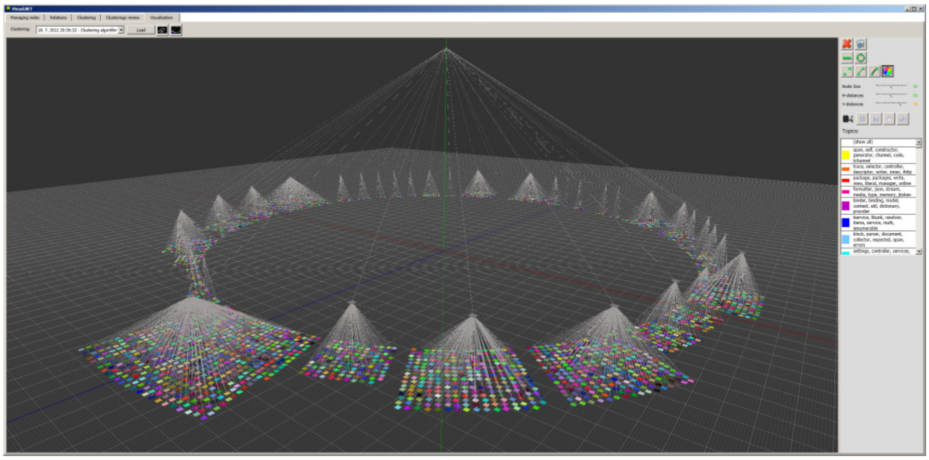


Fig. 6. Tent graph

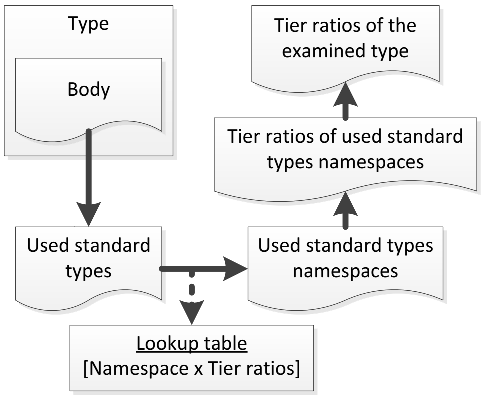

Type
Body
Tier ratios of the examined type
Tier ratios of used standard types namespaces
Used standard types
Used standard types namespaces
Lookup table
[Namespace x Tier ratios]

Fig. 7. Determining tiers by examining used standard types

In our approach we calculate the arithmetic average of all extracted ratios for each tier separately to get the final ratio. Fig. 8 shows an example result of this method. Tiers were determined for a fragment of a system, where mostly the presentation tier is present. Empty rows represent types, which were not determined, as they can belong to any tier.

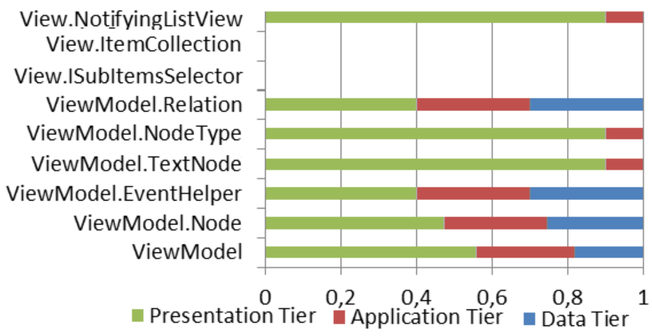


Fig. 8. Example - final ratios of examined types

Keywords
It is common that the name of a type in object-oriented source code describes its purpose. A type with the name "DbCommand", for example, suggests that it represents a database command. Our method uses this practice to identify known keywords in code identifiers to recognize the tier of a given type. Four identifiers are used for this purpose: *type name, base type name, namespace and project name*.

Keyword dictionary, which represents a set of known keywords and their tier assignments, is used as an input. Each tier assignment of a keyword is perceived as a rate, which defines how much the keyword is specific for a given tier. An example of a simple manually constructed dictionary is presented in the following table.

Table 3. Example keywords dictionary

Keyword	**Data**	**App**	**Presentation**
Data	0,90	0,10	0,00
Table	0,60	0,00	0,40
Workflow	0,10	0,80	0,10
Control	0,00	0,00	1,00
…	…	…	…

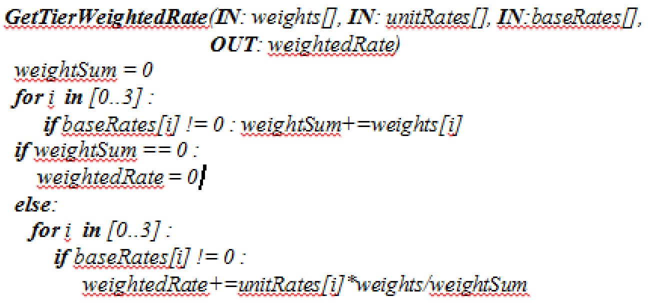
Identifier must be separated into words that can be compared with keywords in the dictionary. For example in case of interface "IDBTable" its name will be divided into "DB" and "Table", ignoring first letter "I" that is usually used to mark interface types. For this purpose, regular expressions that define the split points in the identifier's name are used.

After extracting the words from the identifiers, the partial association rate of each identifier type is computed. Each partial rate is in the range of <0.0, 1.0>. Words, which haven't been successfully matched with any keyword, are not included in the computation. Therefore they are not lowering the resulting rate. These partial rates are merged into final weighted rate. For this purpose for each identifier type a weight in the range of <0.0, 1.0> must be given. Example of these weights is presented in the following table.

Table 4. Example identifier type weights

Identifier	Weight
Name	1
Base Type Name	0.7
Namespace	0.6
Project Name	0.6

Pseudo code shown in the following figure represents computation of final weighted rate.



```
GetTierWeightedRate(IN: weights[], IN: unitRates[], IN:baseRates[],
                    OUT: weightedRate)
weightSum = 0
for i in [0..3] :
    if baseRates[i] != 0 : weightSum+=weights[i]
if weightSum == 0 :
    weightedRate = 0
else:
    for i in [0..3] :
        if baseRates[i] != 0 :
            weightedRate+=unitRates[i]*weights/weightSum
```

Fig. 9. Pseudo code of final wighted rate computation

Case Study

In this part we are going to show an application of this method on a very small set of types (Table 5).

Table 5. Example types

Type	BaseType	Project	Namespace
IGetBlob Helper		Frm. DataInterfaces	DataInterf. Blob
Graph3d Control	UserControl	Graph3d. WinForm	Graph3d. WinForm

Table 5. (*Continued*)

IWorkflow Helper		Frm. DataInterfaces	DataInterf. Workflow
IWorkflow Entity		Frm. DataInterfaces	DataInterf. Workflow
WSFrmData Context	DataContext	Frm. DataClasses	DataClasses
IDBHistorical Table		Frm. DataInterfaces	DataInterf. HistTable

The following charts display computed partial rates for each keyword type.

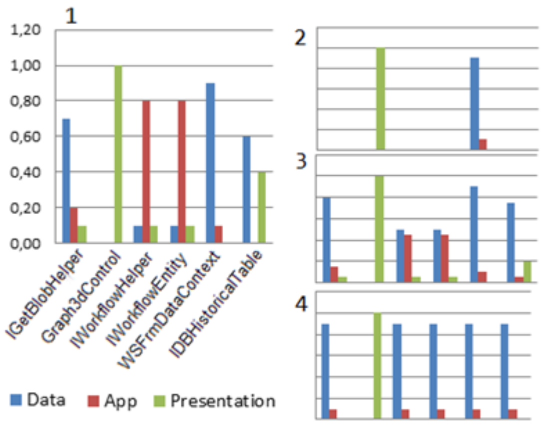


Fig. 10. Unit rate assignments (1-by name; 2-by base types; 3-by namespace; 4-by project)

These rates are then composed into the final result shown in the following figure.

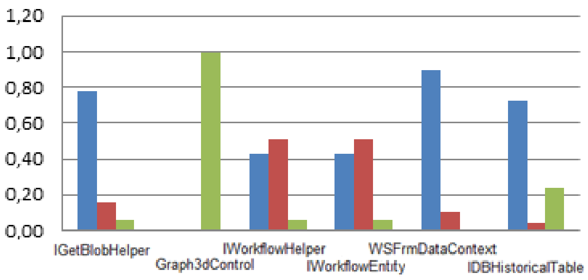


Fig. 11. Weighted assignment rates

4.2 Source Code Users Recognition

Our method collects information about users and their activities on code entities. Two methods, that gather and process different kinds of such information to increase collaboration in software development team is presented here.

Code Entities Checkout

In common revision control systems users are able to discover, which source code files are being modified by which users at the current time. This usually helps to avoid conflicting changes of multiple users in a single file.

We go deeper into files and look for concrete entities the source code is composed of. This enables us to determine, which specific parts of source code are being modified and lock for modification only those parts rather than whole files.

The following is the algorithm this method uses to determine currently edited source code entities:

```
GetChangedCodeEntities(OUT:changedCodeEntitySet)
  changedLocalFiles = GetChangedLocalFiles()
  for localFile in changedLocalFiles
     originalFile = DownloadOriginalVersionFromRCS(localFile)
     changedLineIndices = Compare(originalFile, localFile)
     originalFileAst = ExtractAst(originalFile)
       for lineIndex in changedLineIndices
           changedCodeEntity= originalFileAst.GetCodeEntityAt(lineIndex)
           changedCodeEntitySet.Add(changedCodeEntity)
```

User Activity on Source Code Entities

We monitor concrete activities that users perform on source code entities. Activities include modifying of a source code entity, pressing a mouse button over it, reaching it in the source code etc.

We measure the *intensity* of activities performed by each user. Each activity is assigned a value in the closed interval <0.0, 1.0>. 1.0 means the user is performing the activity intensively, while 0.0 represents inactivity.

Final Intensity of a Source Code Entity

A single user can perform multiple activities on a single source code entity at the same time. This includes different activities as well as the same activity performed at different places (e.g. in different code editors). Fig. 12 shows, how the final intensity of multiple activities performed by a single user on a single source code entity is calculated.

First, from all activities of the same type the one with the highest intensity is taken. This gives information, what different activities and how intense the user performs on the source code entity. By summing up all these partial activities the final intensity is calculated.

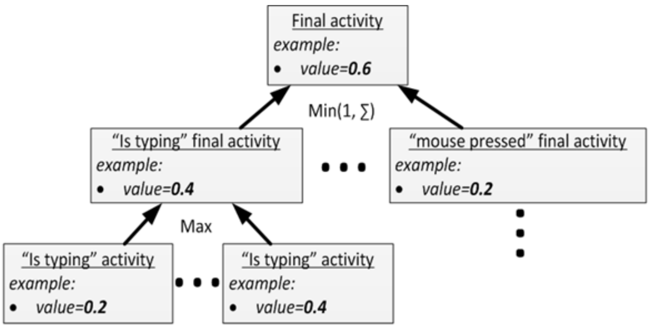


Fig. 12. The final intensity calculation of all activities performed by a single user on a single code entity

Activity Initial Intensity and Cooling of Activities
Each type of activity has a predefined maximum and minimum intensity. When an activity occurs, its intensity is maximal. When it is not performed for a period of time, its intensity starts to decrease down to the minimal value. We call this process *cooling of activities* and it expresses the decreasing interest of the user for the source code entity.

Case Study
Fig. 13 shows a prototype we implemented. Three activities are monitored for a single user:

- In viewport – the code entity is reached in a code editor (scrolling etc.)
- Mouse down – the user pressed the mouse over the code entity
- Is typing – the user types into the code entity

The figure also shows how activities are being cooled down.

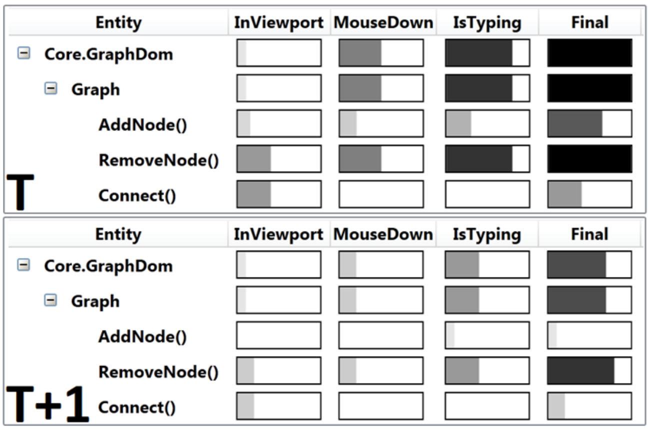


Fig. 13. Activities for a single user. Activities are being cooled down.

4.3 Source Code Authors Recognition

Introduction

To evaluate programmer's coding skills and quality of his work, we usually review some set of functional units - code entities and apply selected metrics. In collaborative environment many authors participate in source code development, so we must distinguish particular author contributions on given code entities. Everyone, who changes the content of a source code file, can be an author of contained source code. In our approach, we divide authors into three basic groups:

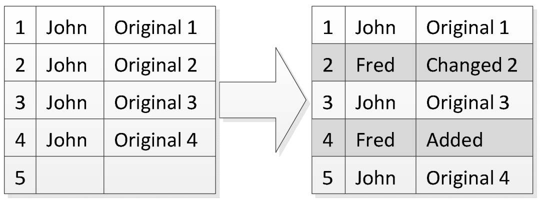
- real authors, who modify logical nature of source code (add, modify or delete code entities)
- editors, who modify form of a source code, but not its logical meaning (they are refactoring, sorting entities, formatting code, …)
- reviewers, who comment code or update code due the newer version of used libraries

Authors, members of a development team, share and contribute to source code via Revision Control System (RCS). Author's contribution is defined by a set of changes in one or more source code versions. Change information should cover: author, commit ID, date of change and type of change (add, edit, delete, move...).

There are code entity authors, whose source code parts have persisted to latest or particular source code version and authors, whose source code parts were deleted or considerably modified by the time. In code evolution every version is based on previous versions to certain degree. Ideas represented in older versions are kept, transformed or reused in newer versions. Therefore, it is a matter of principle to assess source code in its whole history, not only in resultant form. For proper authorship metrics, we must consider both persisted and perished changes.

Difference Built AST

Our intention for proper authorship evaluation comes from the use of a Microsoft Team Foundation Server (TFS) Client extension for Visual Studio called *Annotate* [10]. Annotate downloads every version of a particular file and annotates the output with attributes showing the changeset, date, and user who last changed each line in the file. The implementation does not show deleted lines [11]. We can say it works on principle of overlapping author information for individual lines of code using line oriented changes from first to latest source code file version (Fig. 14).



1	John	Original 1
2	John	Original 2
3	John	Original 3
4	John	Original 4
5		

1	John	Original 1
2	Fred	Changed 2
3	John	Original 3
4	Fred	Added
5	John	Original 4

Fig. 14. Principle of Annotate function

Annotate is based on file and line representation of source code versions/changes in TFS. It is a fast method, but it takes into account only changes, which persisted to the latest version; therefore, it is inaccurate for our authorship metrics. Another disadvantage is the fact, that it ignores deleted lines. Furthermore, this solution does not solve similarity of source code parts, thus moving a part of code is interpreted as deleting lines in one place and adding lines in another place. This also deforms authorship information.

To address the disadvantages of Annotate (or code line representation of changes in general), we created a solution, where, for every version of source code, line changes are projected to code entities. This information is joined to histories of code entity changes. This approach determines authorship of source code entities in object oriented paradigm, where syntax units can be represented as nodes in an Abstract Syntax Tree (AST). It is based on extraction of source code entity changes. The extraction can be divided into several phases (Fig. 15):

1. Extraction of source code files from a software solution. Files can be added into, moved within or deleted from the software solution. It is important to identify all source code files contained in the solution throughout its history.
2. Extraction of source code from all files valid for given versions. This is done repetitively for each two adjacent versions.
3. Representation of each two adjacent versions as ASTs using appropriate software tools [12]. Transformation to AST representation is restricted to level of meaningful syntactical units (classes, functions, properties). Lower level content is represented as lines.
4. Comparison of selected source code versions using *Diff* function based on solving the longest common subsequence problem [13]. Output of *Diff* function can be represented as a set of code changes, which can be of type Add, Delete or Modify. Each change holds information about the author and the range of affected lines – add in newer version of source code, delete in older and modify in both versions.
5. Mapping of code differences to code entities and representation of source code entity changes. In this phase, we create source code entity changes, which are relations between source code elements and source code changes. Code change falls to code entity, if intersection of their ranges is not an empty set. One change can fall to:

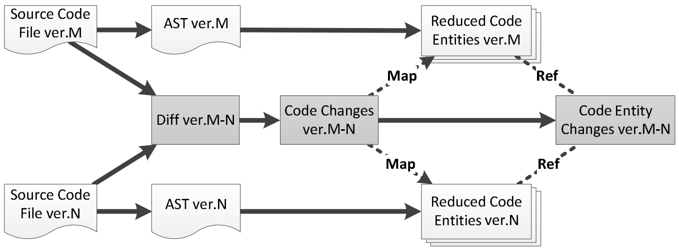
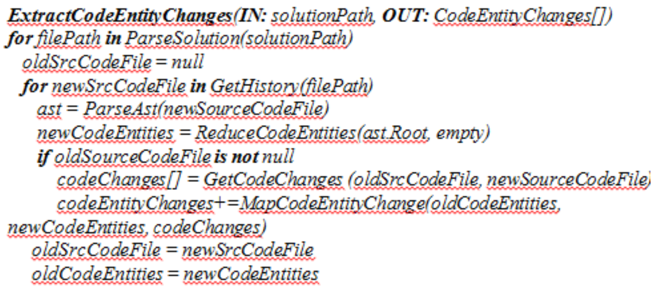


Fig. 15. Determination of authorship is based on extraction of source code entity changes, divided to several phases

6. one code entity in new, old or both compared versions of source code
7. to multiple code entities in one level of AST (for example two functions)
8. to multiple code entities in multiple levels of AST (method, class, namespace)
9. One code change can produce many code entity changes, so it is important to create relations (between change information unit, old and new version of a code entity) only between syntax units of the same type and on the same level of AST.

The following pseudo code represents presented extraction process:



```
ExtractCodeEntityChanges(IN: solutionPath, OUT: CodeEntityChanges[])
for filePath in ParseSolution(solutionPath)
  oldSrcCodeFile = null
  for newSrcCodeFile in GetHistory(filePath)
    ast = ParseAst(newSourceCodeFile)
    newCodeEntities = ReduceCodeEntities(ast.Root, empty)
    if oldSourceCodeFile is not null
      codeChanges[] = GetCodeChanges (oldSrcCodeFile, newSourceCodeFile)
      codeEntityChanges+=MapCodeEntityChange(oldCodeEntities,
newCodeEntities, codeChanges)
    oldSrcCodeFile = newSrcCodeFile
    oldCodeEntities = newCodeEntities
```

Fig. 16. Extraction of source code entity changes pseudocode

The result of this process is represented by sets of changes attached to corresponding code entities. For each code entity, its change history can be constructed. Using this information we can construct relations between versions of abstract syntax trees (Fig. 17).

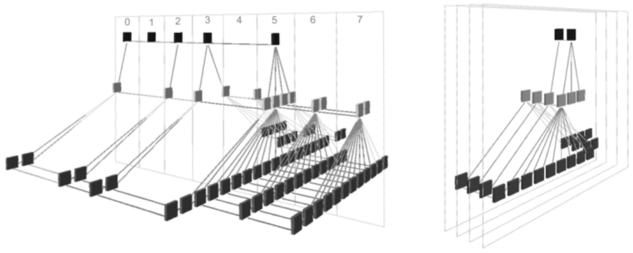


Fig. 17. Relations between versions of abstract syntax trees

The author of each change can be determined and the authorship of each entity can be evaluated and visualized (Fig. 18).

The extracted information provides variables for various metrics. The variables are:

- author
- entity
- type of change
- line range of change

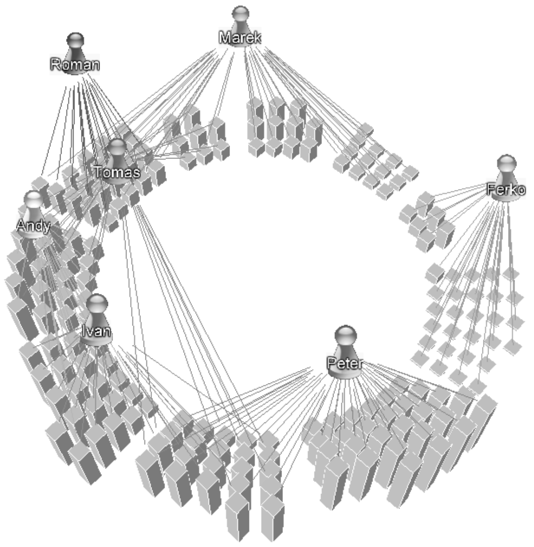


Fig. 18. Authorship of source code entities presented in *Tent graph*

Also this process has some drawbacks, which can affect the accuracy of metrics results. The most significant are:

- Change in the order of code entities at one AST level. This change is represented by *Diff* as removed and added entities. We would need to compare the similarity to other entities, rather than just using only the information describing the change.
- Renaming of entities. In this case, a relation between entities with different identifiers (name of entity and its placement in AST) is changed. Using only the extracted information, it is not possible to determine, whether the change represents a rename of an entity or it is caused by some other change overlapping neighbour entities.
- Renaming of the identifier of a superior entity. This change is represented as a change of a superior entity. Child entities are not changed. In the process of history construction, entity versions cannot be mapped using their name and placement in the AST. This scenario is interpreted as extinction of the original entity and creation of a new entity. Information concerning the similarity of entities would also be needed.
- A multiline change overlapping multiple entities at single AST level. For example, if two adjacent entities are affected by a single change, four change information units are created. Each change information unit relates to different combination of entities (two original and two new versions). An entity version graph is created in the process of entity history construction. Therefore, it is not clear, which graph path represents the entity history. Also here, the information concerning the similarity of entities would be required.

From previous, we can sum up, that the main problem in the authorship evaluation is the following. An author is defined by changes, that he made in some version and a part of source code. But if we want to evaluate authorship in whole history of code entity, we must match code entities between several versions of source code, which is not trivial due to change of identifiers of code entities and changes of entities positions in AST [14]. Identification of source code entities is given by their similarities or matching [15].

Conclusion and Future Work for Author Recognition

Deficiencies in previous solutions led to design of a RCS based on the AST representation. AST RCS does not substitute RCS used to manage source code versions in daily collaborative development. AST RCS is a supplement of RCS. It imports file and line oriented data and transforms them to historical AST, containing linked code entity versions, using principles similar to [14] and [15], eliminating all deficiencies presented previously. This platform offers unified representation of data imported from several RCSs.

AST RCS is also used for source code marking, where source code parts are labeled with metadata. These metadata – marks – can, among others, contain information about source code quality – effectivity of code, use of code patterns, occurrence of design patterns, smells, etc. This way, we can not only use quantitative, but also qualitative variables in authorship metrics to asses developer's contributions and effects, these contributions have to source code quality. By code marking, we can also label concepts reused in source code development process, identified using methods as presented in [16]. Besides detection of identical source code reusing (copy), source code parts move or refactoring, provided by AST RCS platform, this allows to identify application of concepts or ideas of other authors.

Application of authorship metrics in the context of source code entities by using this approach should be much more accurate and resistant then code line based metrics to specific, but frequently occurring source code changes presented in this paper. The creation of qualitative authorship metrics is matter of metadata type available and can contribute to construction of user profiles (developer profiles) in the wider context of our next research.

5 Prototype MEAD.NET for Topics Recognition

We implemented prototype in .NET framework, C# language and named it Mead.NET. It consists of four screens:

1. Managing index – with adding and viewing indexed documents.
2. Relations – with view of identified relations.
3. Clustering screen – to adjust parameters and view the process of clustering.

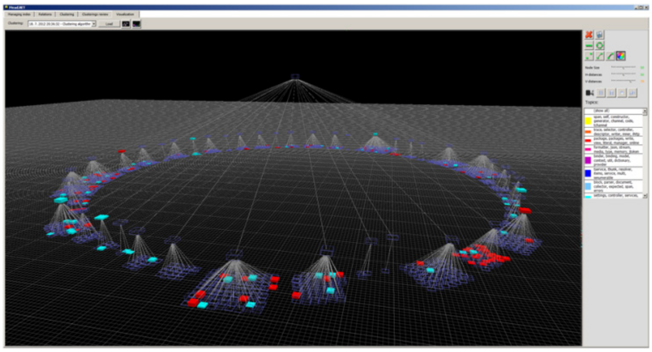


Fig. 19. Tent graph with selected topics

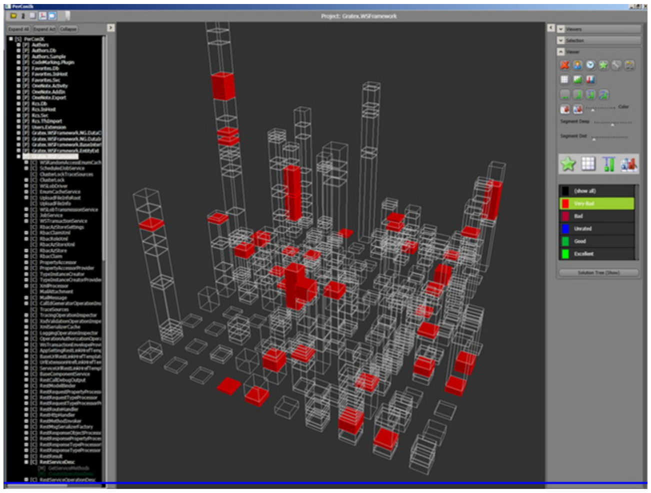


Fig. 20. Manhattan graph with selected topics

4. Visualisation – to view the graph for a particular clustering (Fig. 6, Fig. 19 and Fig. 20).

6 Case Studies, Evaluation, Further Research

The main problem with the testing of our approach is the non-existence of a sample dataset. We propose the creation of a dataset containing a project ideally clustered and labeled with topics.

Our goal is to show that our changes and add-ons to proven and useful method [1] tend to better results. We have done it in two steps.

First of all, one of the crucial parts of this method is clustering. So we tested used unsupervised clustering algorithm (*FPA*) together with *LSI* in comparison with some supervised algorithms (*SVM* and *C4.5*).

We used *Tech-TC100* dataset. We compared the purity of the clusters identified by mentioned algorithms. The results (Fig. 21) were positive having in mind the future use of *ABC* algorithm which promises even better results.

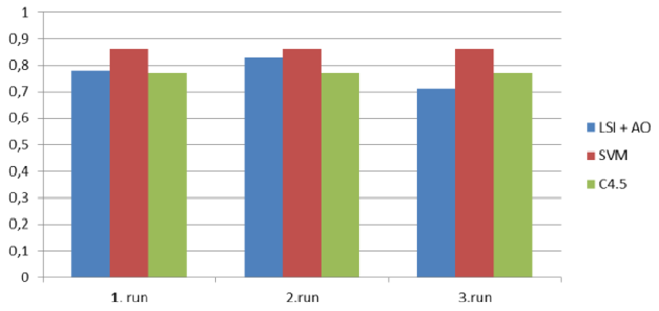


Fig. 21. Clustering results

Afterwards, we tested the whole process of our method on a wide range of software projects. The results are in Table 6. The method is applicable on all of the tested projects with promising results, but we deal with time consumption methods in complex projects and huge applications.

The tested projects are of varying size and complexity. The tests were performed with the following configuration:

- Size of the side of the grid (gs) is specified by formula $gs = \sqrt{10.(Number\ of\ documents)}$
- Bee count is equal to grid side size.
- We used enrichment via relations. The enrichment via Wordnet ontology was turned off.
- Each project was tested with 100, 1000 and 10000 clustering iterations.

We used research (Mead.NET, PerConIk), commercial (Asp.NET MVC, Robot Emil) and open-source (Quartz.net, IronPython) projects. In the case of research projects some of the identified clusters were analyzed by developers of the particular project. The results are promising. Most of the clusters include the documents within similar topic. Also the extracted top terms represent the topic which is contained in them. Few exceptions mostly due to the generality of Wordnet were identified. Consequently, the enrichment via Wordnet was disabled for further testing.

Every clustering provides slightly different view on the software project. The main parameter is the number of iterations which directly influences the granularity and the purity of clusters.

Specialized datasets in the form of categorized structural parts of the software project on different levels of granularity will be needed to thoroughly test the impact of proposed approaches. The categories will be topics identified in the particular software project.

For daily use application we propose two approaches. First approach (Fig. 22) consists of the following steps:

1. Use method presented in this paper to cluster software project and extract topics. Use of *FPA* algorithm. Number of iterations depends on size of the project.
2. If the software developer adds new classes or methods, system can insert them randomly to the grid, which is the result from the first step. Then, it can cluster the set for a reasonable number of iterations. It will take only milliseconds and the user will not wait for the latest results.
3. In the last step we apply optimization of existing clustering results through *ABC* in the next parallel thread until the next change in project.

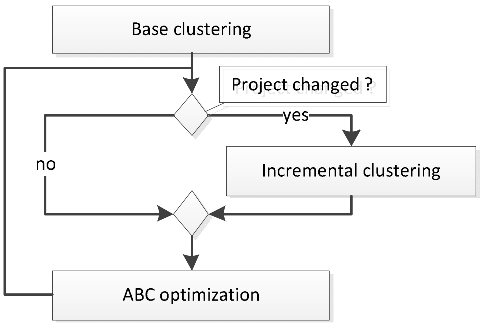


Fig. 22. Incremental clustering approach

Second approach (Fig. 23) is designed with performance in mind. It consists of the following steps:

1. Use *FPA* to cluster and extract topics as in the first approach.
2. Add new classes or methods to existing clusters based on the similarity with cluster centers. It will take milliseconds and minimize the time consumption for clustering.
3. Optimization via ABC until the next change in the project.

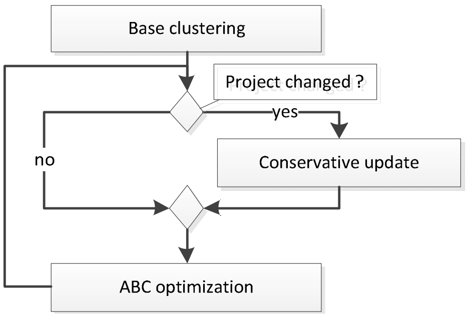


Fig. 23. Update approach

7 Future Work

Our primary goal is to create a dataset suitable for testing the extraction of topics from the source code. The idea is to manually label classes with topics in some open source project of sufficient size (cca. 300 classes). With this dataset we will be able to thoroughly test the contribution of term-document matrix enrichment in topic extraction.

As we mentioned before WordNet is not ideal for our particular field. Therefore, our plan is to extract our own ontology from software projects, *MSDN* (*Microsoft Developer Network*) and *SDN* (*Sun Developer Network*) which could be a lot more suitable.

Also, implementing time optimization (*Incremental clustering* and *Update approaches*) for daily use and another term extraction algorithms in our prototype can contribute to better results. Furthermore, we would like to add a module for parsing projects written in Java to test the suitability of our method in a different programming language.

Table 6. Evaluation on real-world projects

Project	summary counts					optimal algorithm settings (*)		iterations	duration of		found	result
	prj/srcFiles	code lines	classes	interfaces	methods	Bee Count	Cluster Cells		indexing	clustering	clusters	shrink
Mead.Net	10/88	15 000	**107**	3	442	32	1024 (32x32)	100	0:08	0:05	54	45%
								1000	0:08	0:45	24	22%
								10 000	0:08	8:05	11	10%
PerConIK	45/466	81 200	**757**	46	2 720	87	7569 (87 x 87)	100	0:18	0:41	423	56%
								1000	0:18	4:32	221	29%
								10 000	0:18	41:00	101	13%
PerConIK+WsFm	52/650	212 000	**1923**	(?)	7 072	138	19044 (138 x138)	100	0:32	2:30	1507	78%
								1000	0:32	24:10	830	43%
								10 000	0:32	(4:00:00)	(400 ?)	(20%)
GKO.RIMIS	7/87	19 000	**141**	6	255	37	1369 (37 x37)	100	0:10	0:02	79	56%
								1000	0:10	0:18	42	29%
								10 000	0:10	2:30	22	15%
GKO	17/705	230 000	**1021**	158	6 679	100	10000 (100x100)	100	0:25	2:00	679	66%
								1000	0:25	14:45	371	36%
								10 000	0:25	1:40	151	14%
Quartz.Net	4/299	65 000	**323**	46	2 221	56	3136 (56x56)	100	0:24	0:10	165	51%
								1000	0:24	1:10	91	28%
								10 000	0:24	9:20	35	10%
Iron Python 4	14/952	254 000	**1906**	76	14 316	138	19044 (138 x 138)	100	1:40	4:15	1213	63%
								1000	1:40	35:36	673	35%
								10 000	1:40	5:40	269	14%
ASP.Net MVC	41/2482	342 000	**3207**	123	17 310	179	32041 (179 x179)	100	4:00	13:40	2001	62%
								1000	4:00	1:50:00	1098	36%
								10 000	4:00	(18:20:00)	(500?)	(15%)
Robot Emil	1/6	8 300	7	0	**145**	38	1444 (38 x 38)	100	0:05	0:05	50	33%
								1000	0:05	0:35	31	21%
								10 000	0:05	1:50	13	8%

8 Conclusions

We proposed a method based on previous research [1], and enhancement and combination of methods. We designed changes such as different clustering algorithms, optionality of *LSI*, more advanced parsing, and relation identification. Also, we proposed add-ons to this method such as *WordNet* term-document enrichment and new results visualisation tool.

We implemented the prototype and tested usefulness of our contributions. Our future work is to optimize (*Incremental clustering* and *Update approach*) and prepare the method for daily use, advance ABC clustering algorithm, prepare our own ontology and new dataset for the testing of our approach.

Acknowledgment. Part of this work was published in the Proceedings of the Fourth World Congress on Nature and Biologically Inspired Computing (NABIC 2012) [19]. This work was partially supported by the grants VG1/0971/11, VG1/1221/12, and it is the partial result of the Research & Development Operational Programme for the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF.

References

1. Kuhn, A., Ducasse, S., Girba, T.: Semantic clustering: Identifying topics in source code. Information and Software Technology 49(3), 230–243 (2007) ISSN 0950-5849
2. Karaboga, D., Bahriye, A.: A survey: algorithms simulating bee swarm intelligence. Artificial Intelligence Review 31(1), 61–85 (2010)
3. Návrat, P., et al.: The Bee Hive At Work: Exploring its Searching and Optimizing Potential. INFOCOMP Journal of Computer Science 11(1), 32–40 (2012) ISSN 1807-4545
4. Rajasekhar, A., et al.: A Hybrid Differential Artificial Bee Algorithm based tuning of fractional order controller for PMSM drive. In: Proceedings of the Third World Congress on Nature and Biologically Inspired Computing (NABIC 2011), pp. 1–6. IEEE (2011) ISBN 978-1-4577-1122-0
5. Kazemian, M., Ramezani, Y., Lucas, C., Moshiri, B.: Swarm Clustering Based on Flowers Pollination by Artificial Bees. In: Abraham, A., Grosan, C., Ramos, V. (eds.) Proceedings of Swarm Intelligence in Data Mining. SCI, vol. 34, pp. 191–202. Springer, Heidelberg (2006)
6. Karaboga, D., Ozturk, C.: A novel clustering approach: Artificial Bee Colony (ABC) algorithm. Applied Soft Computing 11(1), 652–657 (2011)
7. Carmel, D., Roitman, H., Zwerdling, N.: Enhancing cluster labeling using Wikipedia. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009 (2009) ISBN 978-1-60558-483-6
8. Amine, A., Elberrichi, Z., Simonet, M.: Evaluation of text clustering methods using wordnet. Int. Arab J. Inf. Technol., 349–357 (2010)
9. Hoth, A., Staab, S., Stumme, G.: Wordnet improves Text Document Clustering. In: Proceedings of the SIGIR 2003 Semantic Web Workshop, pp. 541–544 (2003)
10. View File Changes Using Annotate. Team Foundation Server 2010, MSDN, Microsoft, `http://msdn.microsoft.com/en-us/library/bb385979.aspx` (accessed September 25, 2012)
11. Hodges, B.: Annotate (also known as blame) is now a power toy. MSDN Blogs, Microsoft, `http://blogs.msdn.com/b/buckh/archive/2006/03/13/annotate.aspx` (accessed September 25, 2012)
12. Grunwald, D.: NRefactory. SharpDevelop, `http://wiki.sharpdevelop.net/NRefactory.ashx` (accessed September 25, 2012)
13. Eppstein, D.: Longest Common Subsequences. ICS 161: Design and Analysis of Algorithms Lecture notes (February 29, 1996), `http://www.ics.uci.edu/~eppstein/161/960229.html` (accessed September 25, 2012)

14. Fluri, B., Wursch, M., Pinzger, M., Gall, H.: Change Distilling: Tree Differencing for Fine-Grained Source Code Change Extraction. IEEE Transactions on Software Engineering 33(11) (2007) ISSN 725-743
15. Neamtiu, I., Foster, J.S., Hicks, M.: Understanding source code evolution using abstract syntax tree matching. In: Mining Software Repositories (MSR 2005), pp. 1–5. ACM, New York (2005) ISBN 1-59593-123-6
16. Marcus, A., Sergeyev, A., Rajlich, V., Maletic, J.I.: An Information Retrieval Approach to Concept Location in Source Code. In: Proceedings of the 11th Working Conference on Reverse Engineering, WCRE 2004 (2004) ISSN 1095-1350
17. Polášek, I., Ruttkay-Nedecký, I., Ruttkay-Nedecký, P., Tóth, T., Černík, A., Dušek, P.: Information and Knowledge Retrieval within Software Projects and their Graphical Representation for Collaborative Programming. In: Acta Polytechnica Hungarica, vol. 10(2), Óbuda University (2013) ISSN 1785-8860
18. Navrat, P., Sabo, S.: What's going on out there right now? A beehive based machine to give snapshot of the ongoing stories on the Web. In: Proceedings of the Fourth World Congress on Nature and Biologically Inspired Computing (NABIC 2012), pp. 168–174. IEEE (2012) ISBN 978-1-4673-4767-9
19. Uhlár, M., Polášek, I.: Extracting, identifiyng and visualisation of the content in software projects. In: Proceedings of the Fourth World Congress on Nature and Biologically Inspired Computing (NABIC 2012), pp. 72–78. IEEE (2012) ISBN 978-1-4673-4768-6

Beehive Based Machine to Give Snapshot of the Ongoing Stories on the Web

Pavol Návrat[1] and Štefan Sabo[2]

Faculty of Informatics and Information Technologies
Slovak University of Technology,
Bratislava, Slovakia
{navrat,sabo}@fiit.stuba.sk

Abstract. In this paper we present an approach, inspired by honey bees, that allows us to take a glance at current events by exploring a portion of the Web and extracting keywords, relevant to current news stories. Not unlike the bees, that cooperate together to retrieve little bits of food, our approach uses agents to select random keywords and carry them from one article to another, landing only on the articles relevant to the keyword. Keywords that best represent multiple articles are selected, while keywords not relevant to articles are subsequently discarded and not explored further. Our results show, that with this approach, it is possible to extract keywords relevant to news stories, without utilizing learning methods, or analysis of a data corpus.

1 Introduction

With the rising prominence of Web as a primary source of current news and event coverage, the potential to attract a reader depends on the ability to convey information quickly in a condensed form. This way, the reader can decide whether he is interested in further reading by glancing at the title of an article. When multiple articles deal with the same event, or are otherwise interconnected, popular labels are used, that make it easier for the reader to identify the general topic of the article. An example of such labels would be"*Fukushima incident*" or "*EU crisis*".

In this paper we refer to these popular labels as to "*keywords*", however it is important to note, that in addition to describing the content of the article itself, the keyword in context of this paper also needs to imply relevance of the given article to a set of articles, comprising a story. We use this expanded notion of keywords relevant to a certain story, because we consider these keywords to be crucial for the reader navigating the space of news articles on the Web, as they direct the reader to the articles interesting to him.

Our main goal is to provide the means of automatic on-the-fly extraction of these story relevant keywords from a set of articles without the need of preprocessing or supervised learning. In order to achieve this, we are utilizing a multi-agent system of web crawlers to download news articles from the Web and to identify the desired keywords, that represent news stories being covered in the available set of articles.

Organization of our web crawler system is based on Beehive Metaphor [12], which models the organization observed within colonies of honey bees when foraging for food sources. Agents move between articles and look for suitable keywords representing a news story, similarly to honey bees navigating their surroundings in search for suitable food source. The keywords that best represent the articles comprising a story are subsequently propagated and explored further, while the keywords of local significance only are discarded over time.

2 Related Work

In this section we outline the related work in two fields. First we concentrate on different approaches to keyword extraction, mostly focused on the domain of news articles, as this is also the domain of our proposed approach. Secondly we present related work on Beehive Metaphor, relevant to our research.

2.1 Keyword Extraction

Most of current keyword extraction techniques require either a data collection, or a supervised learning approach, to determine the weights of candidate keywords and select the best candidates to represent the document. Lee and Kim [6] propose an approach based on *term frequency–inverse document frequency* weighting model to extract keywords from news articles. This approach is adapted to extract keywords for a pre-selected list of article domains, which is similar to our goal of extracting keywords relevant to news stories. Kriz [4] attempted to base keyword extraction on implicit feedback.

Toda et al. [19] present an approach to extraction of topical structure, but instead of keywords, they utilize temporal information present in documents to asses mutual relevance of documents. Other way of determining the topical structure of an article set is to use clustering algorithms [8,18]. All of these approaches require the whole dataset to be available in order to determine the covered topics. Vadrevu et. al [21] offer a solution by introducing a system with incremental clustering ability. This incremental clustering requires an offline clustering to be performed first. Subsequently it uses results of offline clustering as a ground truth for construction of incremental classifier.

Wang and Kitsuregawa [22] propose a method of search results clustering based on link analysis. The aim of this method is to cluster together results from pages, that share common links. Bohne et al. [1] propose a method of automated keyword extraction by utilizing Helmholtz principle. Every document is divided into smaller windows and the occurrence of terms is compared with a random baseline. Keywords are chosen according to their deviations from the baseline. This approach allows for keyword selection without supervised training, or general analysis of dataset. However, it requires individual terms to be statistically independent and as such is not well suited for use with news articles. Clustering on social web was attempted by Kuzar [5].

Other way of approaching news topic recognition is to determine the topic of articles by utilizing supervised learning. KEA algorithm [23] extracts keywords from a set of documents by utilizing naive Bayes classifier. Training set is composed from documents with already established keywords. A comprehensive overview of supervised learning clustering algorithms is offered by [20].

2.2 Social Insect

The gathering of documents from the web and subsequent potential keyword evaluation is performed by a group of agents, inspired by behaviour of honey bees. The pattern of communication is described by the beehive metaphor model, introduced in [12] as an enhancement of a previous work by Lorenzi [9]. Beehive metaphor itself has been studied from various angles. Lemmens et al. [7] or Memari et al.[10] approached it within the concept of multi-agent systems. The idea of a collaborative foraging approach to web browsing was presented by Schultze [16]. Rambharose and Nikov [15] applied a similar concept to personalization of interactive web systems. For excellent surveys, cf. e.g. Karaboga [3] and Teodorovic[17]. Related *Artificial Bee Colony* (*ABC*) model was proposed by Karaboga in [2]. The ABC model was suited for numeric optimization, beehive metaphor was utilized more for web search related tasks.

In our previous work, an approach was proposed in [14], utilizing beehive metaphor as a basis for web search engine. Results have been retrieved by intelligent agents cooperating as a bee swarm, communicating and propagating the best sources by dancing. Further development of this idea was pursued in [13] by elaborating on the web crawler part of the proposed search engine. In a follow up research, further possibilities of BM deployment for function optimization and hierarchical optimization are presented in [11].

3 Keyword Extraction Algorithm

When performing a keyword extraction on a number of news articles, our main goal is to obtain information about current events and stories, that span multiple articles and evolve over time. To achieve this, we take steps to extract only the keywords that are relevant to a set of articles, by utilizing honey bee inspired agents, that travel from article to article and look for keywords that suit more than one article at a time. The analysis of news articles is performed by a set of web crawling agents called *bees*. Every bee remembers a single article and a single keyword, that may change over time. Bees act independently and there is no central mechanism of coordination. Bees may however communicate individually by dancing.

The algorithm starts by initialization phase, depicted in Fig. 1. Each bee is assigned a random article from a pre-selected set of starting article and selects a candidate keyword from this article. We are currently selecting keyword candidates from nouns in articles titles. This approach is based on [18], which uses named entities as terms for clustering. In addition to named entities, we are also extracting other nouns, as they can also be significantly related to a news story. It is important to note, that although we extract the starting keyword candidates only from the titles of articles, this serves simply as a starting point and does not limit the set of possible keywords of an article only to its title. In following steps, more candidate keywords may be introduced by bees for evaluation, as described later.

```
Require: Non-empty set of $Bees$.
1: for all $n$ of $Bees$ do
2:     $a(n) \leftarrow$ random article
3:     $key(n) \leftarrow$ random keyword from $Article(n)$
4:     $status(n) \leftarrow$ "foraging"
5: end for
```

Fig. 1. Initialization of bees

After initialization, the main phase commences, during which the keywords are being extracted and their weights determined. In accordance with BM model, bees switch between three states during the main phase:

1. foraging
2. observing
3. dancing

3.1 Foraging

Foraging is the default state that bees assume after initialization. The algorithm of foraging for a single bee is depicted in Fig. 2. When entering foraging phase a bee always has an article a_c and a keyword candidate key selected and is trying to find another article, relevant to current keyword candidate. Bee starts foraging by selecting a new article a_n and evaluating relevance of key, separately for a_c and a_n. Thus $r_c = relevance(key, a_c)$ and $r_n = relevance(key, a_n)$.

The relevance of a given keyword to a given article may be established in two ways. If key has already been identified as a possible keyword for the article a_c, weight of key is used as a relevance. If key is new to the article, the relevance is based on the content of a_c. The relevance based on content is determined by calculating a score for key according to its occurrence in a_c. The article is divided into blocks of fixed length and number n_c of blocks, that contain key is counted. This number represents the base score of key, which is further increased, if key occurs in the title or first paragraph of a_c. Relevance $r_c = relevanc(key, a_c)$ based on the content of an article is then calculated as $r_c = n_c/N_c$, where N_c

```
Require: current article $a_c$, keyword $key$
 1: new article $a_n \leftarrow selectNewArticle()$
 2: $r_c \leftarrow relevance(key, a_c)$
 3: $r_n \leftarrow relevance(key, a_n)$
 4: $p = min(r_c, r_n)$
 5: Decide whether to leave using probability $1 - p$
 6: if decided to leave then
 7:    $status \leftarrow$ "observing"
 8:    Leave the article and start observing
 9: else
10:    Decide whether to dance using probability $p$
11:    if decided to dance then
12:       $newConnection(a_c, a_n)$
13:       $connectionKeyword \leftarrow key$
14:       $connectionStrength \leftarrow p$
15:       $updateKeywords(a_c, key)$
16:       $updateKeywords(a_n, key)$
17:       $status \leftarrow$ "dancing"
18:       Start dancing for the current article and keyword
19:    else
20:       Continue foraging
21:    end if
22: end if
```

Fig. 2. Foraging phase of a single bee

is maximum possible score for a_c. The obtained value serves only as a estimate of the relevance and is used to establish probabilities of bees leaving the article and dancing for it.

Alternative relevance functions may be used as well, however it is important, that the returned values use the full interval of $[0, 1]$. The reason for this being, that the returned value further serves to determine the probability $p = min(r_c, r_n)$, that the bee will keep on foraging from the same article. Minimum of both relevance values is used, because our goal is to extract story relevant keywords and as such, every keyword is relevant only up to its ability to establish a connection between multiple articles. In other words, keywords relevant only to a single page are not interesting for us. After completing the keyword evaluation, a bee makes a decision, whether to abandon the article, probability of it being $1 - p$. If it decides to leave, both current keyword and article are abandoned and the state of bee changes from foraging to observing. If the decision of the bee is not to abandon the source, it makes further decision, whether to propagate the article-keyword pair by dancing, probability of this being also p. If the bee decides to dance, a connection is established between both articles, *key* serving as an attribute of this connection and p serving as a weight of *key*. Value of p serves as a measure of confidence, that a connection based on *key* exists between two articles. The weight of connection between two articles thus depends on the relevance of *key* to both concerned articles. If relevance of *key* to at least one of both articles is low, the weight of mutual connection between respective articles will be low as well. Also the weight of a connection is not final and it may change over time, if relevance of *key* to one of concerned articles is reviewed in following iterations of the algorithm.

After establishing a connection between two articles, individual sets of keywords for both respective articles are also updated, adding *key* with weight of $max(r_c, r_n \times d)$ to r_c and $max(r_c \times d, r_n)$ to r_n, effectively "trading" the keyword between both articles. Constant d is a decay factor, that reduces the weight of a keyword for an article in case, it has been adopted from an another article. In our experiments we have used values of $d \in [0.6, 0.9]$. The described mechanism of connecting articles via common keywords is illustrated by Fig. 3. The adopted keyword is equivalent to keywords of an article selected from its own title and may also be retrieved by other foraging bees. Note, that even if adopted from an another article, a certain degree of relevance is guaranteed, otherwise the p probability would be zero and a bee would abandon the article immediately, never proceeding to dance for it. If a bee decides not to dance for the article-keyword pair, the bee proceeds to forage from the article further, by selecting a new article a_n.

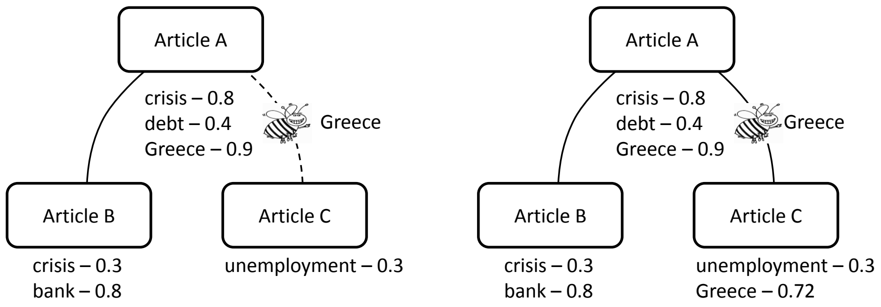


Fig. 3. A bee has chosen keyword "*Greece*" from Article A and aims to explore possible connection with Article C (left). After having chosen to dance, bee establishes connection between Article A and Article C and transfers keyword, its weight reduced by decay factor of 0.8 (right)

3.2 Observing

If a bee decides to abandon its article during the foraging phase, an observing phase commences. The algorithm of observation phase is depicted in Fig. 4. Observing lasts for a set number of iterations, during which a bee can decide to adopt a keyword-article pair from the ones being currently propagated by dancing bees. The probability p_a of adopting a keyword-article pair is proportional to the number of bees currently dancing, $p_a = dancingBees/allBees$.

```
Require: time to leave $ttl$
 1: if $ttl = 0$ then
 2:     $a \leftarrow$ random article
 3:     $key \leftarrow$ random keyword from $a$
 4:     $status \leftarrow$ “foraging”
 5:     End observing and start foraging from new source
 6: else
 7:     $p_a = dancingBees/allBees$
 8:     Decide whether to choose a new article using $p_a$
 9:     if decided to choose new article then
10:         bee to follow $followBee \leftarrow$ random dancing bee
11:         $article \leftarrow$ article of $followBee$
12:         $key \leftarrow$ keyword of $followBee$
13:         $status \leftarrow$ “foraging”
14:         Start foraging the new article
15:     else
16:         $ttl \leftarrow ttl - 1$
17:     end if
18: end if
```

Fig. 4. Observing phase of a single bee

When a bee b_c decides to choose a new article from the ones being currently propagated, a random dancing bee b_d is selected. Subsequently b_c adopts the article and keyword propagated by b_d and proceeds to start foraging from its newly chosen article. Otherwise the bee does nothing and proceeds to next iteration. If a bee fails to choose a new article in a set amount of iterations, it returns back to foraging, choosing a random article and a keyword from it in the same way as during the initialization phase.

3.3 Dancing

The notion of dancing bees is taken from real honey bees, that use the famous waggle-dance to relay the position of a food source to other bees. With our approach, the source of nutrition is represented by a keyword-article pair. If a bee finds a suitable keyword-article pair, that is potentially related to a news story, it may decide to propagate the pair and invite other bees to explore the possible relationships with other articles.

A bee decides to propagate its keyword-article pair during the foraging phase. In contrast to the strictly set duration of observation phase, the duration of dancing phase is variable. The length of dance l is proportional to the probability p of dancing (see Fig 2), $l = \lceil p \times DT \rceil$, where DT is a parameter of BM model that represents the maximum possible length of a dance, measured in iterations [14]. After the dance ends, the bee returns to foraging, keeping the same article and keywords, that it propagated.

The number of bees dancing at a given time is influenced by the overall relevance of keywords currently examined by foraging bees. If the overall relevance of keywords is relatively high, the articles will generally be propagated more. Thus the chance of an observing bee to pick a new article from the ones being propagated will be higher, as the chance of choosing a propagated article is directly proportional to the number of dancing bees. If, on the other hand, the relevance of currently found keywords is low, the number of dancing bees will be low as well and observing bees will more often tend to forage from random articles and try different keywords, increasing the chance of finding new relevant keywords.

4 Keyword Extraction

To evaluate the proposed approach, we have conducted an experiment by deploying a system implementing our approach on the Web and daily tracking the changes in retrieved keywords. This experiment was performed on the Reuters web-site [1] over a period of 9 days from August 4th to 12th, 2012. Each day an independent keyword extraction was performed, running for 100 iterations with parameters of *BIOR = 0*, *BISB = 100*, *MDT = 4* and *OT = 2*. The choice of 100 bees and 100 iterations affects only the scope of search. Parameter of MDT affects the rate of dancing and OT influences the rate of scouting for new articles. Values of MDT and OT have been set empirically, based on previous unpublished work, as they provide balance between dancing and scouting. For further description of model parameters please refer to [14]. During the extraction a portion of Reuters web site was indexed, focusing on the most recent articles. We have extracted 298 unique keywords, 99 of which being proper nouns, accounting for 33.22%. For every day i we have noted the number n_i^k of articles for which the keyword k was recognised, which we consider a measure of popularity of keyword k. The overall most popular keywords are presented in Table 1, along with $n^k = \sum_i n_i^k$, the sum of recognised articles over all days for a given keyword and a percentage of $N = \sum_k n^k$, total sum of all articles for all identified keywords.

Table 1. Overall most popular keywords

keyword	n^k	$n^k/N[\%]$	*keyword*	n^k	$n^k/N[\%]$
Syria	189	7.32	court	54	2.09
Euro	79	3.06	Samsung	50	1.94
Apple	74	2.86	ECB	49	1.90
Egypt	83	3.21	attack	47	1.82
Afghan	61	2.36	trial	38	1.47
shooting	64	2.48	murder	33	1.28
China	63	2.44	Libor	29	1.12
Colorado	55	2.13	Aleppo	26	1.01

[1] `http://www.reuters.com/`

We can see that although proper noun keywords represent only 33.22% of all the identified keywords, the proportion of proper nouns among the most popular keywords is considerably higher, with 8 of 10 top keywords being proper nouns. This corresponds to our intuitive assumption that proper nouns have a higher potential of representing a news story than common nouns.

The popularity development for top eight keywords over the duration of experiment is depicted in Fig. 5. We can observe keywords relevant to long term stories, such as "*Syria*", "*Euro*" and "*Apple*", along with keywords "*Colorado*" and "*shooting*", representing events that have been recent in time of the experiment, namely shootings in Colorado and Wisconsin Sikh temple. Although Colorado shooting received broader media coverage just prior to our experiment, the events in Wisconsin reopened the talks about the Colorado shooting and thus "*Colorado*" was also identified as a currently interesting keyword. The keywords of "*Sikh*" and "*temple*" have also been identified, however they have ranked lower, as the number of articles covering these events had been considerably smaller.

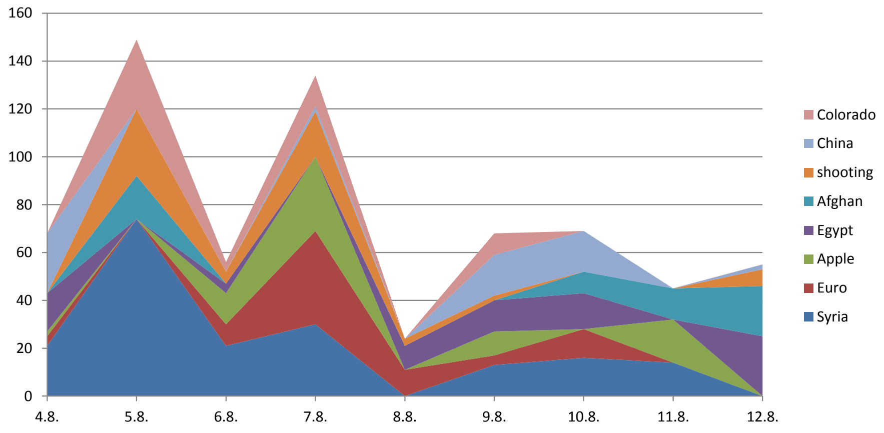


Fig. 5. Development of keyword popularity over time, measured in number of articles relevant to a keyword. Note the popularity increase of keywords "*shooting*" after Sikh temple shootings from August 5th and "*Apple*" after developments in US trial of Apple-Samsung from August 6th and 11th.

After further scrutiny of indexed articles, we have identified additional events connected to the keywords. "*Afghan*" was mentioned in August 10th after attack of Afghan policeman on U.S. soldiers, "*China*" witnessed a major trial versus Gu Kailai, "*Egypt*" experienced an attack on border guard and "*Apple*" gained attention in connection to Apple-Samsung trial. All of these news stories receiving significant coverage on Reuters web site.

4.1 Normalization of Results

Although the number of iterations performed every day was firmly set, the rate of daily keyword extraction differed significantly, ranging from 124 keyword-article pairs on 11th to 440 on 7th, with daily mean of 286.89 keyword-article pairs. This is mostly attributed to the fact, that when presented by a smaller variety of topics with highly relevant keywords, the rate of propagation will be higher and observing bees will be more likely to select a new article from the propagated set, decreasing their chance to explore and look for new articles. As a result, various numbers of keyword-article pairs may be extracted every day, depending on the article structure. Therefore the keywords extracted in days, when less keyword-article pairs have been recognised, will seem less popular, when compared to keywords extracted in other days.

To compensate for this, we have normalized the data by assigning every day i a factor $f_i = N_i/\bar{N}$, N_i being the total number of keyword-article pairs obtained on the day i and $\bar{N}$ being the mean value. Subsequently we have used f_i as a weight to adjust the numbers of identified articles. The Table 2 summarizes the adjusted overall numbers of articles for the most popular keywords, preserving the order of keywords from Table 1 for comparison. The development of adjusted keyword popularity is presented in Fig. 6. From the presented data we can see, that the presented approach is able to identify popular keywords related to news stories in a set of articles. Most of the keywords are related to long term or general stories, but keywords related to individual recent events are also present, due to intensive coverage of the concerned events. Changes over time are also captured and may be mapped to real events.

Table 2. Most popular keywords after normalization

keyword	n^k	$n^k/N[\%]$	*keyword*	n^k	$n^k/N[\%]$
Syria	177.30	6.87	court	49.90	1.93
Euro	75.50	2.92	Samsung	55.71	2.16
Apple	92.65	3.59	ECB	49.85	1.93
Egypt	98.10	3.80	attack	49.41	1.91
Afghan	78.23	3.03	trial	28.90	1.12
shooting	56.32	2.18	murder	26.38	1.02
China	55.30	2.14	Libor	27.75	1.07
Colorado	41.79	1.62	Aleppo	25.31	0.98

4.2 Grouping of Keywords According to News Stories

In previous experiment we have outlined the results of our keyword extraction mechanism by measuring the popularity of extracted keywords. The popularity of a given keyword k was measured for each day i as a number of articles n_k^i for which k was evaluated as relevant. Overall popularity of k is then calculated as $n^k = \sum_i n_i^k$. Although this gives us some insight into what story related

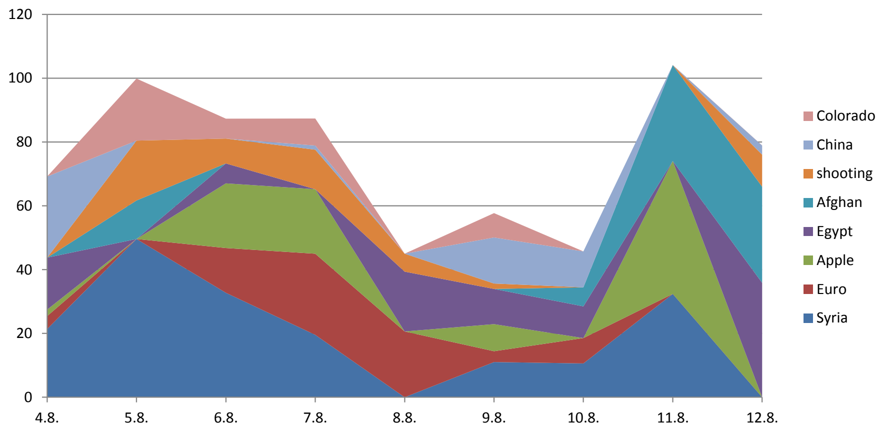


Fig. 6. Development of keyword popularity over time, adjusted for different daily rates of keyword extraction. The significance of keywords from August 11th has increased notably, as the least article-keyword pairs have been identified on this day.

keywords were popular during the testing period, the most popular keywords listed in Table 2 only account for 38.27% of all identified article–keyword pairs. Furthermore, the group of top 8 keywords tracked in Fig. 6 represents only 26.15% of all article–keyword pairs.

In order to assess the ability of our approach to track news stories, instead of single most popular keywords, we increase the scope of the analysis to include a broader range of less popular keywords, by generalizing the notion of single keyword popularity to *story popularity.* Therefore instead of tracking popularity n^k of every keyword individually, we group the keywords according to their relevance to news stories, that they represent. The grouping in outlined experiment was performed manually, after thorough scrutiny of retrieved articles. We have omitted ambiguous keywords with no clear reference to a particular news story, such as "*parliament*" or "*banks*". The following stories have been identified:

1. Syria uprising
2. China Bo Xilai trial
3. Egypt police clashes
4. Colorado shooting
5. Apple vs Samsung trial
6. Euro debt crisis
7. Libor manipulation scandal
8. Afghan conflict

Popularity m of an individual story s is evaluated as $m^s = \sum n^k$ for all n^k related to s. No keyword is related to more than one story. Popularity of resulting news stories is listed in Table 3. First column of the table shows the popularity m^s of story s and second column expresses the popularity of the story as percentage of all retrieved article–keyword pairs. The total number of article–keyword pairs for all keywords included in all stories reached 1357, that is 52.55% of all retrieved article–keyword pairs, compared to 994 article–keyword pairs related to top 16 keywords listed in Table 1, which represents 38.50% of all article–keyword pairs.

Table 3. Overall story popularity

story	m^s	$m^s/N[\%]$
Syria uprising	303	11.74
China trial	197	7.63
Egypt clashes	188	7.28
Colorado shooting	177	6.86
Apple vs Samsung	163	6.31
Euro debt crisis	147	5.69
Libor scandal	99	3.83
Afghan conflict	83	3.21
Total	1357	52.55

A more detailed look at dynamics of story popularity is given in Fig. 7. The total number of article–keyword pairs relevant to stories is higher than the number of article–keyword pairs relevant to single keywords, as depicted in Fig. 5. However the number of article–keyword pairs still differs significantly from day to day. Therefore same normalization as with previous experiment is necessary, in order to equalize for different daily article retrieval rates.

Normalized values of story popularity are presented in Table 4. The order of stories is slightly changed, with respect to Table 3. Story of "*Syria uprising*" retains first place, but second place is occupied by "*Egypt clashes*" and "*Apple vs Samsung*" moves up from fifth place to third. Total number of article-keyword pairs after normalization reached 1399.57, which is 54.21% of all retrieved article-keyword pairs. This is attributed to the fact, that not all identified keywords can be confidently related to a specific story.

Development of story popularity after normalization is given in Fig. 8. The shape is similar to the one given in Fig. 6, as 7 out of 8 stories are strongly represented by a single keyword from the set depicted in Fig. 6. However, multiple other keywords have been included, along with the addition of a story labelled "*Libor scandal*" that is not represented by any keyword from the group of top 8 keywords. The number of article-keyword pairs represented by tracked stories has thus increases considerably, from 26.15% to 54.21%.

4.3 Agent Distribution

During the course of article retrieval, employed agents may take up three different tasks of *foraging*, *observing* and *dancing*, which serves as a propagation

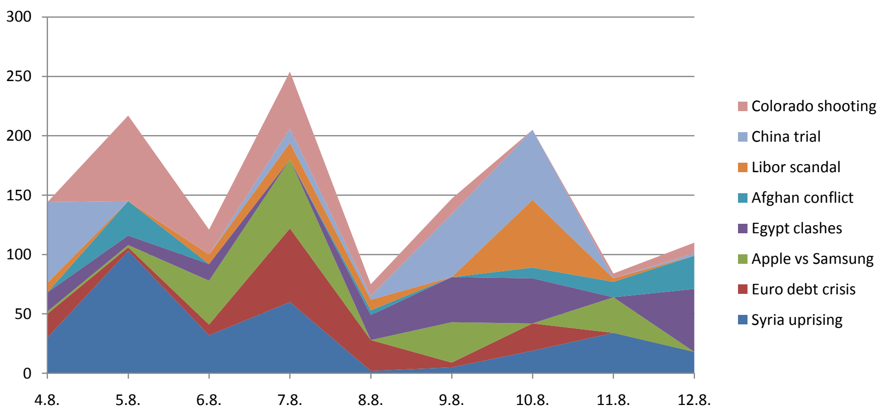


Fig. 7. Popularity of stories over time, measured as a number of articles relevant to keywords, summed over each keyword related to a given story. Most of top keywords from Table 1 represent a distinct story, except "*shooting*", which is grouped with "*Colorado*" in "*Colorado shooting*" story. Furthermore an additional story of "*Libor scandal*" is included, which is not represented by any keyword from the group of most popular keywords tracked in Fig. 5.

Table 4. Popularity of stories after normalization

story	m^s	$m^s/N[\%]$
Syria uprising	313.64	12.15
China trial	169.57	6.57
Egypt clashes	216.30	8.38
Colorado shooting	164.25	6.36
Apple vs Samsung	197.15	7.64
Euro debt crisis	144.20	5.58
Libor scandal	91.32	3.54
Afghan conflict	103.14	3.99
Total	1399.57	54.21

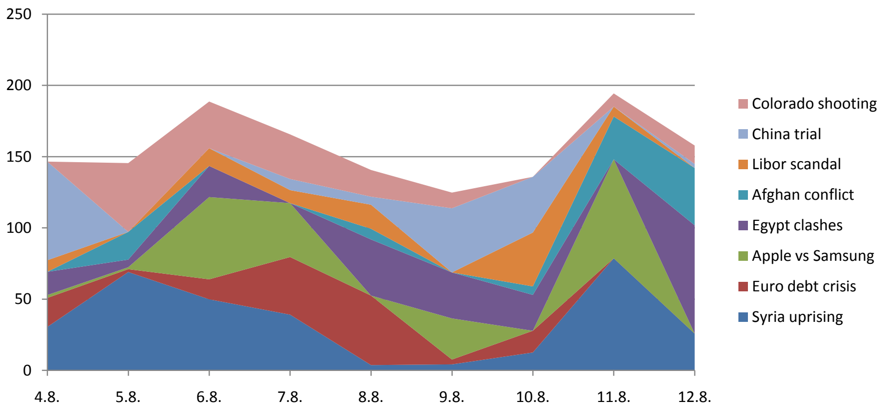


Fig. 8. Development of story popularity over time, adjusted for different daily rates of keyword extraction. Although the number of articles has been normalized to account for different daily rates of keyword extraction, the relative popularity of story related keywords still fluctuates from day to day, when compared to total popularity of all extracted keywords extracted on a given day.

mechanism for suitable article–keyword pairs. The distribution of agents among these tasks determines the behaviour of the system as a whole.

The number of foraging agents determines the rate of keyword extraction. Foraging agents carry a single keyword candidate and evaluate visited articles in order to identify suitable article–keyword pairings. The number of dancing agents influences the exploration rate of new, previously unexplored article–keyword pairs, as the proportion of dancing agents in population determines the chance of a propagated pair to be selected by an observing agent. A high number of dancing agents therefore inhibits the exploration, as observing agents will tend to select their new article–keyword pairs from the propagated pool, thus rejecting the possibility of adopting a new, previously untested keyword. On the other hand, high number of dancing agents reinforces the tendency of agents to focus on exploration of article–keyword pairs with already popular keywords. Therefore a suitable agent distribution is crucial in order to maintain efficient exploration of article–keyword space.

Fig. 9 depicts the distribution of agents attained during the experimental run outlined in this section. The initial distribution of agents is arbitrary and subject to setting, however after a certain number of iterations, the distribution of agents reaches an equilibrium. The precise equilibrium point is determined by the parameters of the run. Generally a longer OT (*observation time*) shifts the equilibrium towards higher number of observing agents, while longer MDT (*mean dance time*) increases the number of agents dancing.

Fig. 10 documents a similar scenario as with Fig. 9, however the equilibrium is reached earlier. This is standard behaviour and it is strongly dependent on the

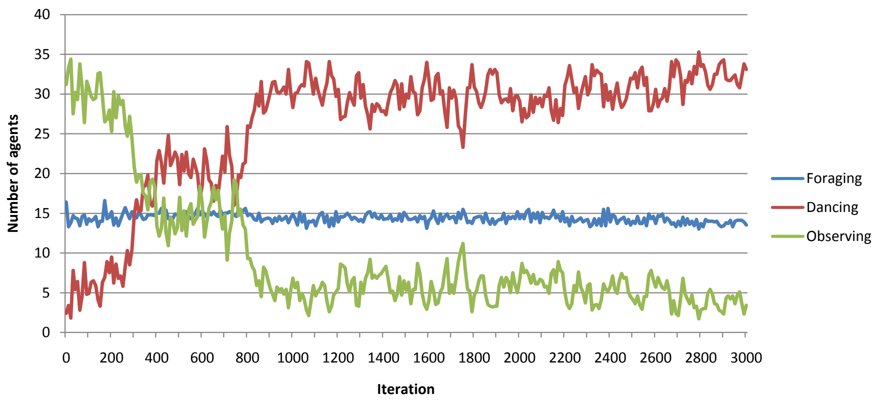


Fig. 9. Distribution of 50 agents among individual tasks over time. Only first 3000 iterations from a 10 000 iteration run are shown for the sake of clarity. Initially the portion of observing agents is high and drops gradually, as suitable keywords are established. After first thousand iterations, the distribution among tasks reaches equilibrium and from this point only minor fluctuations occur until the end of the run.

initial landscape of the articles. A faster reaching of equilibrium suggests that a smaller number of articles with specific keywords have been retrieved in the early stages of the run.

4.4 Discussion

The aim of the performed experiment was twofold: to demonstrate the ability of the proposed approach to extract meaningful keywords representing news stories and to determine, whether the system is able to react to dynamic environment and pick up on new keywords, as news stories develop.

After empirical analysis of the results, we generally consider the retrieved keywords to be relevant to news stories. The most popular keywords tend to be more general and encompass broader range of events, such as "*Syria*", or "*Euro*". However, less general keywords relevant to specific events are also present, such as "*Colorado*". Keywords not relevant to stories have been also recognised, due to the fact, that some keywords generally appear in multiple related articles, without pointing to a specific event, or story. An example of such keywords would be "*oil*", or "*economy*". Fortunately, although these keyword also appear frequently in news coverage and have been also extracted by our system, they rank considerably lower than major story related keywords.

When comparing results obtained from different days, we observe, that the system indeed reacts to different unwinding stories. The main problem is, that while identified keywords are relevant to new stories, the most popular keywords are too general and although they hint that new events took place in a certain general storyline, we are currently unable to pinpoint these new events simply by looking at retrieved keywords. To do so, we had needed to perform manual

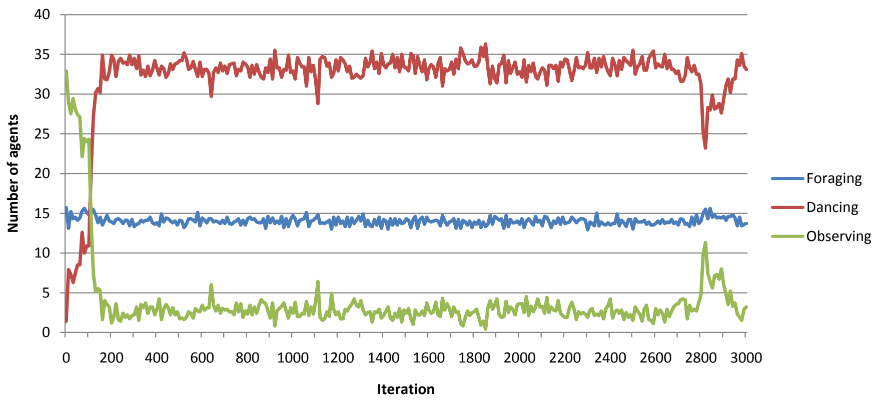


Fig. 10. Alternative run with the same settings, as in Fig. 9. The equilibrium is reached considerably earlier and maintained afterwards. The sharp increase in the number of observing agents towards the end is attributed to the retrieval of a large set of fresh articles, which is subsequently gradually processed.

inspection of retrieved data. However, it is important to add, that the specific keywords that narrow down the storyline, or pinpoint a single event, are also extracted, although they rank lower than the best general keywords. An example of such more specific keywords would be "*Aleppo*", related to "*Syria*", or "*Gu*" and "*Xilai*" related to "*China*".

One possible way of incorporating the less popular keywords into the overall result is by generalizing the idea of keyword popularity through addition of multiple keywords, related to a single news story, resulting in an overall measure of story popularity. This way, we have been able to achieve a more complex look at the current news story landscape, increasing the fraction of used article–keyword pairs from 26.15% to 54.21%. Beyond that, the extracted keywords have been too ambiguous, or general to confidently assign them to a specific news story.

At this time, we are unable to precisely evaluate the reaction time of the approach to any given event at a daily granularity of snapshots, based on the data presented in Fig. 5 and Fig. 6. The snapshots have been taken every day at 22:00 GMT, by which time the most of the significant events have been already broadly covered in news. Finer granularity of snapshots will have to be used in order to achieve more significant results regarding the dynamic reaction of the system to new events.

Due to the online nature of the proposed algorithm, it is difficult to compare it directly with common keyword extracting approaches that use *naive Bayes* or *Support Vector Machine* classifiers. The main difference lies in the fact, that our approach processes news articles "on the fly" as they are discovered, without any pre-processing steps or supervised learning. The algorithm can be deployed once to rapidly analyse a number of articles, or continuously to retrieve keywords over a period of time, reacting to changes and new articles dynamically. This is made possible by light-weight relevance function that evaluates only one keyword and one article at a time. The fact that no single bee carries accurate information about article keywords is countered by the multiplicity of bees. The best keywords are allowed to gradually surface through propagation by the swarm, a mechanism similar to the dancing observed with natural honey bees.

5 Conclusion

The approach presented in this paper aims to provide insight into currently ongoing news stories, by extracting keywords relevant to news stories from articles on the Web. The approach is inspired by behaviour of honey bees. A set of agents is used to determine the relevance of selected candidate keywords by comparing them to the articles. The best keywords are propagated through the relevant articles, similarly to pollen being carried by honey bees from one flower to another. This way a keyword has a chance to be identified as relevant to an article, even if it is not within the set of keyword candidates for the given article. Furthermore, the most promising keywords may be explored in this way, by being compared to multiple articles, while irrelevant keywords will be discarded and not explored further.

The performed experiments have shown that the proposed approach is able to retrieve keywords relevant to stories, without previous training, also being able to react to current events. When measuring the popularity of keywords in number of relevant articles, we observe the prominence of more general keywords, that are related to long term general stories. The more specific keywords are also retrieved, they are however ranked lower, as they are not mentioned in many articles. One of the possible approaches to addressing this issue is the grouping of articles according to their relevance to general news stories. This way the popularity of a certain news story reflects both popularity of major, often more general keywords and less popular, but more specific keywords as well. The focus of our current work is aimed at providing a more structured representation of a news story, that would allow us to retain relationships between individual keywords and relevant articles within a news story and also to track the development of a news story in time.

Acknowledgment. This work was partially supported by the grants VG1/0671/11, APVV 0208-10, and it is the partial result of the Research & Development Operational Programme for the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF.

References

1. Bohne, T., Rönnau, S., Borghoff, U.M.: Efficient keyword extraction for meaningful document perception. In: Proceedings of the 11th ACM Symposium on Document Engineering, DocEng 2011, pp. 185–194. ACM, New York (2011)

2. Karaboga, D.: An idea based on Honey Bee Swarm for Numerical Optimization. Tech. Rep. TR06, Erciyes University (October 2005)
3. Karaboga, D., Akay, B.: A survey: algorithms simulating bee swarm intelligence. Artif. Intell. Rev. 31, 61–85 (2009)
4. Kriz, J.: Keyword extraction based on implicit feedback. Information Sciences and Technologies Bulletin of the ACM Slovakia 4, 43–46 (2012)
5. Kuzar, T.: Clustering on social web. Information Sciences and Technologies Bulletin of the ACM Slovakia 5, 34–42 (2013)
6. Lee, S., Kim, H.J.: News keyword extraction for topic tracking. In: Fourth International Conference on Networked Computing and Advanced Information Management, NCM 2008, vol. 2, pp. 554–559 (September 2008)
7. Lemmens, N., et al.: Bee Behaviour in Multi-agent Systems (A Bee Foraging Algorithm). In: Adaptive Agents and MAS III, Berlin, pp. 145–156 (2008)
8. Leuski, A.: Evaluating document clustering for interactive information retrieval. In: Proceedings of the Tenth International Conference on Information and Knowledge Management, pp. 33–40. ACM, New York (2001)
9. Lorenzi, F., Santos, D.S., Bazzan, A.L.C.: Negotiation for task allocation among agents in case-base recommender systems: a swarm-intelligence approach. In: Proceedings of the Workshop Multi-Agent Information Retrieval and Recommender Systems - Nineteenth International Conference on Artificial Intelligence (IJCAI 2005), pp. 23–27 (2005)
10. Memari, A., Amer, M., Marx Gmez, J.: A beehive-like multi-agent solution to enhance findability of semantic web services and facilitate personalization within a p2p network. In: ICT Innovations, pp. 227–236. Springer (2010)
11. Navrat, P., Jelinek, T., Jastrzembska, L.: Bee hive at work: A problem solving, optimizing mechanism. In: World Congress on Nature Biologically Inspired Computing, NaBIC 2009, pp. 122–127 (December 2009)
12. Navrat, P.: Bee hive metaphor for web search. Communication and Cognition-Artificial Intelligence 23(1-4), 15–20 (2006)
13. Navrat, P., Jastrzembska, L., Jelinek, T., Ezzeddine, A.B., Rozinajova, V.: Exploring social behaviour of honey bees searching on the web. In: Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, pp. 21–25. IEEE Computer Society, Washington, DC (2007)
14. Navrat, P., Kovacik, M.: Web search engine as a bee hive. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2006, pp. 694–701. IEEE Computer Society, Washington, DC (2006)
15. Rambharose, T., Nikov, A.: Computational intelligence-based personalization of interactive web systems. WSEAS Trans. Info. Sci. and App. 7, 484–497 (2010)
16. Schultze, S.J.: A collaborative foraging approach to web browsing enrichment. In: CHI 2002 Extended Abstracts on Human Factors in Computing Systems, CHI 2002, pp. 860–861. ACM, New York (2002)
17. Teodorović, D.: Bee colony optimisation (bco). In: Lim, C.P., Jain, L.C., Dehuri, S. (eds.) Innovations in Swarm Intelligence. SCI, vol. 248, pp. 39–60. Springer, Heidelberg (2009)
18. Toda, H., Kataoka, R.: A clustering method for news articles retrieval system. In: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, pp. 988–989. ACM, New York (2005)

19. Toda, H., Kitagawa, H., Fujimura, K., Kataoka, R.: Topic structure mining using temporal co-occurrence. In: Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, New York, NY, USA, pp. 236–241 (2008)
20. Turney, P.D.: Learning algorithms for keyphrase extraction. Information Retrieval 2, 303–336 (2000)
21. Vadrevu, S., Teo, C.H., Rajan, S., Punera, K., Dom, B., Smola, A.J., Chang, Y., Zheng, Z.: Scalable clustering of news search results. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 675–684. ACM, New York (2011)
22. Wang, Y., Kitsuregawa, M.: Link based clustering of web search results. In: Wang, X.S., Yu, G., Lu, H. (eds.) WAIM 2001. LNCS, vol. 2118, pp. 225–236. Springer, Heidelberg (2001)
23. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: Kea: practical automatic keyphrase extraction. In: Proceedings of the Fourth ACM Conference on Digital Libraries, pp. 254–255. ACM, New York (1999)

Incorporating Highly Explorative Methods to Improve the Performance of Variable Neighborhood Search

Mohammad R. Raeesi N. and Ziad Kobti

School of Computer Science
University of Windsor
Windsor, ON, Canada
{raeesim,kobti}@uwindsor.ca

Abstract. Variable Neighborhood Search (VNS) is one of the most recently introduced metaheuristics. Although VNS is successfully applied on various problem domains, there is still some room for it to get improved. While VNS has an efficient exploitation strategy, it suffers from its inefficient solution space exploration. To overcome this limitation, VNS can be joined with explorative methods such as Evolutionary Algorithms (EAs) which are global population-based search methods. Due to its effective search space exploration, Differential Evolution (DE) is a popular EA which is a great candidate to be joined with VNS. In this article, two different DEs are proposed to be combined with VNS. The first DE uses explorative evolutionary operators and the second one is a Multi-Population Differential Evolution (MP-DE). Incorporating a number of sub-populations improves the population diversity and increases the chance of reaching to unexplored regions. Both proposed hybrid methods are evaluated on the classical Job Shop Scheduling Problems. The experimental results reveal that the combination of VNS with more explorative method is more reliable to find acceptable solutions. Furthermore, the proposed methods offer competitive solutions compared to the state-of-the-art hybrid EAs proposed to solve JSSPs.

Keywords: Differential Evolution, Multiple Population, Variable Neighborhood Search.

1 Introduction

Optimization is an area of research where the goal is to optimize a system based on its input parameters. In general, an optimization problem is defined as finding the best set of input parameters to make a system as effective as possible. Optimization problems are categorized into two classes based on the type of their input parameters. The problems with continuous parameters are classified as global optimization problems, while the ones with discrete parameters are considered as combinatorial optimization problems. The focus of this paper is on a permutation problem which is a combinatorial optimization problem.

Various types of algorithms are proposed to solve optimization problems. A number of heuristics are presented to deal with very specific optimization problems. Although they perform well in their applicable domain, they cannot offer the same performance for related problem domains. Local search, for instance, is a heuristic which starts at an initial random solution, looks for a better solution in the neighborhood of its current solution, and returns the best found solution when it converges.

Due to the fact that heuristics are not successful in general, metaheuristics are presented which are general procedures to design heuristics for optimization problems. Metaheuristics are so-called complex heuristics. For example, there are a number of extended versions of simple local search such as Repeated Local Search (RLS) and Iterated Local Search (ILS). Since a local search might trap into local optimal regions, its execution for a number of times increases the chance of finding an acceptable solution. This strategy is known as RLS. ILS is an extended version such that it incorporates the best solutions found in previous executions. In fact, instead of starting from another random solution, ILS perturbs the recently found best solution and continues.

Variable Neighborhood Search (VNS) is one of the most recently introduced metaheuristics proposed by Mladenovic and Hansen [1]. VNS can also be considered as an extended version of ILS. The idea of VNS is to incorporate a number of neighborhood structures and switch among them at the time of local search execution. VNS starts with searching locally with respect to one neighborhood structure and it switches to another one as soon as it converges. This strategy decreases the chance of immature convergence dramatically.

Due to its generality, VNS is successfully applied in various areas such as the Traveling Salesman Problem [2], the Open Vehicle Routing problem [3], the p-Median problem [4], and the Graph problem [5].

The performance of VNS is highly dependent to the definition of its neighborhood structures. Complement neighborhood structures provide VNS a powerful exploitation strategy. However, since VNS is a local search approach, there is still some chance of trapping into a local optimum by spinning in some previously investigated regions. Although a number of different strategies such as Parallel VNS [6], Multi-Start VNS [7], and Population-based VNS [8] are proposed to overcome this limitation, VNS cannot perform well as a global approach to deal with complex optimization problems; it suffers from its inefficient solution space exploration.

A more effective way to enhance VNS is to join it with a highly explorative method. In fact, the solution space exploration should be conducted by the joint method to find promising regions and VNS should be responsible for exploiting the promising regions. It is expected for this combination to have a great performance. In our recently published article [9], we incorporated a Genetic Algorithm (GA) [10] as the joint method. This combination which is a Memetic Algorithm (MA) [11] benefits a global population-based approach for its exploration and a powerful local search method as its exploitation strategy. The results of this combination show that the GA helps VNS to offer a better performance compared to the single VNS as a global method.

In this article, two more approaches are presented to improve the performance of a VNS. The first approach is to combine VNS with a Differential Evolution (DE) [12]. DE is a popular evolutionary method due to its powerful solution space exploration, and therefore it could be the best candidate to be combined with VNS. Although this combination is one of the bests in terms of incorporating both powerful exploration and exploitation strategies, there is still some chance of immature convergence. An approach to decrease this chance is to incorporate multiple populations for DE. Dividing the whole population into a number of sub-populations decreases the chance of trapping into local optimal regions. When a sub-population converges to a local optimum, it can recover itself by incorporating the knowledge migrated from other sub-populations. In other words, in this article two methods are proposed including the joint of DE and VNS and combination of Multi-Population Differential Evolution (MP-DE) and VNS.

The main contribution of this article is to represent the impact of different population-based searches with different levels of exploration on the performance of VNS. Since VNS is a powerful local search with a great exploitation strategy, it is expected that its combination with more explorative methods offer better performance. In order to evaluate the proposed methods, Job Shop Scheduling Problem (JSSP) is considered as our test bed. JSSP is a well-known combinatorial optimization problem which is considered as a complex problem based on the dependency of its input parameters to each other.

The remainder of this article is organized as follows. Section 2 briefly introduces VNS, followed by an introduction of DE and MP-DE in Sections 3 and 4, respectively. The problem domain of JSSP is concisely defined in Section 5. The proposed methods are described in details in Section 6, followed by representing the experimental results in Section 7. Finally the last section illustrates the conclusion remarks.

2 Variable Neighborhood Search

A simple local search method determines a neighborhood area for each solution which is defined based on a neighborhood structure. It starts with a feasible solution which is usually selected randomly from the solution space. It then searches for a better solution by applying some moves with respect to the neighborhood structure. The main drawback of this simple structure is its immature convergence. If a local search is trapped in a local optimum, it spins around the local optimal region and never escape from that region. Therefore, instead of converging to the global optimum, it converges to a local one. This issue is more crucial in complex optimization problems, specially large ones, where the chance of trapping into local optima is higher.

To defeat this issue, VNS is introduced which incorporates a number of neighborhood structures. VNS starts searching with respect to the first neighborhood structure and as soon as it finds a local optimum it switches to the next neighborhood structure. This routine continues until an optimum with respect to all neighborhood structures is reached. Therefore, complement neighborhood structures provides a more powerful VNS.

It can be concluded that the idea of VNS comes from the following facts:

1. A local optimal solution in one neighborhood area is not necessarily a local optimum in another neighborhood area.
2. A global optimal solution is a local optimal solution for all possible neighborhood areas.
3. In general, local optimal solutions within one or more neighborhood structures are relatively close to each other.

Although the last one is not really a fact for all optimization problems, it is an empirical observation implying that a local optimal solution may carry some useful information about other local optimal solutions and even the global optimum.

3 Differential Evolution

Evolutionary Algorithms (EAs) are a class of algorithms inspired by the natural selection. EAs are global population-based search methods incorporating evolutionary operators including recombination, modification and selection. Differential Evolution (DE) is the most recently introduced EA proposed by Storn and Price [12] . DE which is designed to solve continuous optimization problems shows remarkable performance on various continuous optimization problems such as space trajectory optimization [13] and multi-area economic dispatch [14].

The key characteristic of DE which makes it a popular EA is its effective solution space exploration. DE incorporates a number of mathematical equations in order to highly explore the solution space. DE defines each solution as a d-dimensional vector of real numbers and uses its mutation and crossover equations to generate offspring. Although DE has a strong exploration strategy, due to its inefficient exploitation mechanism it is not able to perform well on complex optimization problems. Therefore, DE is a good candidate to be joined with a local search method. Recently, a number of successful hybrid DEs are reported in the literature such as the combination of a DE and an adaptive local search published by Noman and Iba [15].

As mentioned before, DE was designed to deal with continuous optimization problems. In order to apply it on combinatorial optimization problems, two approaches have been introduced. The first approach is to incorporate a transformation procedure such that each solution can be mapped from continuous domain to discrete one and vice versa. It should be noted that the mapping from continuous domain to discrete must be a many-to-one mapping ($n \rightarrow 1$). Incorporating this strategy provides a continuous domain for DE to be applied on. There are a number of successful applications of this approach on permutation optimization problems such as the ones published by Onwubolu and Davendra

[16], Qian *et al.* [17], and Zhang and Wu [18]. The second approach is to modify DE's operators to be applicable in a discrete domain. Modifying DE's operators makes it a specific algorithm for the problem domain and there is no guarantee to obtain the same performance. Two successful modified DEs are published by Pan *et al.* [19] and Wang *et al.* [20] to solve flow-shop scheduling. As presented in details in Section 6, the former approach is incorporated in our proposed methods.

All the components of DE are defined mathematically. The solution space S is a d-dimensional domain of real numbers and each solution s is determined as a d-dimensional vector:

$$S = D_0 \times D_1 \times ... \times D_{d-1} \tag{1}$$

$$s = [x_0, x_1, ..., x_{d-1}], \qquad x_j \in D_j \tag{2}$$

where x_j represents the value of solution s for dimension j ranging from 0 to $d-1$.

DE starts with an initial population of randomly generated solutions. The population is evolved over a number of generations. A solution within the population so-called target vector is denoted by the generation number and an index. For instance, target vector $X_{i,g}$ represents the i^{th} target vector of generation g:

$$X_{i,g} = [x_{0,i,g}, \quad x_{1,i,g}, \quad ..., \quad x_{d-1,i,g}] \tag{3}$$

where $x_{j,i,g}$ represents the value of target vector $X_{i,g}$ for dimension j ranging from 0 to $d-1$.

The recombination and modification operators of DE are also defined as mathematical formulae. The mutation operator applies on a target vector $X_{i,g}$ and generates a new vector which is called mutant vector denoted by $V_{i,g}$. The basic mutation equation is presented in Equation 4:

$$V_{i,g} = X_{r1,g} + F \times (X_{r2,g} - X_{r3,g}) \tag{4}$$

where $X_{r1,g}$, $X_{r2,g}$, and $X_{r3,g}$ are three different randomly selected target vectors. $X_{r1,g}$ is called the base vector and the other two are called perturbing vectors. F is a scale factor to determine how much to perturb the base vector.

There are various mutation equation incorporated by different researchers. Each equation is determined based on its base vector and the number of perturbations. The basic mutation equation illustrated in Equation 4, for instance, is called $DE/rand/1$ which means that the base vector is a randomly selected target vector and the perturbation is done only one time. Four more equations are presented by Price *et al.* [21] which are presented as follows:

$DE/rand/2$:

$$V_{i,g} = X_{r1,g} + F_1 \times (X_{r2,g} - X_{r3,g}) + F_2 \times (X_{r4,g} - X_{r5,g}) \tag{5}$$

$DE/best/1$:

$$V_{i,g} = X_{best,g} + F \times (X_{r1,g} - X_{r2,g}) \tag{6}$$

$DE/best/2:$

$$V_{i,g} = X_{best,g} + F_1 \times (X_{r1,g} - X_{r2,g}) + F_2 \times (X_{r3,g} - X_{r4,g}) \tag{7}$$

$DE/current-to-best/1:$

$$V_{i,g} = X_{i,g} + F_1 \times (X_{best,g} - X_{i,g}) + F_2 \times (X_{r1,g} - X_{r2,g}) \tag{8}$$

where $X_{r1,g}$, $X_{r2,g}$, $X_{r3,g}$, $X_{r4,g}$, and $X_{r5,g}$ are randomly selected target vectors, $X_{best,g}$ denotes the best found solution so far, and F, F_1, and F_2 are scale factors.

Equation 9 represents one more mutation equation which is introduced by Wisittipanich and Kachitvichyanukul [22]:

$DE/localbest/1:$

$$V_{i,g} = X_{ilbest,g} + F \times (X_{r1,g} - X_{r2,g}) \tag{9}$$

where $X_{r1,g}$ and $X_{r2,g}$ are randomly selected target vector, $X_{ilbest,g}$ denotes the local best solution of target vector $X_{i,g}$, and F is a scale factor.

These equations use different base vectors and different levels of perturbation. The more important point is that although Equations 8 and 9 incorporates the target vector $X_{i,g}$ either explicitly or implicitly, other equations generate the mutant vector $V_{i,g}$ regardless of the target vector $X_{i,g}$. Nevertheless, the crossover operator incorporates both the target vector $X_{i,g}$ and the mutant vector $V_{i,g}$ to generate a trial vector $Z_{i,g}$. The most popular crossover operator for DE is binomial crossover which is illustrated in Equation 10:

$$z_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } r_j \le Cr \text{ or } j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{10}$$

where r_j is a random number uniformly distributed in interval $[0, 1)$ selected for the j^{th} dimension, Cr is the crossover probability which could be either fixed or dynamic, and j_{rand} is a randomly selected dimension to ensure that the trial vector $Z_{i,g}$ differs from target vector $X_{i,g}$ at least in one dimension.

After generating trial vectors, a selection function is incorporated by DE to select the better solutions for the next generation. Comparing the target vector $X_{i,g}$ and the trial vector $Z_{i,g}$, this function selects the one with the better objective value as a target vector for the next generation denoted by $X_{i,g+1}$.

$$X_{i,g+1} = \begin{cases} Z_{i,g} & \text{if } f(Z_{i,g}) \le f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \tag{11}$$

4 Multi-population Differential Evolution

In Multi-Population Differential Evolution (MP-DE), the whole population is divided into a number of sub-populations and each sub-population is evolved by a local DE. Local DEs may communicate with each other in order to exchange knowledge. The main reason to incorporate a number of sub-populations instead of a single population is to decrease the chance of premature convergence. In fact, this strategy helps the method to maintain the population diversity and consequently it increases the chance of exploring unvisited regions.

There are various MP-DEs with different characteristics proposed in the literature. A parallel DE is proposed by Tasoulis *et al.* [23] in which the local DEs communicate with each other by migrating their best found solutions in a ring topology. The migrated solutions replace the randomly selected solutions in the destination sub-populations. Tasgetiren and Suganthan [24] proposed a MP-DE which incorporates a re-grouping strategy. The mutation strategy in this method selects the random target vectors from the whole population instead of just the corresponding sub-population.

In addition to the solution migration, there are a number of strategies for MP-DE in which only the DE parameters are exchanged among the sub-populations. Yu and Zhang [25] proposed a MP-DE in which in each generation the successful local DEs send their own parameters to other local DE to adjust theirs. The factor value in mutation equation and the crossover probability are considered as the exchanging control parameters in this method.

Furthermore, there are a number of strategies where no exchange occurs at all. The MP-DE proposed by Mendes and Mohais [26] which is called DynDE defines an acceptable distance between the best solutions of different sub-populations as a threshold. If such a distance gets smaller than the threshold, one of the sub-populations will be re-initialized.

5 Job Shop Scheduling Problem

In order to evaluate our proposed methods, the Job Shop Scheduling Problem (JSSP) is selected as our test bed. Being applicable in various research areas makes JSSP a well-known class of combinatorial optimization problems. JSSP is defined as the process of sequencing a number of jobs to be completed on a number of machines in order to utilize the resources as efficient as possible. The most popular objective in JSSP is to decrease the total time required to perform all the existing jobs. Therefore, the goal is to minimize the maximum completion time of all the jobs, called the *makespan*.

Garey *et al.* [27] proved that the JSSP problems including more than two machines are NP-complete which implies that there is no exact algorithm capable to find the optimal solution for all the sample problems in an acceptable time. Since JSSP is a complex permutation optimization problem which is still an open problem, it is a great case study to evaluate the new proposed methods.

Although there are various versions of JSSP represented in literature, in general, JSSP is defined by Baker [28] as a task of scheduling N jobs denoted by J_i to be processed on M machines denoted by m_k, where i is the job index and k is the machine index ranging from 1 to N and 1 to M, respectively. Each job consists of a number of operations which have to be processed in a pre-defined sequence. Each operation is denoted by O_{ij} where i is the job index and j is the operation index in the i^{th} job. In classical JSSP, there are a number of rules which may be partially shared with other types of JSSP. These rules are presented as follows:

1. Jobs are independent to each other and are available at the beginning.
2. There is no due date for any jobs.
3. All the jobs have the same number of operations which is equal to the number of applicable machines.
4. Each machine processes only one operation of a job which cannot be interrupted.
5. There is only one applicable machine for each operation such that there is no machine selection flexibility for operations.
6. The processing time of each operation on its applicable machine is known and the machine set up time and the movement time between machines are considered negligible.

A sample classical JSSP is illustrated in Table 1 in two different formats, namely job-based and machine-based. The sample problem is a 3×3 problem including 3 jobs to be processed on 3 machines. The table represents the applicable machine for each operation and its corresponding processing time. The second operation of the first job (O_{12}), for instance, has to be processed on the first machine (m_1) for 3 time units. A sample schedule for this problem is depicted in Fig. 1. The sample schedule has the makespan of 12 which is not the minimum makespan for this sample problem.

Table 1. A sample 3×3 classical JSSP

Jobs	Operation Index		
	O_1	O_2	O_3
J_1	m_3,3	m_1,3	m_2,2
J_2	m_1,2	m_2,3	m_3,2
J_3	m_1,2	m_2,3	m_3,2

Machines	Jobs		
	J_1	J_2	J_3
m_1	O_{12},3	O_{21},2	O_{31},2
m_2	O_{13},2	O_{22},3	O_{32},3
m_3	O_{11},3	O_{23},2	O_{33},2

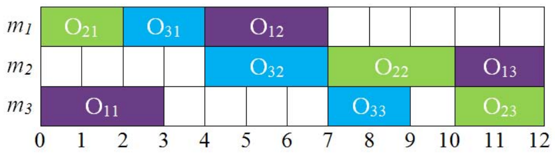


Fig. 1. A sample schedule for the sample problem

There are a number of important concepts in JSSP such that reaching the optimal solution without considering these concepts is almost impossible. The most important concepts are critical paths, critical blocks and critical operations. A critical path is the longest path of consecutive operations in a schedule starting from time zero and ending at the makespan. A schedule has at least one critical path. Critical paths are important because they determine the makespan and the only way to decrease the makespan is to break all the critical paths. Therefore, in order to define efficient neighborhood structures for local search heuristics, breaking the critical paths should be considered as an approach to reach better solutions.

The operations on the critical paths are called critical operations. In other words, critical operations are such operations that any delay in their processing increases the makespan of the whole schedule. A critical operation may belongs to more than one critical paths. A sequence of adjacent critical operations on the same machine is called a critical block.

In the sample schedule presented in Fig. 1, for example, there are two critical paths including:

$$CriticalPaths : \begin{cases} O_{21} \prec O_{31} \prec O_{32} \prec O_{22} \prec O_{13} \\ O_{21} \prec O_{31} \prec O_{32} \prec O_{22} \prec O_{23} \end{cases} \tag{12}$$

Therefore, there are 6 critical operations in the sample schedule where 4 of them are on two different critical paths. The critical blocks are as follows.

$$CriticalBlocks : \begin{cases} m_1 : O_{21} \prec O_{31} \\ m_2 : O_{32} \prec O_{22} \prec O_{13} \\ m_3 : O_{23} \end{cases} \tag{13}$$

In fact, there is only one critical operation on the third machine and there is no sequence of critical operations. Therefore, considering it as a critical block does not have any effect on the process of optimization. The first and the last operations in a critical block are called block head and block rear, respectively, and others are called internal operations.

In addition to the critical operations, there are a number of terms which should be defined clearly. The following terms are used later to describe neighborhood structures. There are two different kinds of operation sequence in a schedule including job operation sequence and machine operation sequence. The former determines the sequence of operations of a job which is predefined in classical JSSP, while the latter one represents the sequence of operations which have to be processed on a specific machine. The adjacent operations, namely the previous operation and the next one, of an operation in a sequence are called its predecessor and successor operations, respectively. In fact, in a job operation sequence they are called Job-Predecessor operation and Job-Successor operation of an operation $O_{i,j}$ denoted by $JP(O_{i,j})$ and $JS(O_{i,j})$, respectively. $MP(O_{i,j})$ and $MS(O_{i,j})$ also represent Machine-Predecessor and Machine-Successor operations of an operation $O_{i,j}$, respectively. Since the operation sequence for each job is predefined, the following statements are always right, provided $O_{i,j+1}$ and $O_{i,j-1}$ exist:

$$JS(O_{i,j}) = O_{i,j+1}$$

$$JP(O_{i,j}) = O_{i,j-1}$$

Another valuable concept in JSSP is active schedule which is very useful to limit the solution space. An active schedule is defined by Croce *et al.* [29] as a schedule which does not have any operation that can be started earlier without delaying the process of another operation. Based on this definition, an optimal solution is more likely an active solution and even if it is not, it has an equivalent active schedule which is optimal as well. It should be noted here that an equivalent schedule is a schedule with the same makespan and the same critical paths which could have one or more different machine operation sequences. Each active schedule may have many equivalent non-active schedules and therefore the solution space of active schedules is much smaller than the main solution space. Consequently, exploring the active solution space is more efficient compared to searching in the whole solution space. Various strategies are represented in order to incorporate the active schedule concept such as the gap reduction rule proposed by Hasan *et al.* [30] and the priori knowledge introduced by Becerra and Coello [31].

6 Proposed Methods

As mentioned before, VNS has a powerful exploitation strategy which suffers from its inefficient search space exploration. In order to enhance VNS, we published recently a combination of a GA and a VNS [9]. The results show that incorporating a population-based global search improves the results of VNS applications. In this article, more explorative methods are joined with VNS in order to illustrate their effects on the final results. It is expected that the results should be improved more compared to ones recently published [9]. Two methods are proposed in this article which are the combination of VNS with two different DEs. The first proposed method incorporates a simple DE while the second one benefits from a MP-DE which has a more explorative mechanism.

In our proposed methods, VNS is combined with EAs in order to improve its performance. As defined by Moscato [11], these combinations are considered as Memetic Algorithms (MAs). In general, a MA is defined as a combination of a population-based global search and a local search heuristic to solve optimization problems. The population-based global search provides an effective solution space exploration for a MA and the local search highly exploits the promising regions. Therefore, the performance of a MA should be higher than each of the combined methods. Various successful applications of MAs have been reported in the literature such as the MAs published by Gao *et al.* [32], Caumond *et al.* [33], and Chiang *et al.* [34] to deal with scheduling problems. Since both proposed methods in this article as well as the methods published previously [9] are instances of MA, in order to differentiate them in this article they are called based on their combination as VNS+DE, VNS+MPDE and VNS+GA, respectively.

The details of both proposed methods are represented in the following subsections starting with the description of solution representation in Subsection 6.1. The neighborhood structures incorporated in our proposed VNS are described in Subsection 6.2 followed by the definition of the genetic operators of the proposed DE and MPDE in Subsection 6.3. Finally, the frameworks of both proposed methods are illustrated in Subsection 6.4.

6.1 Solution Representation

As mentioned before, in order to apply DE on combinatorial optimization problems, there are two approaches. In our proposed methods, the transformation strategy is incorporated. Therefore, two different solution representations are considered; one in discrete domain and one in continuous. The proposed DE and MPDE deal with the representation in a continuous domain, while in order to evaluate each solution it should be transformed to a permutation domain to be considered as a JSSP solution.

Various representations with different characteristics are proposed for JSSP. Permutation with repetition representation introduced by Bierwirth [35] is one of the well-known representations. As an operation-based representation, this representation encodes a schedule based on the sequence of the operations into a string of digits. The length of the string equals to the total number of operations in a scheduling problem and each operation is denoted by its job index in the string. Therefore the operations of the same job are denoted by the same index. Considering the operation dependency within a job operation sequence, the operations with the same index are differentiated based on the occurrence number of the corresponding index. For instance, the second occurrence of the third job's index denotes the second operation of the third job (O_{32}). A sample solution for the sample problem illustrated in Table 1 is represented in permutation with repetition representation as follows:

$$\{1, 2, 3, 1, 3, 2, 3, 2, 1\}$$

The decoding of this sample solution results in the following operation sequence:

$$O_{11} \prec O_{21} \prec O_{31} \prec O_{12} \prec O_{32} \prec O_{22} \prec O_{33} \prec O_{23} \prec O_{13}$$

This operation sequence generates the following schedule which is depicted as a Gantt chart in Fig. 1.

$$m_1 : O_{21} \prec O_{31} \prec O_{12}$$

$$m_2 : O_{32} \prec O_{22} \prec O_{13}$$

$$m_3 : O_{11} \prec O_{33} \prec O_{23}$$

The key advantage of permutation with repetition representation is that all the possible permutations in this representation are feasible solutions. Therefore, the decoding mechanism is straightforward such that no repair mechanism is required. The main drawback of this representation is its inefficient many-to-one mapping ($n \rightarrow 1$). In fact, there could be a huge number of different permutations which are decoded to the same schedule. The schedule illustrated in Fig. 1 might be decoded from the following permutations (not limited to):

$$\{1,2,3,1,3,2,3,2,1\} \quad \{1,2,3,3,1,2,3,2,1\}$$
$$\{1,2,3,3,2,1,3,2,1\} \quad \{1,2,3,3,2,3,1,2,1\}$$
$$\{1,2,3,3,2,3,2,1,1\} \quad \{2,1,3,3,2,3,2,1,1\}$$
$$\{2,3,1,3,2,3,2,1,1\} \quad \{2,3,3,1,2,3,2,1,1\}$$
$$\{2,3,3,2,1,3,2,1,1\} \quad \{2,1,3,1,3,2,3,2,1\}$$
$$\{2,3,1,1,3,2,3,2,1\} \quad \{2,3,1,3,1,2,3,2,1\}$$
$$\{2,3,1,3,2,1,3,2,1\} \quad \{2,3,1,3,2,3,1,2,1\}$$

However, due to its coding and decoding efficiency, permutation with repetition representation is incorporated by various researchers. In our proposed method, we also use this representation.

As a representation in continuous domain, random key representation is selected for our proposed methods. In this representation, a solution is represented as a vector of real numbers. Each dimension in this vector corresponds to one operation. The following vector illustrates a sample solution for the sample problem described in Table 1.

$$\{0.49, 0.98, 0.38, 0.42, 0.73, 0.51, 0.48, 0.89, 0.63\}$$

The corresponding value for each operation in this sample solution is presented as follows. The minimum value, for instance, corresponds to O_{13} which means that this operation should be started first. Due to the operation dependency within a job operation sequence, O_{11} is processed first instead of O_{13}.

Operation	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
Job Index	1	1	1	2	2	2	3	3	3
Random Key	0.49	0.98	0.38	0.42	0.73	0.51	0.48	0.89	0.63

In order to transform a solution from random key representation to permutation with repetition representation, Smallest Position Value (SPV) rule [36] is incorporated, in which the job indices should be sorted based on their corresponding random key values ascendingly which is as follows:

Random Key	0.38	0.42	0.48	0.49	0.51	0.63	0.73	0.89	0.98
Job Index	1	2	3	1	2	3	2	3	1

The sorted string of job indices is the equivalent solution in the discrete domain which is

$$\{1,2,3,1,2,3,2,3,1\}$$

where its encoded operation sequence is presented as follows.

$$O_{11} \prec O_{21} \prec O_{31} \prec O_{12} \prec O_{22} \prec O_{32} \prec O_{23} \prec O_{33} \prec O_{13}$$

In addition to the fitness evaluation function, VNS works in the discrete domain as well. Therefore, the results of VNS should be transformable to the continuous domain. In order to do so, each solution in discrete domain updates its random key when a swapping or insertion occurs, accordingly.

6.2 Neighborhood Structures

In order to be able to compare the proposed methods with the published ones [9], the same neighborhood structures are incorporated in this article. In our recently published methods [9], two neighborhood structures $N4$ and $N5$ were incorporated. These structures are two of the six popular neighborhood structures reviewed by Blazewicz *et al.* [37] which are denoted by $N1$ to $N6$ by the authors. These neighborhood structures are described briefly as follows:

– Neighborhood Structure $N1$: This structure defines the largest neighborhood area by considering the swapping of two adjacent critical operations as a valid move [38].
– Neighborhood Structure $N2$: This neighborhood structure considers the swapping of two critical operations p and q as a valid move if either p is a block head or q is a block rear. In order to improved the neighbor solutions two additional moves are also considered which include the swapping of $MP(JP(p))$ and $JP(p)$ and the swapping of $JS(q)$ and $MS(JS(q))$ [39].
– Neighborhood Structure $N3$: Considering p and q are two adjacent critical operations, this structure looks into all permutations of three operations $\{MP(p), p, q\}$ as well as three operations $\{p, q, MS(q)\}$ in which p and q are swapped [40]. Since the neighborhood area of this structure is very large, a limited version is introduced which is called $N3'$ in which either p or q should be a block end.
– Neighborhood Structure $N4$: Moving an internal operation to the very beginning or to the very end of a block is considered as a valid move in this structure [40].
– Neighborhood Structure $N5$: This structure only swaps the first two operations or the last two operations of a critical block which makes the smallest neighborhood area [41].
– Neighborhood Structure $N6$: This structure is an extension of all previously described neighborhood structures. The valid moves in this structure include moving q right before p if $JP(p)$ belongs to the critical path and moving p right after q if $JS(q)$ belongs to the critical path, given p and q as two critical operations on a critical block [42].

As mentioned above, only two neighborhood structures $N4$ and $N5$ are considered in our proposed methods without any pre-processing or post-processing procedures. The structure of the incorporated VNS is exactly the same as the one recently published [9] which includes a *Shake* method followed by a *LocalSearch* method. The *Shake* method consists of four consecutive random moves with respect to the following neighborhood structures.

$$Shake : N5 \rightarrow N4 \rightarrow N5 \rightarrow N4$$

The *LocalSearch* method incorporates both the regular neighborhood search and the nested neighborhood search. In regular one, both neighborhood structures $N4$ and $N5$ are considered while for the nested search only the smaller neighborhood ($N5$) is incorporated. In terms of local search strategy, in order to save the computational time, the first improvement strategy is incorporated instead of the best improvement one [9].

6.3 Genetic Operators

As DE is incorporated in order to increase the exploration power, it should be designed using more explorative evolutionary operators. The most explorative mutation strategy is $DE/rand/1$ which is illustrated in Equation 4. Incorporating this strategy in our experiment declares that this strategy is not useful for JSSP because it does not keep any information regarding previous generations. Other strategies presented in Equations 5 through 9 are more exploitative than explorative. Therefore, a new mutation strategy is introduced in this article which is called $DE/current/1$ presented as follows:

$$V_{i,g} = X_{i,g} + F \times (X_{r1,g} - X_{r2,g}) \tag{14}$$

where $X_{i,g}$ is the current target vector, $X_{r1,g}$ and $X_{r2,g}$ are two different randomly selected target vectors, and F is a scale factor.

Although this strategy incorporates the existing target vectors, it is more explorative compared to other strategies while benefiting from the evolved solutions instead of just random ones. The combination of both $DE/current/1$ and $DE/rand/1$ is also evaluated on some JSSP benchmark problems. In this combination, for a top portion of a population $DE/current/1$ is incorporated while for the rest of the population $DE/rand/1$ is used. It was expected to obtain good performance for this combination, but it returns poor results. Therefore, only $DE/current/1$ is incorporated as the mutation strategy in the proposed DE and MP-DE with a small difference. In order to provide a more explorative mutation strategy, a modification is considered for MP-DE which will be described later in this section. To incorporate more information of the best found solution so far in case of no improvement for a number of generations, the mutation strategy switches to $DE/current - to - best/1$ illustrated in Equation 8.

For the crossover operator, the binomial crossover illustrated in Equation 10 is considered for both proposed methods exactly the same. Like for the crossover operator, both proposed methods incorporate the same selection function which is presented in Equation 11. It should be noted here that in order to consider more information in selection procedure, instead of a simple evaluation function, a Priority-Based Fitness Function (PBFF) [43] is incorporated. The idea of this function comes from the fact that in simple evaluation functions when a tie happens the winner is selected arbitrarily while it could be selected based on a comparison with respect to another factor. In PBFF, a number of factors are defined with different priorities and then the comparison procedures considers lower priority factors in case of ties in higher priority factors.

In our proposed methods, a PBFF is incorporated such that the first priority factor is the makespan and the second one is the number of critical machines. Based on this fitness function, if two schedules have the same makespan, the selection function selects the one with the lower number of critical machines. If they have the same number of critical machines, one of them will be selected randomly.

6.4 Frameworks

The first proposed method is a combination of DE and VNS, so-called VNS+DE. The framework of the proposed DE+VNS is illustrated in Fig. 2 which starts with an initial population of random solutions (line 03). *PopSize* denotes the size of the population which is fixed during the evolution. In each generation, first DE evolves the population and then VNS applies on a top best portion of the evolved population. DE incorporates its mutation, crossover and selection operators to explore the solution space (lines 05 through 09). Then the exploitation of the promising regions (the neighborhood of the top best solutions) is conducted by VNS (lines 10 through 12). The top best solutions are determined using *TopBest* parameter which is fixed for all generations. In our proposed method, in order to find out the top best solutions the whole population is sorted in each generation. After the application of VNS, its results combined with the rest of the population provides the population for the next generation denoted by P_{g+1} (line 13). This routine continues until the criteria are met (line 15). A maximum number of generations is considered as our termination criterion denoted by *MaxGen*.

The second proposed method is a combination of a VNS and a Multi-Population Differential Evolution denoted by VNS+MPDE. Although this method is similar to the proposed VNS+DE, it benefits from the concept of multiple populations. In this method, first the whole population is divided into a number of sub-populations denoted by $SubPopsNo$ and then each sub-population is evolved by a local DE.

As mentioned before, the local DEs incorporate both $DE/current/1$ and $DE/current-to-best/1$ illustrated in Equations 14 and 8, respectively. The point is that in order to enhance the solution space exploration the random target vectors in both mutation strategies for each sub-population is selected from other sub-populations. This mechanism increases the population diversity and improves the chance of reaching unexplored regions.

7 Results

Both proposed algorithms are implemented using the *java* programming language version 1.6.0.18 and the experiments are conducted on a system with

```
PROCEDURE: VNS+DE
INPUT: Algorithm Parameters and Problem Specification
OUTPUT: Optimal or Near-Optimal Solutions
01  BEGIN
02     $g \leftarrow 0$
03     Generate an initial population $P_0$ with
              $PopSize$ random solutions
04     REPEAT
05        FOR ( Each Target Vector $X_{i,g}$ of $P_g$)
06           $V_{i,g} \leftarrow Mutate(X_{i,g})$
07           $Z_{i,g} \leftarrow Crossover(X_{i,g}, V_{i,g})$
08           $X_{i,g+1} \leftarrow Selection(X_{i,g}, Z_{i,g})$
09        END FOR
10        FOR ( Each Top Target Vector $X_{i,g+1}$ )
11           $L_{i,g+1} \leftarrow VNS(X_{i,g+1})$
12        END FOR
13        Construct population $P_{g+1}$
14        $g \leftarrow g + 1$
15     UNTIL (termination criteria are met)
16     Output The Best Found Solution
17  END
```

Fig. 2. The framework of the VNS+DE

Intel(R) Core(TM)2Quad 2.50*GHz CPU* and 8.00*GB RAM*. The algorithm parameters presented in Table 2 are adjusted through extensive experiments. A population of 1000 individuals with the top 100 best individuals are evolved for 200 generations.

Table 2. Parameters of the Proposed Methods

Parameter	Value
$MaxGen$	200
$PopSize$	1000
$TopBest$	100
$SubPopsNo$	5
$Mutate(X_{i,g})$	Equations 14 and 8
$Crossover(X_{i,g}, V_{i,g})$	Equation 10
$Selection(X_{i,g}, Z_{i,g})$	Equation 11 with a PBFF

In order to evaluate the performance of the both proposed methods, classical JSSP is considered as the our test bed. The data set introduced by Lawrence [44] is one of the well-known benchmark for classical JSSP which includes 40 different problems with various sizes. Both proposed methods are applied on all the 40 test problems for 50 independent runs. Both proposed methods find the optimal solution for 28 test problems in every run. The results of the experiments on the rest of the problems which are more challenging are presented in Table 3.

In order to have a fair comparison with the recently published methods [9], these methods including the simple VNS and the combination of the VNS with a GA are also re-applied on the benchmark problems for 50 independent runs. The results of these experiments are also presented in Table 3 which is slightly different than the ones presented in the previously published article. This small difference is due to the fact that the published results are experimented for 10 independent runs while the new ones are averaged over 50 runs.

As presented in Table 3, the simple VNS can find the optimal solution for 5 challenging test problems out of 12, while the VNS+GA and the VNS+DE are able to find the optimal solution for the same 6 problems, and the VNS+MPDE is able to offer the optimal solution for 7 test problems. In total, the simple VNS, the VNS+GA, the VNS+DE and the VNS+MPDE find the optimal solutions for 33, 34, 34 and 35 test problems out of 40, respectively.

In addition to offering the optimal solution for more test problems, the VNS+MPDE offers more acceptable solutions compared to the other methods. In other words, not only the VNS+MPDE can find better solutions with a lower makespan, but also the range of its results for 50 independent runs is smaller compared to the result ranges of other methods. The average, median and worst results presented in Table 3 which are calculated over the 50 runs proves this claim. For test problem $la38$, for instance, not only the best solution is found by the VNS+MPDE, but also the range of its obtained solutions in the 50 runs is from 1202 to 1220 (19 time units wide) which is thinner compared to this range for other methods specially the simple VNS which is from 1207 to 1265 (59 time units wide).

In order to compare all the four method over all the 12 challenging test problem, the Error Rate (ER) parameter is incorporated which is defined as follows:

$$ER = \frac{C - LB}{LB} \times 100\%$$

where C is the makespan of a solution found by the methods and LB is the best-known solution. The ER values for the best, average and worst solutions for each methods are calculated and represented in Table 4. In this table, the zero errors are emphasized with bold face and the average ERs over all the 12 test problems are presented in the last three rows. Comparing all the ERs illustrates that the proposed VNS+MPDE is the most reliable method to find acceptable solutions. The next reliable method is the proposed VNS+DE, and the least reliable one is the simple VNS. This statement is represented in a graph in Fig. 3 more clearly.

This comparison proves our hypothesis mentioning that combining a VNS with more explorative methods offers better solutions. Explorative methods increases the chance of finding the promising regions in a solution space which can be exploited by the VNS. Exploiting more promising regions improves the final results which increases the reliability of finding acceptable solutions.

In order to evaluate the performance of both proposed methods, they are also compared with the state-of-the-art methods in this area. Three recently

Table 3. Results on the challenging problems of LA Benchmark

Problem	Method	Best	Average	SD	Median	Worst	Hit
*LA*20	VNS	**902**	906.72	1.13	907.0	907	2
	VNS + GA	**902**	906.46	1.49	907.0	907	3
10 × 10	VNS + DE	**902**	906.30	1.63	907.0	907	3
902	VNS + MPDE	**902**	905.98	1.95	907.0	907	7
*LA*21	VNS	**1046**	1058.18	7.96	1056.0	1077	4
	VNS + GA	**1046**	1053.28	3.85	1054.0	1060	8
15 × 10	VNS + DE	**1046**	1049.86	3.49	1049.0	1056	16
1046	VNS + MPDE	**1046**	1049.40	3.11	1048.0	1054	18
*LA*24	VNS	**935**	944.42	5.60	944.0	960	1
	VNS + GA	**935**	943.00	3.52	943.0	949	1
15 × 10	VNS + DE	**935**	941.16	2.44	941.0	945	2
935	VNS + MPDE	**935**	940.98	2.38	941.0	944	3
*LA*25	VNS	978	984.62	4.38	984.0	1000	2
	VNS + GA	**977**	982.48	2.22	983.0	987	2
15 × 10	VNS + DE	**977**	981.18	2.21	982.0	984	3
977	VNS + MPDE	**977**	979.50	1.88	979.0	983	7
*LA*27	VNS	1238	1251.44	8.66	1251.0	1265	3
	VNS + GA	1238	1251.60	6.58	1252.0	1263	4
20 × 10	VNS + DE	1236	1248.58	6.74	1250.0	1262	1
1235	VNS + MPDE	**1235**	1248.12	6.78	1250.0	1256	3
*LA*28	VNS	**1216**	1219.76	6.49	**1216**.0	1234	29
	VNS + GA	**1216**	1218.72	3.61	1217.0	1228	23
20 × 10	VNS + DE	**1216**	1216.68	1.56	**1216**.0	1223	37
1216	VNS + MPDE	**1216**	1216.26	0.60	**1216**.0	1218	41
*LA*29	VNS	1169	1188.26	10.96	1190.5	1210	1
	VNS + GA	1163	1180.88	7.29	1180.5	1194	1
20 × 10	VNS + DE	1163	1177.12	6.47	1177.5	1190	2
1152	VNS + MPDE	1163	1176.86	6.69	1178.0	1187	2
*LA*36	VNS	1281	1291.88	5.77	1291.0	1309	7
	VNS + GA	1277	1289.48	6.35	1291.0	1302	4
15 × 15	VNS + DE	1274	1287.78	6.12	1291.0	1298	1
1268	VNS + MPDE	1271	1284.70	5.94	1286.0	1291	1
*LA*37	VNS	**1397**	1420.22	11.08	1420.0	1442	1
	VNS + GA	**1397**	1403.40	8.07	1401.0	1433	2
15 × 15	VNS + DE	**1397**	1409.46	6.99	1408.5	1421	5
1397	VNS + MPDE	**1397**	1408.74	6.08	1408.0	1418	5
*LA*38	VNS	1207	1229.34	16.72	1229.5	1265	1
	VNS + GA	1207	1220.52	5.76	1219.0	1237	1
15 × 15	VNS + DE	1206	1215.88	3.62	1216.0	1225	1
1196	VNS + MPDE	1202	1215.38	3.76	1216.0	1220	1
*LA*39	VNS	1240	1246.76	4.11	1248.0	1256	4
	VNS + GA	1240	1248.40	3.17	1249.0	1253	5
15 × 15	VNS + DE	1238	1247.52	2.60	1249.0	1249	2
1233	VNS + MPDE	1238	1246.80	3.25	1248.0	1249	3
*LA*40	VNS	1231	1240.98	4.83	1242.0	1249	1
	VNS + GA	1230	1238.94	5.40	1239.0	1252	2
15 × 15	VNS + DE	1228	1235.62	3.49	1234.5	1242	1
1222	VNS + MPDE	1228	1235.26	3.99	1234.0	1242	2

Table 4. Comparison of the best, average and worst results of the proposed methods

Problem		VNS	VNS+GA	VNS+DE	VNS+MPDE
$LA20$	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
	Average	0.52%	0.49%	0.48%	0.44%
	Worst	0.55%	0.55%	0.55%	0.55%
$LA21$	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
	Average	1.16%	0.70%	0.37%	0.33%
	Worst	2.96%	1.34%	0.96%	0.76%
$LA24$	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
	Average	1.01%	0.86%	0.66%	0.64%
	Worst	2.67%	1.50%	1.07%	0.96%
$LA25$	Best	0.10%	**0.00%**	**0.00%**	**0.00%**
	Average	0.78%	0.56%	0.43%	0.26%
	Worst	2.35%	1.02%	0.72%	0.61%
$LA27$	Best	0.24%	0.24%	0.08%	**0.00%**
	Average	1.33%	1.34%	1.10%	1.06%
	Worst	2.43%	2.27%	2.19%	1.70%
$LA28$	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
	Average	0.31%	0.22%	0.06%	0.02%
	Worst	1.48%	0.99%	0.58%	0.16%
$LA29$	Best	1.48%	0.95%	0.95%	0.95%
	Average	3.15%	2.51%	2.18%	2.16%
	Worst	5.03%	3.65%	3.30%	3.04%
$LA36$	Best	1.03%	0.71%	0.47%	0.24%
	Average	1.88%	1.69%	1.56%	1.32%
	Worst	3.23%	2.68%	2.37%	1.81%
$LA37$	Best	**0.00%**	**0.00%**	**0.00%**	**0.00%**
	Average	1.66%	0.46%	0.89%	0.84%
	Worst	3.22%	2.58%	1.72%	1.50%
$LA38$	Best	0.92%	0.92%	0.84%	0.50%
	Average	2.79%	2.05%	1.66%	1.62%
	Worst	5.77%	3.43%	2.42%	2.01%
$LA39$	Best	0.57%	0.57%	0.41%	0.41%
	Average	1.12%	1.25%	1.18%	1.12%
	Worst	1.87%	1.62%	1.30%	1.30%
$LA40$	Best	0.74%	0.65%	0.49%	0.49%
	Average	1.55%	1.39%	1.11%	1.09%
	Worst	2.21%	2.45%	1.64%	1.64%
Avgerage ER	Best	0.42%	0.34%	0.27%	0.22%
	Average	1.44%	1.13%	0.97%	0.91%
	Worst	2.82%	2.01%	1.57%	1.34%

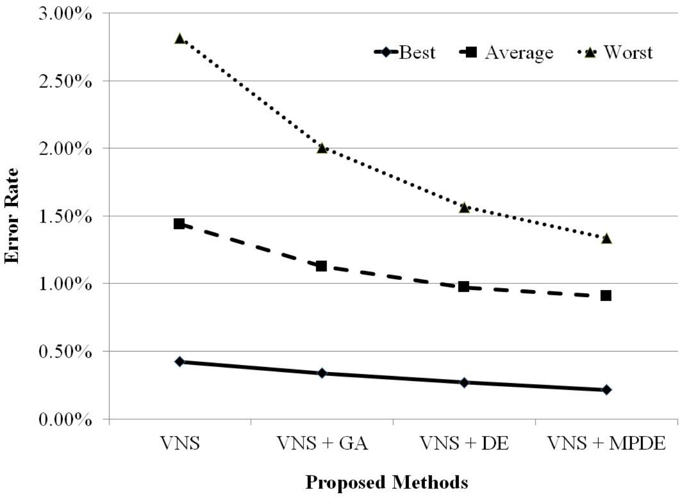


Fig. 3. Comparison of the best, average and worst results of the proposed methods

published methods are considered for this comparison including a hybrid EA proposed by Zobolas *et al.* [45], our published MA [43] and a hybrid GA proposed by Qing-dao-er-ji and Wang [46]. The hybrid EA incorporates VNS as its local search method and uses DE to generate a valuable initial population. Our recently published MA incorporates a GA combined with a local search heuristic. The hybrid GA introduces new genetic operators and incorporates a new local search method specifically designed for JSSP.

The results of this comparison are presented in Table 5 which illustrates the ER values in brackets, the optimal solutions in bold face, and the best solutions with an asterisk (*). Since the hybrid GA [46] cannot find the optimal solution for test problem $LA22$, this problem should be added to the 12 challenging test problems. Therefore for this comparison, 13 test problems are considered and the last row in the table illustrates the average ERs over the 13 test problems. It should be noted here that in the hybrid EA [45] the result for the test problem $LA20$ is not reported. Therefore, excluding this test problem the average ER equals to 0.27%, while considering that it may find the optimal solution for this test problem decreases its average ER to 0.25%. However, our proposed VNS+MPDE offers the best average ER.

The proposed VNS+MPDE can find the optimal solution for 8 test problems out of 13 which is the highest number compared to the other methods. Furthermore, this method offers the best solution for two more test problems. The results of the proposed VNS+MPDE is not better than the others just for 3 test problems $LA29$, $LA36$ and $LA39$. Among these five methods, only three methods can find solutions with ER less than 1 for all test problems which include the hybrid EA [45] and the both proposed methods. Although there is not a significant difference between the proposed methods and the hybrid EA, our proposed VNS+MPDE slightly outperforms all the mentioned state-of-the-art methods.

Table 5. Comparison with different hybrid EAs proposed recently to solve JSSP

Prob.	BK	hEA (2009) [45]	MA (2012) [43]	hGA (2012) [46]	Proposed VNS+DE	Proposed VNS+MPDE
LA20	**902**	-	907 (0.55%)	907 (0.55%)	**902** (0.00%)	**902** (0.00%)
LA21	**1046**	**1046** (0.00%)	1053 (0.67%)	**1046** (0.00%)	**1046** (0.00%)	**1046** (0.00%)
LA22	**927**	**927** (0.00%)	**927** (0.00%)	935 (0.86%)	**927** (0.00%)	**927** (0.00%)
LA24	**935**	**935** (0.00%)	941 (0.64%)	953 (1.93%)	**935** (0.00%)	**935** (0.00%)
LA25	**977**	**977** (0.00%)	984 (0.72%)	981 (0.41%)	**977** (0.00%)	**977** (0.00%)
LA27	**1235**	1236 (0.08%)	1256 (1.70%)	1236 (0.08%)	1236 (0.08%)	**1235** (0.00%)
LA28	**1216**	1224 (0.66%)	1223 (0.58%)	**1216** (0.00%)	**1216** (0.00%)	**1216** (0.00%)
LA29	**1152**	1160* (0.69%)	1187 (3.04%)	1160* (0.69%)	1163 (0.95%)	1163 (0.95%)
LA36	**1268**	**1268** (0.00%)	1276 (0.63%)	1287 (1.50%)	1274 (0.47%)	1271 (0.24%)
LA37	**1397**	1408 (0.79%)	1401 (0.29%)	1407 (0.72%)	**1397** (0.00%)	**1397** (0.00%)
LA38	**1196**	1202* (0.50%)	1208 (1.00%)	**1196** (0.00%)	1206 (0.84%)	1202* (0.50%)
LA39	**1233**	**1233** (0.00%)	1240 (0.57%)	**1233** (0.00%)	1238 (0.41%)	1238 (0.41%)
LA40	**1222**	1229 (0.57%)	1233 (0.90%)	1229 (0.57%)	1228* (0.49%)	1228* (0.49%)
Avg. ER		0.27% (0.25%)	0.87%	0.56%	0.27%	0.22%

8 Conclusions

VNS is one of the most recent metaheuristics proposed to deal with optimization problems. Although it has been successfully applied in various research areas, its mechanism is highly exploitative and it lacks an efficient search space exploration strategy. Recently, we published an article claiming that VNS is highly a local search metaheuristic and in order to have a great performance as a global search it should be combined with an explorative method [9]. In the published article, VNS is combined with a GA and results show that this combination outperforms the simple VNS. In this article, two more explorative methods are combined with VNS in order to improve the final results. The first combined method is DE which is more explorative compared to GA. The second one is a more explorative method which is a DE incorporating multiple populations. Dividing the whole population into a number of sub-populations decreases the chance of premature convergence and encourages the method to look into different regions.

Both proposed combinations are evaluated on the same benchmark. The experiments show that the proposed VNS+MPDE outperforms all the other combinations by offering highly acceptable solutions in every run. While the incorporated VNS is the same for all the combinations, as expected more explorative methods offer better solutions. Compared to the single VNS which returns solutions with different qualities, the joined VNS with more explorative methods show higher reliability in terms of finding acceptable solutions. Although the proposed methods are just introduced to show the effects of combining explorative methods with VNS in its performance, they offer competitive solutions compared to the state-of-the-art methods.

Acknowledgment This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

References

1. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research 24, 1097–1100 (1997)
2. Felipe, A., Ortuno, M.T., Tirado, G.: The double traveling salesman problem with multiple stacks: a variable neighborhood search approach. Computers and Operations Research 36(11), 2983–2993 (2009)
3. Fleszar, K., Osman, I.H., Hindi, K.S.: A variable neighbourhood search algorithm for the open vehicle routing problem. European Journal of Operational Research 195, 803–809 (2009)
4. Fleszar, K., Hindi, K.S.: An effective vns for the capacitated p-median problem. European Journal of Operational Research 191(3), 612–622 (2008)
5. Brimberg, J., Mladenovic, N., Urosevic, D., Ngai, E.: Variable neighborhood search for the heaviest k-subgraph. Computers and Operations Research 36(11), 2885–2891 (2009)
6. Crainic, T., Gendreau, M., Hansen, P., Mladenovic, N.: Cooperative parallel variable neighborhood search for the p-median. Journal of Heuristics 10, 289–310 (2004)
7. Mansini, R., Tocchella, B.: The traveling purchaser problem with budget constraint. Computers and Operations Research 36(7), 2263–2274 (2009)
8. Wang, X., Tang, L.: A population-based variable neighborhood search for the single machine total weighted tardiness problem. Computers and Operations Research Archive 36(6), 2105–2110 (2009)
9. Raeesi N., M.R., Kobti, Z.: Incorporating a genetic algorithm to improve the performance of variable neighborhood search. In: The Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 144–149 (2012)
10. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
11. Moscato, P.: Memetic algorithms: A short introduction. New Ideas in Optimization 14, 219–234 (1999)
12. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 341–359 (1997)
13. Vasile, M., Minisci, E.A., Locatelli, M.: An inflationary differential evolution algorithm for space trajectory optimization. IEEE Transactions on Evolutionary Computation 15(2), 267–281 (2011)
14. Sharma, M., Pandit, M., Srivastava, L.: Reserve constrained multi-area economic dispatch employing differential evolution with time-varying mutation. International Journal of Electrical Power and Energy Systems 33(3), 753–766 (2011)
15. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. IEEE Transactions on Evolutionary Computation 12(1), 107–125 (2008)
16. Onwubolu, G., Davendra, D.: Scheduling flow shops using differential evolution algorithm. European Journal of Operational Research 171(2), 674–692 (2006)
17. Qian, B., Wang, L., Hu, R., Wang, W.L., Huang, D.X., Wang, X.: A hybrid differential evolution method for permutation flow-shop scheduling. The International Journal of Advanced Manufacturing Technology 38(7), 757–777 (2008)

18. Zhang, R., Wu, C.: A hybrid differential evolution and tree search algorithm for the job shop scheduling problem. Mathematical Problems in Engineering 2011, Article ID 390593 (2011)
19. Pan, Q.K., Tasgetiren, M.F., Liang, Y.C.: A discrete differential evolution algorithm for the permutation flowshop scheduling problem. Computers & Industrial Engineering 55(4), 795–816 (2008)
20. Wang, L., Pan, Q.K., Suganthan, P.N., Wang, W.H., Wang, Y.M.: A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Computers & Operations Research 37(3), 509–520 (2010)
21. Price, K., Storn, R., Lampinen, J.: Differential evolution: a practical approach to global optimization (Natural Computing Series). Springer, New York (2005)
22. Wisittipanich, W., Kachitvichyanukul, V.: Two enhanced differential evolution algorithms for job shop scheduling problems. International Journal of Production Research 50(10), 2757–2773 (2012)
23. Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Parallel differential evolution. In: IEEE Congress on Evolutionary Computation (CEC), pp. 2023–2029 (2004)
24. Tasgetiren, M.F., Suganthan, P.N.: A multi-populated differential evolution algorithm for solving constrained optimization problem. In: IEEE Congress on Evolutionary Computation (CEC), pp. 340–354 (2006)
25. Yu, W.J., Zhang, J.: Multi-population differential evolution with adaptive parameter control for global optimization. In: Genetic and Evolutionary Computation Conference (GECCO), Dublin, Ireland, pp. 1093–1098 (2011)
26. Mendes, R., Mohais, A.S.: DynDE: A differential evolution for dynamic optimization problems. In: IEEE Congress on Evolutionary Computation (CEC), vol. 2, pp. 2808–2815 (2005)
27. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1, 117–129 (1976)
28. Baker, K.R.: Introduction to Sequencing and Scheduling. Wiley (1974)
29. Croce, F., Tadei, R., Volta, G.: A genetic algorithm for the job shop problem. Computers in Operations Research 22, 15–24 (1995)
30. Hasan, S., Sarker, R., Essam, D., Cornforth, D.: Memetic algorithms for solving job-shop scheduling problems. Memetic Computing 1, 69–83 (2008)
31. Becerra, R.L., Coello, C.A.C.: A cultural algorithm for solving the job-shop scheduling problem. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation. STUDFUZZ, vol. 167, pp. 37–55. Springer, Heidelberg (2005)
32. Gao, J., Sun, L., Gen, M.: A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers and Operations Research 35(9), 2892–2907 (2008)
33. Caumond, A., Lacomme, P., Tcherneva, N.: A memetic algorithm for the job-shop with time-lags. Computers and Operations Research 35(7), 2331–2356 (2008)
34. Chiang, T.C., Cheng, H.C., Fu, L.C.: NNMA: An effective memetic algorithm for solving multiobjective permutation flow shop scheduling problems. Expert Systems with Applications 38(5), 5986–5999 (2011)
35. Bierwirth, C.: A generalized permutation approach to job shop scheduling with genetic algorithms. OR Spectrum. Special Issue on Applied Local Search 17(2-3), 87–92 (1995)
36. Tasgetiren, M.F., Sevkli, M., Liang, Y.C., Gencyilmaz, G.: Particle swarm optimization algorithm for single-machine total weighted tardiness problem. In: Congress on Evolutionary Computation (CEC), Portland, Oregan, USA, pp. 1412–1419 (2004)

37. Blazewicz, J., Domschke, W., Pesch, E.: The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research 93(1), 1–33 (1996)
38. Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K.: Job shop scheduling by simulated annealing. Operations Research 40(1), 113–125 (1992)
39. Matsuo, H., Suh, C., Sullivan, R.: A controlled search simulated annealing method for the general job shop scheduling problem. In: Working Paper 03-04-88. University of Texas at Austin (1988)
40. Dell'Amico, M., Trubian, M.: Applying tabu search to the job-shop scheduling problem. Annals of Operations Research 41(3), 231–252 (1993)
41. Nowicki, E., Smutnicki, C.: A fast taboo search algorithm for the job shop scheduling problem. Management Science 42(6), 797–813 (1996)
42. Balas, E., Vazacopoulos, A.: Guided local search with shifting bottleneck for job shop scheduling. Management Science 44(2), 262–275 (1998)
43. Raeesi, M.R.N., Kobt, Z.: A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic. Memetic Computing - Special Issue on Optimization on Complex Systems 4(3), 231–245 (2012)
44. Lawrence, S.: Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. Master's thesis, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania (1984)
45. Zobolas, G.I., Tarantilis, C.D., Ioannou, G.: A hybrid evolutionary algorithm for the job shop scheduling problem. Journal of the Operational Research Society 60, 221–235 (2009)
46. González, M.A., Vela, C.R., Varela, R.: A new hybrid genetic algorithm for job shop scheduling problem. Computers and Operations Research 39(10), 2291–2299 (2012)

Self Organising Maps on Compute Unified Device Architecture for the Performance Monitoring of Emergency Call-Taking Centre

Václav Snášel[1], Petr Klement[2], Petr Gajdoš[1], and Ajith Abraham[1,3]

[1] IT4Innovations, Department of Computer Science,
VŠB-Technical University of Ostrava, 708 33 Ostrava-Poruba, Czech Republic
{vaclav.snasel,petr.gajdos}@vsb.cz

[2] Medium Soft a.s., Cihelní 14, 702 00 Ostrava, Czech Republic
petr.klement@mediumsoft.cz

[3] Machine Intelligence Research Labs (MIR Labs),
Scientific Network for Innovation and Research Excellence, WA, USA
ajith.abraham@ieee.org

Abstract. The collaborative emergency call-taking information system in the Czech Republic forms a network of cooperating emergency call centers processing emergency calls to the European 112 emergency number. Large amounts of various incident records are stored in the databases. The data can be used for mining spatial and temporal anomalies, as well as for the monitoring and analysis of the performance of the emergency call- taking system. In this paper, we describe a method for knowledge discovery and visualization targeted at the performance analysis of the system with respect to the organization of the emergency call-taking information system and its data characteristics. The method is based on the Kohonen Self-Organising Map (SOM) algorithm and its extension, the Growing Grid algorithm. To handle the massive data, the growing grid algorithm is implemented in a parallel environment using compute unified device architecture. Experimental results illustrate that the proposed method is very efficient.

Keywords: Emergency Call, Self-Organising Map, Growing Grid, Knowledge Discovery in Databases.

1 Introduction

Emergency call taking in the Czech Republic, is supported by a distributed collaborative information system operated at fourteen regional emergency call centres, or PSAPs (Public Safety Answering Points). Each of these PSAPs serves emergency calls from its home region primarily. If the home PSAP is occupied or out of order, the system automatically reroutes emergency calls to another PSAP, where the call is processed in the same way as it would be processed by the PSAP of the home region. Every single operator knows the actual operational status and language skills of all the other operators logged into the system. Thanks to the cooperative functionality based on an instant messaging subsystem, which is transparent to the user, operators can ask for help or offer their free capacity and skills in conference mode to the other operators. As all the descriptive and operational data are shared or replicated between system nodes in the background, any operator can receive an emergency call from any region, regardless of his/her position with respect to the location of the incident.

Experience from the operation of the emergency call-taking system suggests that in a normal situation calls are smoothly processed by operators in the region where the incident originated, without any special demands on the system settings. In highly critical situations, when many incidents happen in a short period of time (e.g. during storms or floods), or many people are announcing the same incident (e.g. a plane crash, gas explosion, or large fire), the system could be locally overloaded. In this case some intelligent reconfiguring scheme would help to balance the system load with respect to the resources available. By means of routing schemes via which calls are distributed, the quality of the service affecting the network throughput and the prioritisation of critical services, as well as postponing the replication of less important data to lower network traffic in overloaded regions, can be managed dynamically, with the goal of improving system responses in critical situations.

In order to be able to apply proper and timely management actions, the system must first recognise the critical or anomaly situation. There is a central database, containing records of emergencies, or incidents, from the whole territory of the Czech Republic. This database can be used for the monitoring and analysis of the current situation with respect to the system and operators' performance, as well as for learning from historical incidents and mining spatial and temporal anomalies.

This paper presents an enhancement of the previous work described in [14,15]. Building on their experience with the detection of anomalies, the authors focused on the cooperative characteristics of the emergency call-taking system in order to show a non-traditional approach to the monitoring and analysis of the system's performance. The authors describe the detection of an anomaly situation from emergency data by unsupervised machine learning, namely the application of the Kohonen Self - Organising Map (SOM) algorithm. Measures taken after the critical situation is detected are supposed to be applied outside this analytic process and are not discussed here.

This article is organized as follows. In Section 2, we compare our approach with other related works. Section 3 describes the principles of the SOM algorithm used for clustering and Section 4 deals with features of the software tools. Section 5 illustrates the basic experimental results with data transformations applied to the training set to form a suitable search space and achieve satisfactory outputs. This experiment is focused on the nature of the incident in order to search for similarities in the type and place of the incident, bearing in mind the proof of concept in general. Section 6 depicts a set of experiments with the classical and Growing Grid SOM applied to the incident report attributes, describing rather the architecture and technology in the system background. In this way we can reveal global or temporal weaknesses in the emergency call-taking system as a whole to improve its performance. In Section 8 we discuss the results followed by conclusions in Section 9.

2 Related Works

Anomaly detection methods for various professional domains have been designed and well described. SOM has been used for network intrusion detection [18,22], fraud detection [3], mechanical fault detection [26], and anomaly detection in the generic time series data [9] etc. A common approach in using SOM for anomaly detection is to build a classifier distinguishing between anomalous and non-anomalous classes of data. The non-anomalous data are used to create a model of the correct situation. After that, a single input vector is presented to the trained SOM and the winner neuron closest to the input vector is found. If the distance from the winner's representative is below a certain limit, the input is classified as belonging to the winner's cluster of non- anomalous data. Otherwise, the input is considered anomalous. These classification methods assume that either a non-anomalous subspace is known before the learning starts [22] or the resulting clusters are compared with an expert classification [3]. Generating artificial anomalous cases with a Negative Selection Algorithm inspired by the human immune system [2,5] in combination with a back-propagation neural network [9] falls into this category. Thus the detection of anomalies in this traditional view is based on supervised learning.

Our approach is based on two facts. First, distinguishing between anomalous and non- anomalous cases within emergency calls is disputable. Second, if an emergency situation exceeded the "normal" scale, it would probably be reported by a set of single emergency cases having certain attributes in common. We are therefore interested in revealing certain patterns in the emergency call data, rather than deciding whether a fresh new case is somehow strange compared to the previous experience. After the patterns are recognised automatically, the resulting map is always presented to a person for them to analyse the situation.

To enhance this concept, providing that the algorithm is being run periodically, the new map is shown to the supervisor or to the network management module if the composition of the patterns found in the recent run is different from the composition formed in the previous run. SOM quality measures include an index of the map's disorder [20] or the goodness of the map based on the distances between the winner and the second-best match node [13]. While the SOM algorithm has been widely used for classification tasks, using it for clustering data analysis has been relatively outside the focus of the research community [25]. This paper describes another application of SOM in cluster analysis. Some authors analysed [21] the performance of a business-oriented call centre with respect to its operator's activity, efficiency, and ability to meet client and business needs. In this paper, we focus on the performance of the Emergency Call Centre (ECC) mainly from the technological point of view, although we also attempted to address the operator's activity analysis.

3 Clustering and Self-organizing Maps

Cluster analysis groups objects (data records) into classes (clusters) in such a way that objects in the same cluster are very similar, while objects in different classes are quite distinct. One of the possible clustering methods is using competitive learning [8]. Given the training set of objects, competitive learning finds an artificial object (representative) most similar to the objects of a certain cluster.

A commonly used application of competitive learning is the Kohonen Self-Organising Map [16], or SOM, described by Teuvo Kohonen in 1982. SOM is inspired by the cortex of the human brain, where information is represented in structures of 2D or 3D grids. Formally, SOM is a type of artificial neural network [11] with two fully interconnected layers of neurons, the input layer and the output or Kohonen layer.

The first step of Kohonen learning is competition. Given the training vector on the network's input and weight vector for each neuron of the Kohonen layer, the neuron with the minimal (usually Euclidean) distance between the weight and input vectors is excited or selected as the winner of the competition [11].

The second step is adaptation. The neurons of the Kohonen layer are organised in a one-, two-, or three- dimensional lattice, reflecting its biological inspiration. A topological neighbour-affecting function is defined on the Kohonen layer, assigning a degree of participation in the learning process to the neurons neighbouring the winning neuron. In every learning step the weight vectors of the winning neuron and its neighbours are adjusted to move closer to the input training vector.

In the batch version of the SOM algorithm, equivalent to Lloyd's vector quantisation [19], the winning neuron weights are not adapted immediately after the competition step. When all the training set is consumed, the weight vector of the output neuron $N_i, i = 1, \ldots, n$, where n is the number of neurons in the network, is replaced by the weighted mean value of the training cases assigned to the clusters represented by the neuron N_i and its neighbours, using the neighbour-affecting function as the weight function for the mean calculation.

The trained network finally sets its weights in such a way that the topologically near neurons represent similar training cases, while distant ones reflect different cases. This is analogous with the cortex of the human brain, where similar knowledge is represented by adjacent parts of the cortex. The topology of a trained SOM forms an inherently useful base for clustering.

SOM realises the transformation of the relations of the objects from the m-dimensional input space in a two-dimensional map of nodes (neurons) of the resulting Kohonen network. The complexity of the input space is reduced significantly and, in conjunction with colouring the nodes of the resulting network, data clusters can be effectively visualised.

4 Characteristics of the Software Tools

The authors developed a SOM learning module and a corresponding SOM explorer. The tools were also used in the framework of another project [1]. In the course of this work the SOM explorer has been enhanced by the capability of revealing nodes according to the values of the selected attribute and a Growing Grid has been implemented in the SOM learning algorithm. The learning module exploits a classical SOM learning algorithm, with the neighbourhood radius initially set to 50% of the largest SOM dimension and the exponential radius shrinking.

Some of the commercially available products have the capability to build clusters automatically from the trained SOM, using, for example, the agglomerative clustering over the SOM-Ward distance [24]. Our learning module, however, generates a SOM density map, which shows for every node the mean distance to its neighbouring nodes. The lines formed by nodes having relatively distant neighbours can be taken as cluster borders.

The Growing Grid algorithm [7], compared to the classical fixed-grid SOM, allows the input data to influence the size and shape of the resulting SOM grid. When the entropy of the input data set is low, the growing SOM needs fewer nodes to approximate the input space, the algorithm is much faster, and the resulting SOM is more consistent in terms of having fewer empty nodes. Another advantage of the growing SOM lies in its capability to partially adapt its shape. When the input data have dominant variance in a certain direction, one of the dimensions of the resulting maps extends along this direction. This feature can be amplified by setting the weights of the initial SOM nodes to values evenly distributed between the maximal and minimal values of the attributes with the k-largest variances (for k-dimensional SOM), identifying them after the k-largest eigenvalues of the input data set covariance matrix [12]. Even if this adaptability is not as powerful as in the growing neural gas [6], for example, we consider it useful with respect to emergency data containing spatial information. In our experiments incidents belonging to adjacent geographic areas proved to occupy nodes in neighbouring clusters, thus contributing to the logical composition of the map.

The Growing Grid algorithm starts with an initial size of 2×2 nodes. After every learning phase, the node N with the highest number of input records assigned [7] or with the largest quantisation error [4] is found. Within the closest neighbours of the node N the node D is selected in such a way that the distance $d(N, D)$ is maximal with respect to the distance metric used in the algorithm. In our experiments the commonly used Euclidean distance [8] was exploited. Between nodes N and D, depending on the mutual position of the nodes, either a row or column of new nodes is added. The new nodes are initialised with the mean values of the weight vectors of the adjacent original nodes and the new learning phase starts on the enlarged SOM. The original nodes keep their weights from the previous iteration, and thus the weights of the new nodes fit in between them smoothly. In this way the training vectors are distributed roughly between a few nodes in the first learning phases and the distribution is further refined in the following phases on the maps, which grow in the direction where the probability of refinement is high, and the relationship between the records distributed in the previous iterations is preserved.

The iterative growth-and-learning process can be finished either after a fixed number of iterations or a predefined number of nodes is reached, or when a condition characterising the precision of the map learning is met. We calculated the vector $\bar{x}$ of the training set's attributes means and a training set variance δ as the sum of the distances of the training vectors x_k from the vector $\bar{x}$:

$$\delta = \sum_{k=1}^{n} ||x_k - \bar{x}|| \tag{1}$$

where n is the number of input vectors in the training set. We evaluated the absolute quantisation error q of the whole map after every learning phase:

$$\delta = \sum_{i=1}^{m} \frac{1}{c_i} (\sum_{j=1}^{c_i} ||x_j - w_i||) \tag{2}$$

where m is the number of nodes in the current iteration map, c_i is the number of vectors assigned to the node i in the previous training phase, $x_j, j = 1, \ldots, c_i$ are the vectors assigned to the node i in the previous training phase, and w_i is the weight vector of the node i. The process stops when the expression (3) is encountered.

$$q < \frac{\delta}{\varphi} \tag{3}$$

where φ was the input parameter defining how many times lower than the initial training set variance the learning precision had to be to stop the algorithm. In expression (3) we compare the factor of inaccuracy in the trained map defined by the quantisation error q of the expression (2) with the initial variance of data defined by the expression (3). The algorithm stops when the learning inaccuracy factor is satisfactorily lower than a certain fraction of the initial variance of data, defined by the parameter φ. For φ we used values between 10 and 100, the lower ones resulting in a shorter run time but a final map of poorer quality, consisting of fewer nodes and vice versa with the higher φ values. In comparison with the fixed number of iterations [4] or the maximum number of nodes [7] the stopping criterion (3) used in our algorithm adapts to the original data distribution, and allows better tuning of the algorithm with respect to the run time and accuracy of the final map.

To be able to experiment with the algorithm speed, we implemented a heuristic consisting of the count of variable epochs per learning phase. We increased the epoch count in each new learning phase, according to:

$$\epsilon_t = {\epsilon_{max}}^{1 - \frac{q_{t-1}}{q_1}} \tag{4}$$

where ϵ_t is the epoch count in a learning phase t, ϵ_{max} is the input parameter defining the maximal epoch count, q_{t-1} is the map quantisation error after the learning phase $t-1$, and q_1 is the quantisation error after the first learning phase. In this way the initial iterations, when the data were roughly separated, had a lower epoch count, while the final iterations, when the SOM was to be fine-tuned, reached the maximal epoch count.

Expression (4) is a natural mathematical formulation of the intention of having a lower epoch count at the beginning of the learning of the map and reaching the maximal epoch count towards the end of the learning process to fine-tune the map.

As the weight vectors of the nodes of the map are quite distant from the training vectors and the number of nodes is small at the beginning of the training, fewer learning epochs are needed to roughly organize the input space before the map grows and the next iteration starts. At the end of the training process, when the quality of the map learning becomes high (quantisation error gets low values) and the map growing is no more needed, the epoch count should reach its maximal value to find the best data distribution with respect to the actual topology of the map.

Although the epoch count, thanks to the stochastic nature of the SOM learning and corresponding quantisation error calculation, does not change monotonically, the function converges in both increasing and decreasing modes, speeding up the learning.

We used the absolute quantisation error in evaluating the learning quality as it implied better results with the variable epoch count heuristic, compared to the mean quantisation error proposed in the literature [4].

5 General Test of SOM Emergency Data

We aim to show that the SOM can be successfully used to reveal anomalies in the emergency incidents database in general. Incident records from the period February 1^{st} March 31^{st} 2008 were used for the experiments. On March 1^{st} 2008 the territory of the Czech Republic was affected by Hurricane Emma.

The input data set consisted of about 25,000 records. The SOM-based procedure was supposed to find records related to Hurricane Emma (Emma records), depicting the Emma cluster formed by nodes representing the Emma records characterised by:

– specific types in the incident classification (storm, danger status removal, obstacle removal);
– a higher frequency of incidents, namely of the above-stated types, in the time period in question;
– a higher frequency of incidents in certain regions (districts).

The training vectors presented to the SOM algorithm consisted of the attributes “time of the incident beginning”, “incident classification”, “region of origin of the incident”, and artificial attributes derived from these primary ones.

As the incident classification and region are attributes of a categorical type, they are not suitable for the conventional SOM learning algorithm [14,15].

Therefore we were looking for a transformation of the categorical attributes into the real numbers domain, which would have preserved the distribution characteristics of the original attribute values. We introduced frequency characteristics in a time quantum, the hour of the incident origin, as the basis for transformation.

The categorical attributes Classification and District were used as the source for artificial numeric attributes produced by a function of relevance, combining the classification or district frequency within the time quantum of origin of the current incident and the occurrence of the same classification or district in the remaining quanta (adopted from the text classification TF-IDF weighing model [23]).

$$IMP\_CL = FREQ\_CL\_H \ \log \frac{N_h}{CL\_HRS} \tag{5}$$

$$IMP\_DS = FREQ\_DS\_H \ \log \frac{N_h}{DS\_HRS} \tag{6}$$

where N_h means the number of hours in the period analysed, $FREQ\_CL\_H$ is the frequency of the current incident classification within the subset of incidents related to the hour of origin of the current incident, $FREQ\_DS\_H$ denotes the frequency of the current incident district within the subset of incidents related to the hour of origin of the current incident, and CL_HRS or DS_HRS denote the number of hours in which the classification or district emerged.

Another attribute that was transformed was the district relevance.

$$IMP\_CL\_DS = FREQ\_CL\_DS \ \log \frac{N_d}{CL\_DS} \tag{7}$$

where N_d means the number of districts in the Czech Republic, $FREQ_C L_D S$ is the frequency of the current incident classification in the current incident district during the whole period analysed, and CL_DS denotes the number of districts where the current incident classification occurred.

The training set with transformed artificial attributes was presented to a static SOM network of 40×40 nodes. Figure 1 shows the final output after 100 training cycles.

Using the node analysis functionality of the SOM explorer, we were able to derive some knowledge from the trained map, as shown in Figure 2.

The results of this basic experiment illustrate that the SOM technique succeeds with anomaly detection in the emergency call-taking system database. In the following section we will further elaborate on this conclusion with respect to the monitoring of the performance of the emergency call-taking system.

6 Performance Monitoring of the Emergency Call-Taking System

The performance monitoring relies on attributes describing the architecture and technology in the system background, rather than the nature of the incident as seen in the previous experiment. A brief overview of the architecture of the emergency call-taking system is provided in Figure 3.

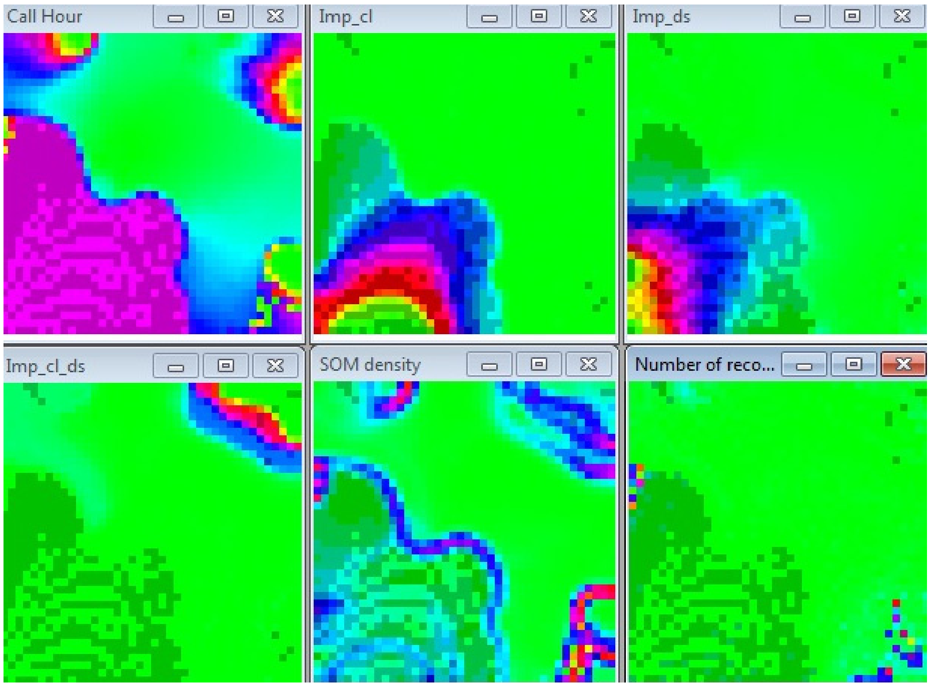
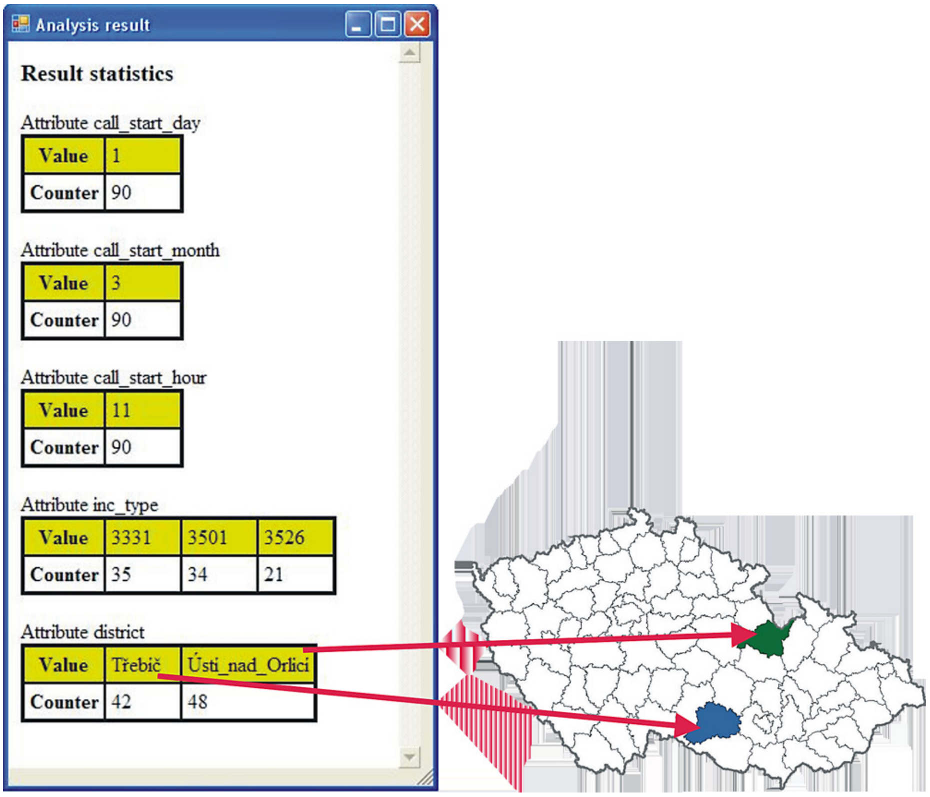


Fig. 1. Shadowed nodes contain Emma records; the Emma cluster is well bordered by the wavy line in the bottom left-hand corner of the SOM density map (bottom middle). The rainbow-like stripes within the Emma cluster, visible in the IMP_DS map (top right), point to the movement of Emma across the territory of particular districts. The anomalies not related to Emma are concentrated in the upper right-hand corner of the IMP_CL_DS map (bottom left), collecting specific incidents assisted by the Prague Municipal Police, pointing to a higher rate of car thefts in Prague and system tests being performed regularly at the particular PSAP.



Result statistics

Attribute call_start_day

Value	1
Counter	90

Attribute call_start_month

Value	3
Counter	90

Attribute call_start_hour

Value	11
Counter	90

Attribute inc_type

Value	3331	3501	3526
Counter	35	34	21

Attribute district

Value	Třebič	Ústi_nad_Orlici
Counter	42	48

Fig. 2. The output of the analysis of the nodes in one of the rainbow-like stripes from Figure 1 states that on March 1st, from 11 to 12 a.m., 90 Emma-related incidents of types 3331 (windstorm), 3501 (removing dangerous objects) and 3526 (removing obstacles) were reported from two districts in the central part of the Czech Republic

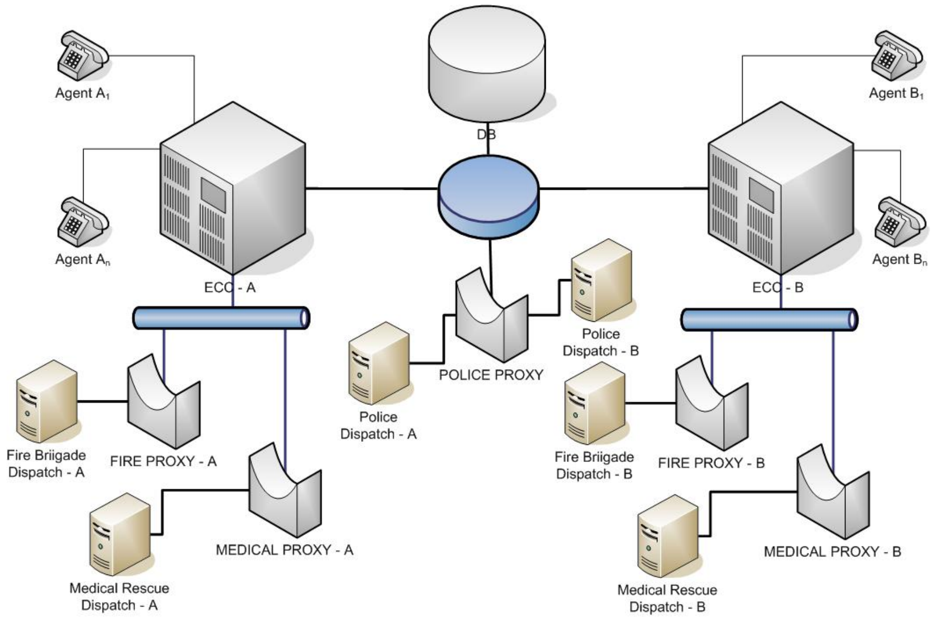


Fig. 3. Example of the architecture of the emergency call-taking system reduced to two communicating Emergency Call Centres (ECC) with related emergency response agencies' proxy services and the central database service

In a normal situation emergency calls are processed by operators in the region, where the related incident originated, e.g. in Region A. Once the call is processed by an operator-agent of the Emergency Call Centre (ECC), the incident report is stored in the database, which is a remote service to most of the centres, accessible through a Wide Area Network (WAN). A delay in the operation of the database may suggest a network overload or a DB service capacity problem. In parallel with the incident report being inserted into the DB, it is passed to the Emergency Response Agencies (ERA) via the respective proxy service. Fire and Medical Rescue services are accessible in the local network with respect to the receiving ECC, while the police provide one central proxy accessible via WAN for all but one ECC. The proxy services, after passing the incident report to the ERA, expect a confirmation message. Delays in receiving this confirmation may give evidence of the ERA systems being overloaded. In the telephone subsystem we can measure the ringing time, i.e. the time interval between the moment when the call arrives at the agent and the moment when the operator picked up their phone. As picking up the phone is automatic (auto-answer mode is preset in the whole system), stretching of the ringing interval may suggest congestion in the telephone system. Moreover, two values characterising the operator's working style are measured, the length of the call and the time interval between the start of the call and the insertion of the incident report into the DB.

In critical situations, when many incidents happen in a short period of time, or many people are announcing the same incident and the operators of ECC A are busy, calls are routed to and processed at the ECC of another region (say B). In the course of processing the calls the DB and telephone call measures have the same meaning as in the previous case. But after the call is processed by ECC B, the reports to the ERA are communicated via the WAN to the ECC of the original region, A. The further incident report flow is identical to the previous case. The time it takes to transfer the incident report between ECC B and the proxy services of ECC A is measured and evaluated.

For the performance monitoring experiments we used data from a special testing of the emergency call- taking system under peak load. In an hour the system in its full capacity (14 ECC, 70 operators) received 3400 emergency calls and produced around 10,000 incident reports. As there was only one incident type reported, the "system test", it made no sense to take the incident classification into account. Table 1 shows the attributes used in our experiments.

6.1 Comparison of SOM Alternatives

At the beginning we show the results of the traditional fixed SOM algorithm compared to its Growing Grid modification with the heuristic of a variable epoch count. We used a fixed epoch count, as well as increasing the epoch count according to the formula (4). The epoch count, node count, and the grid size produced by the growing SOM algorithm were used as input parameters for the fixed SOM which was to be compared. The results are shown in Table 2.

Empirical results illustrate that the Growing Grid SOM is faster than the respective fixed SOM algorithm, achieving a slightly worse learning quality.

Table 1. Description of attributes used in the performance monitoring. The categorical attributes were replaced by natural numbers to suit the SOM algorithm.

Name	Type	Description
ID_ECC_REC	Categorical	ID of the receiving ECC (natural number)
X_ECC_REC, *Y_ECC_REC*	Numeric	Geo coordinates of centroid of the region which the reported incident belongs to
OFFHOOK_DLY	Numeric	Off-hook interval, time delay between the ringing and call start moments
CALL_LENGTH	Numeric	Call length
ID_ERA	Categorical	ID of the Emergency Response Agency (ERA) which is to be alerted (natural number)
ID_WS_REC	Categorical	ID of the agent responding to the call (natural number)
ID_SRV_CENTRE	Categorical	ID of the proxy service responsible for communication with the particular ERA (natural number)
MINING_TIME	Numeric	Time from the start of the call to the moment of saving the incident record
INSERT_DLY	Numeric	Database service response time
DELAY_ECC	Numeric	Time delay of the transfer of the incident in the ECC technology
DELAY_CAD	Numeric	ERA technology response time
ID_ECC_RESP	Categorical	ID of the target ECC which the incident is transferred to (natural number)

Table 2. Comparison of the fixed and growing SOM alternatives. The "Epochs mode" denotes an increasing (Inc) or constant (Const) epoch count, and the "Candidate selection" denotes the way the central node for growing is selected, using either the quantisation error (Qerr) or the number of mapped input records (Match) of the node. The Euclidean distance was used in all the cases presented here. The tests were run on a PC with a 3-GHz Pentium 4 CPU and 2 GB of RAM.

ID	SOM type	Epochs mode	φ	ϵ_{max}	Candidate selection	Epoch count	Node count	Grid size	Run time	Final Map MQE
1	Growing	Inc	10	50	Qerr	571	595	17×35	0:08:51	0.0482
2	Fixed	-	-	-	-	571	595	17×35	0:13:28	0.0333
3	Growing	Const	10	50	Qerr	1850	384	16×24	0:09:33	0.0548
4	Fixed	-	-	-	-	1850	384	16×24	0:31:30	0.0446
5	Growing	Inc	10	50	Match	320	468	26×18	0:05:02	0.0522
6	Fixed	-	-	-	-	320	468	26×18	0:08:20	0.0397
7	Growing	Const	10	50	Match	1750	352	16×22	0:08:33	0.057
8	Fixed	-	-	-	-	1750	352	16×22	0:23:03	0.0482
9	Growing	Inc	25	20	Qerr	797	1517	37×41	0:31:21	0.0256
10	Fixed	-	-	-	-	797	1517	37×41	1:13:46	0.0145
11	Growing	Const	25	20	Qerr	1640	1764	36×49	0:50:04	0.0254
12	Fixed	-	-	-	-	1640	1764	36×49	2:21:29	0.0112

The differences between the mean quantisation errors of the respective growing and fixed SOM represent a max. 3% of the initial variance of the training vectors. The selection of the central node for map growth according to the maximal number of mapped input records produces smaller maps, resulting again in faster and slightly less precise learning. The higher the φ parameter in (3), the larger and more precise, but also more time-consuming, the maps produced are. Figure 4 displays comparable maps produced by Tests 9 and 10 from Table 2.

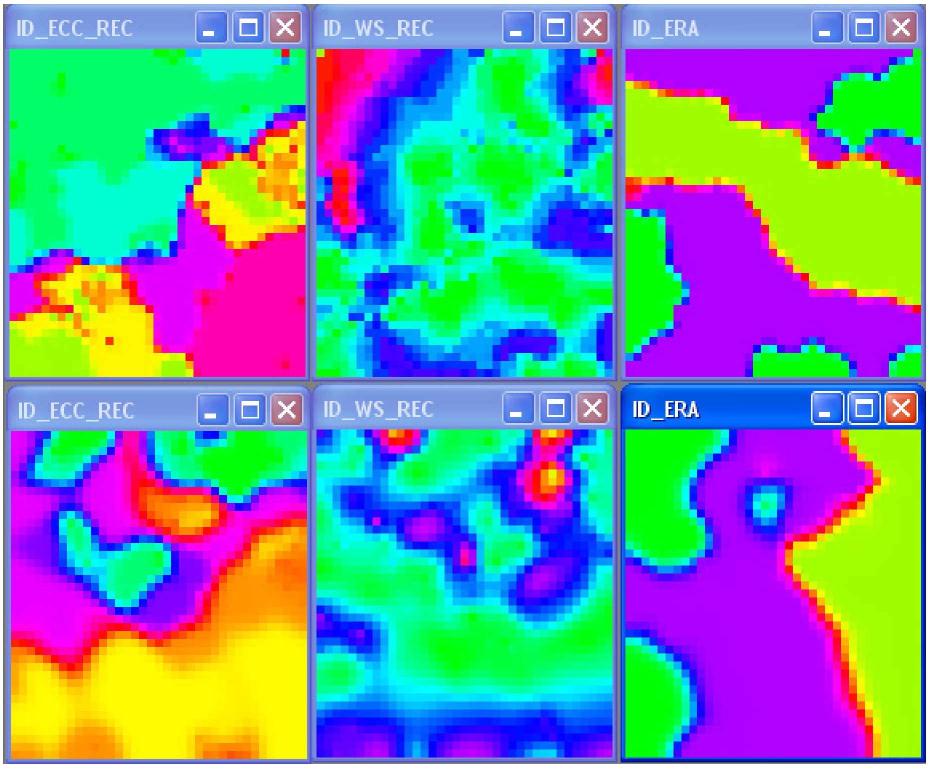


Fig. 4. Maps produced by the fixed SOM (Test 10 in Table 2) in the upper row, and growing SOM (Test 9 of Table 2) in the lower row. The growing SOM maps appear to be more coherent: emergency centres separated by the *ID_ECC_REC* attribute are close to each other according to the geographical adjacency of their regions; workplaces inside emergency centres are pointed out in the *ID_WS_REC* growing SOM map; the *ID_ERA* growing SOM map clearly separates incidents attended by the medical rescue service in the western regions from those in the eastern regions, supporting the geographical dependencies in the maps.

The visual quality of the Growing Grid SOM output in this example is even better that that of the more accurate fixed SOM, which takes double the time. The following experiments exploit the growing SOM algorithm.

6.2 Technology Performance Analysis

In the test of the emergency call taking system the aim was to show the system with a heavy load in unrealistic conditions in order to reveal its hidden flaws. Figure 5 (below) shows the anomalies shown in the system technology.

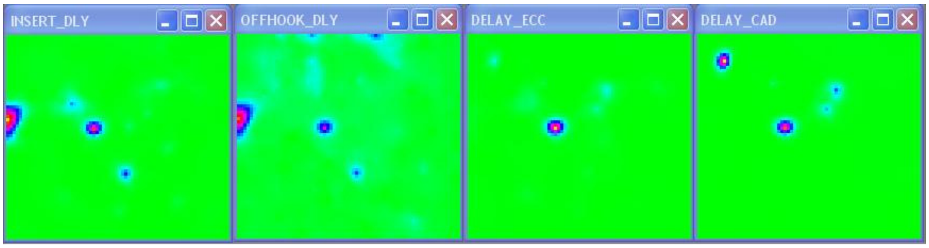


Fig. 5. Maps showing delays in the telephone subsystem (*OFFHOOK_DLY*), database operations (*INSERT_DLY*), incident transfer in the emergency centres' technology (*DELAY_ECC*) and in the responding agencies' technology (*DELAY_CAD*)

Interesting values reported by the analytical module for the maps in Figure 5 are shown in Table 3.

Table 3. An example of the records related to the highlighted nodes from Figure 5. Time intervals characterising delays are in seconds.

ID	ID_ECC_REC	RINGING_TIME	INSERT_DLY	OFFHOOK_DLY	DELAY_ECC	DELAY_CAD	ID_ECC_RESP
1	30	20:03	3	1	1	1	40
2	110	20:05	199	15	8	1	40
3	110	20:05	199	15	8	4	40
4	80	20:11	26	1	128	219	20
5	20	20:12	146	5	500	500	20
6	20	20:12	157	2	500	500	20
7	20	20:13	214	19	500	500	20
8	90	20:13	1	1	228	500	20

Row 1 stands for an example of the normal situation, when emergency centre 30' received an incident for centre 40'. The database response is 3 seconds, which is within acceptable limits. The delay in the telephone subsystem is 1 second, as is the incident transfer time in the emergency centres' network and in the emergency response agencies' technology.

Rows 2 to 8 represent records mined out from the red and blue spots of the four maps in Figure 5. Rows 2 and 3 point to problems with the database and telephone subsystem for centre 110'. The communication with the target centre 40' and its related emergency agencies is alive, even if the slightly higher values may suggest incoming problems. On the contrary, row 4 shows that centre 80', when receiving for centre 20', encounters long database responses, as well as significant delays in communicating the incident report to the target centre. A minute later, records 5-7 show deadlock at centre 20'. With unacceptable database responses the centre cannot even communicate with the other centres and with its own emergency agencies (here the value 500 stands for unknown information). The last row, row 8, shows that another centre, centre 90', handles incidents properly, but cannot propagate incident reports to target centre 20' and its emergency response agencies, definitely because of the problems of centre 20' reported above.

6.3 Obseravtion of Operators' Behaviour

This experiment concentrated on a few attributes which we hoped to use in exploring differences in operators' behaviour under a heavy load. The respective maps are shown in Figure 6.

If there was an anomaly in the time the operator needed to collect basic information from the person calling, it correlated with the length of the call and was caused by the technical problems discussed above. See the correspondence of the red points in the $CALL\_LENGTH$ and $MINING\_TIME$ maps. The ones closest to the centre of the map and its left-hand margin have their counterparts in the $OFFHOOK\_DLY$ and $INSERT\_DLY$ maps from the previous paragraph. These anomalous calls took from 70 to 240 seconds.

The operators of the emergency centres, which did not have technical problems took the emergency calls one after another without any pause and tended to a shorter speaking time, typically 15 to 40 seconds. This is clearly shown by the green areas in the top third of the $CALL\_LENGTH$ and $MINING\_TIME$ maps, while the corresponding continuous area at the top of the ID_WS_REC map indicates that these operators were occupying workplaces in Prague.

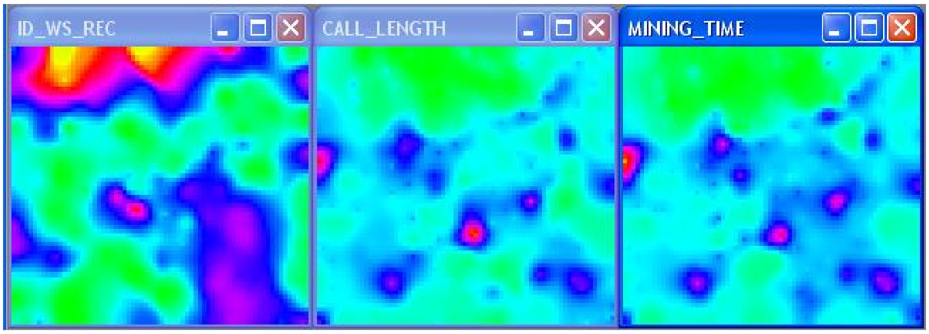


Fig. 6. Maps characterising operators and their activity: ID_WS_REC defines the operator's workplace at the emergency centre, $CALL\_LENGTH$ shows the time the operator is speaking to the caller, and $MINING\_TIME$ measures the time interval from the start of the call to the moment the incident record was saved or the time which the operator needs to get the basic information from the caller.

7 System Performance and Its Improvement

As mentioned earlier, the system response represents the crucial element of the Emergency Call-Taking System and the same holds for all utilized software components. The Table 2 (see the column named Run time) shows that the computation of the SOM algorithm itself can take a lot of time with respect to the individual settings. This times can differ with respect to hardware platform as well. An alternative hardware platform and software modifications, which bring significant improvements, are described in the following text.

7.1 nVIDIA CUDA

Modern graphics hardware play an important role in the area of parallel computing. Graphics cards have been used to accelerate gaming and 3D graphics applications, and now, they are also used to accelerate computations with relatively distant topics, e.g. remote sensing, environmental monitoring, business forecasting, medical applications or physical simulations etc. Architecture of GPUs (Graphics Processing Unit) is suitable for vector and matrix algebra operations. That leads to the wide usage of GPUs in the area of information retrieval, data mining, image processing, data compression, etc. [27]. Nowadays, one does not need to be an expert on graphics hardware because of existence of various APIs (Application Programming Interface), which help programmers to implement their software faster. Nevertheless, it will be always necessary to keep basic rules of GPU programming to write required code more effective.

Andrecut [29] described computing based on CUDA on two variants of Principal Component Analysis (PCA). The usage of parallel computing on GPU improved efficiency of the algorithm more than 12 times in comparison with CPU. Preis et al. [30] applied GPU on methods of fluctuation analysis, which includes determination of scaling behavior of a particular stochastic process and equilibrium autocorrelation function in financial markets. The speed up was more than 80 times than the previous version running on CPU. Patnaik et al. [31] used GPU in the area of temporal data mining in neuroscience. They analyzed spike train data with the aid of a novel frequent episode discovery algorithm. Achievement of more than 430 speed up is described in the mentioned paper.

7.2 GPU Parallelism

Several different implementations of parallel SOM have already been presented in [33], [34], [35] and [36] and there are also studies on how the computer architecture could be modified in order to support highly parallel calculations especially for SOM training [37]. These approaches focus mainly on time efficiency issues and how the mathematical operations can be distributed on different machines to speed up the self-organization process. Different approach was presented by I. Valova [38], [39], where the amount of parallelism is not determined by the number of available hardware resources but rather by the size of the input pattern.

More convenient approach for a completely parallel SOM has been presented by Weigang [40].

All cited approaches have to solve some kind of vector or matrix operations which are more or less time consuming. The power of GPU is shown on a basic concept of self organizing maps, although proposed improvements can be applied on any kind of SOM.

7.3 Basic Notation

The following notation is used to better describe the process of parallel computation of SOM. All experiments and examples in this subsection follow the specification of SOM presented below(see also the Figure ??):

- The SOM is initialized as a network of fixed topology. The variables $dimX$ and $dimY$ are dimensions of such 2-dimensional topology.
- V^m represents an m-dimensional input vector.
- W^m represents an m-dimensional weight vector.
- The number of neurons is defined as $N = dimX * dimY$ and every neuron $n \in< 0, N-1 >$ has its weight vector W_n^m
- The neighborhood radius r is initialized to the value $min(dimX, dimY)/2$ and will be systematically reduced to a unit distance.
- All weights vectors are updated after particular input vector is processed.
- The number of epochs e is know at the beginning.

7.4 Kernel Functions

Well designed CUDA kernel functions enable us to get better performance during runtime. Kernels functions, when called, are executed N times in parallel by N different CUDA threads, as opposed to only once like regular functions. Thus it is necessary to design a kernel function so that every thread access different memory block. We refer to [45] for more information on CUDA memory management.

A kernel function is defined using the *__global__* declaration specifier and the number of CUDA threads for each call is specified using a new $<<< \ldots >>>$ syntax. The brackets $<<<$ and $>>>$ hold grid size, block size, amount of shared memory and stream index. The setting of these parameters affect the distribution of threads over GPU. We refer to [27] for more information.

The principle of PR (Parallel Reduction) brings a significant time savings. A modified version of PR was used in a phase of computation of vector differences and Euclidean distances. The Figure 7 shows an illustrative example of settings of grid, blocks and threads which were applied on SOM, where the number of neurons N = 9 and vector dimension m = 8. The GPU grid contains 3 blocks. Every block has 32 threads, therefore it can cover at most four weights vectors. The setting of kernel function itself play an important role in CUDA. It depends on GPU hardware parameters as well. The parallel reduction is shown in the middle part of the Figure 7 and has following steps:

A: The differences between all elements of a given input vector and an appropriate weight vector are computed and stored in the shared memory on GPU. The elements can be powered by 2 in case of Euclidean distance.
B: A half of previously used threads computes partial sums of elements of difference vector.
C: Last thread computes square root of the final sum and stores the result output into distance vector, which is stored in global GPU memory.

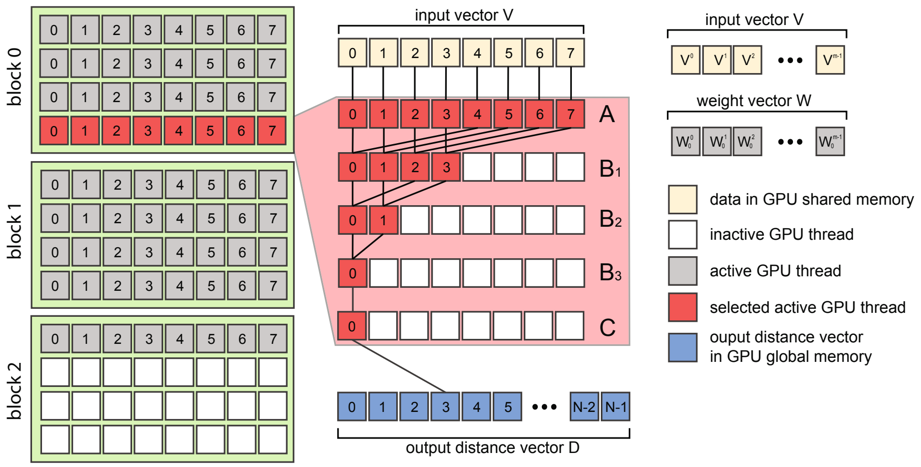


Fig. 7. Processing of a single input vector by parallel GPU threads

when all the blocks finish, the process of parallel reduction is used to find a minimum value in the distance vector. An index of the minimal element represents BMU (Best Matching Unit).

7.5 Parallel SOM Update

Some fix data is precomputed in our application. Such data is stored in the global memory and then partially copied into shared memory by all thread blocks to reduce computational time [45] during SOM update. First, it is a distance matrix which holds the distances among neurons in a SOM. In the figure 8 on the left, there is an example of the SOM with rectangular topology of dimension 7×7. The maximum distance in this topology is $maxDist = (dimX - 1) + (dimY - 1) = (7-1)+(7-1) = 12$. Just for illustration, the black node in the middle represents BMU and all nodes with the same gray scale have the same distance to the BMU with respect to the rectangular topology.

A vector of learning factors F makes the second fix data. In the illustrative figure 8 on the right, there are curves of learning functions for all epochs ($e = 4$). The vector F has a dimension equal to $(maxDist + 1) * e = (12 + 1) * 4 = 48$

in our example and represents a set of values $f_i(d)$, where $f_i(d)$ is the evaluated learning function (learning factors) for a given epoch i, and a distance d, where $d \in < 0, maxDist >$ and $i \in < 0, e >$.

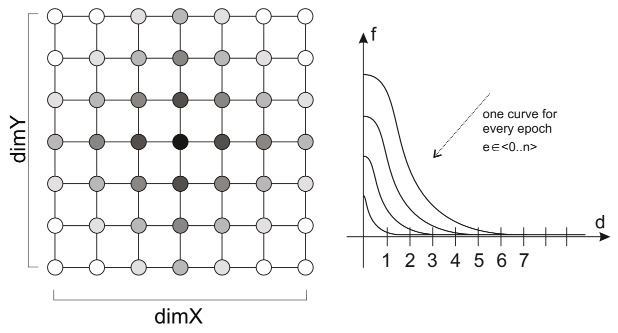


Fig. 8. Neuron distances in the SOM for a given BMU in the center of the SOM (on the left) and curves of learning functions for all epochs

The process of updating of SOM runs in a few steps:

1. The GPU kernel function, that is responsible for updating, is set so that one neuron is processed by m threads, where m is a dimension of weight vector.
2. Fixed distance vector and learning factors for a given epoch are copied into GPU shared memory to ensure better performance.
3. Every thread updates a part of its weight vector with respect to the learning factor and the distance to BMU.

7.6 Kernel Example

Although there is no place to show the whole code of GPU implementation of SOM because it contains hundreds of lines, following part of the code illustrates the philosophy of parallel programming with CUDA. The kernel function *update-CompleteSOM* is called from the main program when BMU is already known. We refer to [27] for more information on GPU programming and to [??] for detailed description of implementation of SOM on GPU.

7.7 Experiments on GPU

All experiments include comparison between GPU and CPU implementation of SOM. The Table 4 includes detailed specifications named GPU, CPU-A and CPU-B witch respect different hardware configuration used for experiments.

```
__global__ void updateCompleteSOM(
   float* weightVectors,
   const unsigned int wSize,
   const unsigned int* distanceMatrix,
   const unsigned int* BMUindex,
   const float *distances,
   const float *learingFactors)
{
   unsigned int tid = threadIdx.x;
   __shared__ unsigned int sBMUindex;

   if (tid == 0) sBMUindex = *bmuIndex;
   __syncthreads();

   unsigned int wid = blockDim.x * blockIdx.x + tid;
   if (wid >= wSize) return;
   unsigned int nid = (wid / vectorDim) * dimX * dimY + sBMUindex;
   weightVectors[wid] += learingFactors[distanceMatrix[nid]]
                      * distances[wid];
}
```

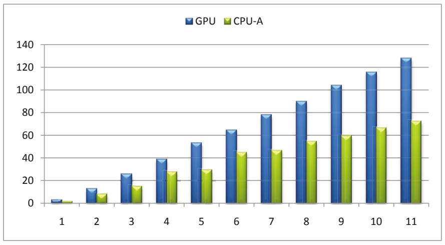
Table 4. Hardware specification

	GPU	CPU-A / CPU-B
CPU	Intel Core 2 Duo 3,0Ghz	4 x AMD Opteron 1,8 GHz
RAM	4 GB	32 GB
GPU	GeForce 280 GTX, 1 GB	-
Threads	depends of GPU	2 (CPU-A), 8(CPU-B)

A SOM with a fix network size is computed in the first experiment. Dimensions of input vectors are fix as well and they are set to 4 in this case. This experiment reveals hidden disadvantages of GPU utilization, as can be seen in the Figure 9 or in the corresponding table. The computation time increases near linear, however, GPU time is approximately 2 times greater in comparison with CPU-A multi-threaded implementation. This is due to additional time costs associated with transactions between RAM and GPU memory. Next, calling *kernels* (GPU functions) and inner thread indexing take some time. These costs make GPU implementation not suitable for SOM in case of small data.

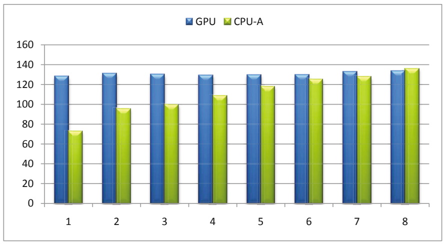
In the Figure 10, there are results of the second experiment which deals with increasing dimension of input and weight vectors. Again, GPU implementation seems to be not effective until dimension 32 because of the same reasons as in the first experiment. However, the trends of graphs predict future computation times and it is clear, that GPU time increases very slowly in comparison with CPU-A.

The third experiment and its results in the Figure 11 show the power of parallelism on neurons (we refer to the section 7.5), whereas the previous two experiments illustrate parallelism on particular vectors (we refer to the section 7.4). The contribution of GPU implementation is perceptible in higher dimension of SOM network.



ID	number of inputs	vector dim.	SOM network		time in seconds	
			dimX	dimY	**GPU**	**CPU-A**
1	1000	4	10	10	3,11	1,64
2	5000	4	10	10	13,11	8,05
3	10000	4	10	10	25,68	14,98
4	15000	4	10	10	38,49	27,80
5	20000	4	10	10	53,23	29,43
6	25000	4	10	10	64,56	44,91
7	30000	4	10	10	78,02	46,41
8	35000	4	10	10	89,95	54,44
9	40000	4	10	10	104,11	59,46
10	45000	4	10	10	115,47	66,38
11	50000	4	10	10	127,90	72,36

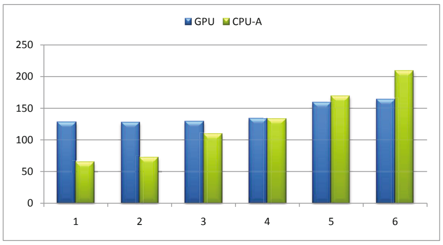
Fig. 9. Variable number of inputs



ID	number of inputs	vector dim.	SOM network		time in seconds	
			dimX	dimY	**GPU**	**CPU-A**
1	50000	4	10	10	127,90	72,36
2	50000	8	10	10	131,06	95,37
3	50000	12	10	10	130,03	99,51
4	50000	16	10	10	129,44	108,37
5	50000	20	10	10	129,79	118,06
6	50000	24	10	10	129,62	125,15
7	50000	28	10	10	133,25	127,58
8	50000	32	10	10	133,53	136,18

Fig. 10. Variable dimension of input vectors

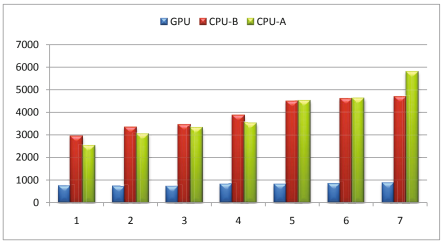
Previous demonstrations confirm, that the power of GPU utilization increases with the size of neural network and with the dimensions of input and weights vectors, respectively. The last two experiments (see the Figure 12 and 13) show the computation times in such cases.



ID	number of inputs	vector dim.	SOM network		time in seconds	
			dimX	dimY	**GPU**	**CPU-A**
1	50000	4	5	5	128,21	64,66
2	50000	4	10	10	127,90	72,36
3	50000	4	15	15	129,11	109,10
4	50000	4	20	20	133,20	133,11
5	50000	4	25	25	158,70	169,38
6	50000	4	30	30	164,21	209,22

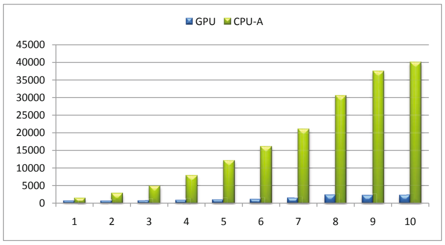
Fig. 11. Variable dimension of SOM network

Although the massive parallelism is not the main goal in case of Monitoring of the Emergency Call-Taking System, it brings additional benefits in the area of data analysis. In case of GPU utilization, the shorter computation time enables us to retrieve more accurate results because more input data can be processed and the size of neural network can be larger in comparison with CPU version.



ID	number of inputs	vector dim.	SOM network		time in seconds		
			dimX	dimY	**GPU**	**CPU-A**	**CPU-B**
1	250000	40	10	10	741,41	2518,67	2943,67
2	250000	50	10	10	716,44	3038,69	3329,97
3	250000	60	10	10	722,97	3325,90	3447,45
4	250000	70	10	10	815,01	3525,53	3871,43
5	250000	80	10	10	819,56	4520,88	4503,06
6	250000	90	10	10	841,52	4617,80	4614,17
7	250000	100	10	10	885,38	5793,50	4689,08

Fig. 12. Variable dimension of input vectors



ID	number of inputs	vector dim.	SOM network		time in seconds	
			dimX	dimY	**GPU**	**CPU-A**
1	250000	4	10	10	692,37	1513,03
2	250000	4	20	20	698,44	2724,38
3	250000	4	30	30	818,93	4846,69
4	250000	4	40	40	905,82	7858,48
5	250000	4	50	50	1042,55	12048,28
6	250000	4	60	60	1176,00	16104,82
7	250000	4	70	70	1459,97	20990,82
8	250000	4	80	80	2268,49	30564,54
9	250000	4	90	90	2199,91	37387,30
10	250000	4	100	100	2307,67	39984,57

Fig. 13. Variable dimension of SOM network

8 Discussions

The emergency call system could be locally overloaded in major crisis situations, natural disasters, or big accidents. As the system is distributed over the whole territory of the state, operators in the regions not affected by a crisis can receive calls from the affected regions. An intelligent system reconfiguration could help in setting up routing policies, optimising network traffic, and balancing the system load with respect to the available system resources.

In the first part of our work, we proposed transformations of categorical attributes, the discrete values of which are not suitable for the SOM algorithm, into the real numbers domain and showed that after this transformation the SOM is generally able to detect anomalies in the emergency data. The transformations are built on the frequencies of incidents in time and place, expressing the relevance of the incident with respect to the time and place of origin of the incident. The learning proceeds on the values of the transformed attributes, which are nevertheless bound to the original records. In this context, the values of the original understandable attributes could be used for explaining the result, as well as for further processing.

In the second part of the paper, we presented the dynamic SOM alternative, the Growing Grid [7], enhanced with a variable learning epoch counter heuristic. It has been shown that the growing SOM with the above- mentioned heuristic produced satisfactory results in a significantly shorter time. We programmed a corresponding software tool and used it in experiments with data collected in the period of a specific emergency system test to simulate performance monitoring of the emergency call centres network and analyse performance issues related to the technology, as well as the human factor the operators of the centres. The proposed SOM modification is well suited to the emergency data domain, which is characterised by relationships of a geographical, organisational, and technological nature. As some of the relationships in the emergency data are inherently hierarchical, we consider the application of the hierarchical SOM [4] in the SW tool developed in the course of this work.

While the first part of the paper was to show the justification of the SOM usage in detecting and visualizing anomalies in the emergency data, in the second part we exploited this partial result in the simulation and performance monitoring of the emergency call centres network under a heavy load. The load was simulated by the system tests which were lasting for one hour. In the tests 80 emergency call operators over the whole country took part together with about 200 of callers who were simulating emergency calls. The data which were describing the behaviour of the system as well as of the operators were collected and processed by the modified growing SOM algorithm. In this way the anomalies in the whole system or in behaviour of all the operators could be seen in the resulting SOM picture.

As one could see and comprehend the status of the whole system at a glance, this holistic visualization of the system performance tends to be much more transparent and intuitive comparing to the traditional monitoring techniques based on trapping and displaying particular events. Moreover, the proposed method inherently includes monitoring of non-technical components of the system, particularly the behaviour of the emergency call operators, which is not covered by traditional trap-based monitoring. From this point of view the method described in this paper could be passed as a contribution to the socio-technical system performance monitoring field.

Traditional anomaly detection methods [9,22,26] use SOM for modelling the anomaly-free space from a set of data approved as correct (bearing signs of supervised learning) and then classify a new case as anomalous if it falls outside the modelled non-anomalous space. As hardly any emergency situation can be considered anomalous, or all of them are anomalous on the other hand, we could not use the two-class classifier approach with supervised learning. Therefore we searched for certain patterns in the data. After the patterns were recognised by unsupervised learning, the composition was always presented to a human for them to analyse the situation.

This concept can be enhanced in such a way that, provided that the algorithm is run periodically, the new composition would be further processed only if the composition in the current run is different from the composition in the previous run. SOM quality measures [20,13] could help in achieving good results here. We expect that the monitoring of the performance of the emergency call-taking system and analysis technique presented in this paper can be effectively combined with a model of resilient incident report transfer between call centres inspired by [10,17]. Such a model could be created and the Growing Grid SOM tool used for its output monitoring and performance analysis, provided that detailed information regarding the underlying technology of the ECC network (the WAN topology, technology and parameters) were available.

9 Conclusion

This paper presented the monitoring and analysis of the performance of the emergency call system, using a novel approach of exploring anomalies in the emergency call system databases. We reused our experience with the detection of anomalies, as described in [14], concentrating on the technology and cooperative characteristics of the emergency call-taking system to show the system's performance issues. Both the technological and personal points of view were included as our method discovered database problems and spotted operators' behaviour in the course of testing a specific system under a heavy load.

We devised a novel method for unsupervised knowledge discovery in the emergency call data, based on the Self-Organising Map (SOM) algorithm and programmed respective software tools: the fixed and Growing Grid SOM learning module and the SOM explorer module. We compared various configurations of SOM learning and showed that the Growing Grid, combined with the variable learning epoch count heuristic, produces the best results in terms of the speed and quality of the output.

The algorithm consumed the training set of records describing the creation and transfer of incident reports, identified a subset of records containing certain common information, and built and visualised clusters over these records. The method devised here proved its ability to discover anomalies hidden in emergency data, visualising them in an effective user-friendly manner and indicating a way towards the further development of the intelligent management of the emergency call information system.

The experiments presented here focused on revealing clusters of incident records which pointed to abnormal situations in time and place and formed a sound basis for the performance monitoring and analysis of the emergency call system.

References

1. Bača, R., Krátký, M., Snášel, V.: Power Outage Data Analysis via SOM Neural Networks. In: Proceedings of ELNET 2008, Ostrava, pp. 87–98 (2008)
2. Bersini, H., Varela, F.J.: Hints for Adaptive Problem Solving Gleaned from Immune Networks. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 343–354. Springer, Heidelberg (1991)
3. Brockett, P.L., Xia, X., Derrig, R.A.: Using Kohonen's Self-organizing Feature Map to Uncover Automobile Bodily Injury Claims Fraud. Journal of Risk and Insurance 65(2), 245–274 (1998)
4. Dittenbach, M., Merkl, D., Rauber, A.: Organizing and Exploring High-Dimensional Data with the Growing Hierarchical Self-organizing Map. In: Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2002), pp. 626–630 (2002)

5. Farmer, J.D., Packard, N.H., Perelson, A.S.: The immune system, adaptation, and machine learning. Physica D 22, 187–204; Reprinted in: Farmer, D., Lapedes, A., Packard, N., Wendroff, B. (eds.): Evolution, Games and Learning, pp. 187–204. North-Holland, Amsterdam (1986)
6. Fritzke, B.: A Growing Neural Gas Network Learns Topologies. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) Advances in Neural Information Processing Systems, vol. 7, pp. 625–632. MIT Press, Cambridge (1995)
7. Fritzke, B.: Growing Grid – A self-organizing network with constant neighbourhood range and adaptation strength. Neural Processing Letters 2(5), 9–13 (1995)
8. Gan, G., Ma, C., Wu, J.: Data Clustering: Theory, Algorithms and Applications. SIAM, Philadelphia (2007)
9. Gonzalez, F., Dasgupta, D.: Neuro-Immune and Self-Organizing Map Approaches to Anomaly Detection: A Comparison. In: Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS), Canterbury, UK, pp. 203–211 (September 2002)
10. Grosan, C., Abraham, A., Hassanien, A.: Designing Resilient Networks Using Multicriteria Metaheuristics. Telecommunication Systems: Modelling, Analysis, Design and Management 40(1), 75–88 (2008)
11. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice-Hall, Upper Saddle (1999)
12. Jolliffe, I.: Principal component analysis, 2nd edn. Springer, Berlin (2002)
13. Kaski, S., Lagus, K.: Comparing Self-Organizing Maps. In: Vorbrüggen, J.C., von Seelen, W., Sendhoff, B. (eds.) ICANN 1996. LNCS, vol. 1112, pp. 809–814. Springer, Heidelberg (1996)
14. Klement, P., Snášel, V.: Anomaly Detection in Emergency Call Data – the First Step to Intelligent Emergency Call System Management. In: Proceedings of the 1st International Conference on Intelligent Networking and Collaborative Systems (INCoS 2009), Barcelona, Spain, pp. 226–232 (November 2009)
15. Klement, P., Snášel, V.: SOM neural network - a piece of intelligence in disaster management. In: Nature & Biologically Inspired Computing, NaBIC 2009, pp. 872–877 (2009)
16. Kohonen, T.: Self-Organizing Maps. Springer, Berlin (1995)
17. Konak, A., Bartolacci, M.R.: Designing survivable resilient networks: a stochastic hybrid genetic algorithm approach, Omega; Special Issue on Telecommunications Applications 35(6), 645–658 (2007)
18. Lichodzijewski, P., Zincir-Heywood, A.N., Heywood, M.I.: Dynamic Intrusion Detection Using Self-Organizing Maps. In: The 14th Annual Canadian Information Technology Security Symposium, CITSS (2002)
19. Lloyd, S.P.: Least Squares Quantization in PCM. IEEE Transactions on Information Theory 28(2), 129–137 (1982)
20. Mitra, S., Pal, S.K.: Self-organizing neural network as a fuzzy classifier. IEEE Trans. Systems, Man, Cybernetics 24(3), 385–399 (1994)
21. Paprzycki, M., Abraham, A., Guo, R., Mukkamala, S.: Data Mining Approach for Analyzing Call Center Performance. In: Orchard, B., Yang, C., Ali, M. (eds.) IEA/AIE 2004. LNCS (LNAI), vol. 3029, pp. 1092–1101. Springer, Heidelberg (2004)
22. Ramadas, M., Ostermann, S., Tjaden, B.C.: Detecting anomalous network traffic with self-organizing maps. In: Vigna, G., Kruegel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 36–54. Springer, Heidelberg (2003)
23. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24(5), 513–523 (1988)

24. Viscovery®. SOMine 5.0. Copyright©1998-2007 by Viscovery Software GmbH, http://www.viscovery.net (cited February 1, 2010)
25. Yang, M.Y.: Extending the Kohonen self-organizing map networks for clustering analysis. Computational Statistics & Data Analysis 38, 161–180 (2001)
26. Ypma, A., Duin, R.P.W.: Novelty detection using self-organizing maps. Progress in Connectionist Based Information Systems 2, 1322–1325 (1998)
27. NVIDIA, Cuda c programming guide (August 2010), http://developer.download.nvidia.com/compute/cuda/3_1/toolkit/docs/NVIDIA_CUDA_C_ProgrammingGuide_3.1.pdf
28. Hager, G., Zeiser, T., Wellein, G.: Data access optimizations for highly threaded multi-core cpus with multiple memory controllers. In: IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, pp. 1–7 (2008), doi:10.1109/IPDPS.2008.4536341
29. Andrecut, M.: Parallel gpu implementation of iterative pca algorithms. Journal of Computational Biology 16(11), 1593–1599 (2009)
30. Preis, T., Virnau, P., Paul, W., Schneider, J.J.: Accelerated fluctuation analysis by graphic cards and complex pattern formation in financial markets. New Journal of Physics 11(9), 093024 (2009), http://stacks.iop.org/1367-2630/11/i=9/a=093024
31. Patnaik, D., Ponce, S.P., Cao, Y., Ramakrishnan, N.: Accelerator-oriented algorithm transformation for temporal data mining. In: IFIP International Conference on Network and Parallel Computing Workshops, pp. 93–100 (2009), doi:http://doi.ieeecomputersociety.org/10.1109/NPC.2009.26
32. NVIDIA, Cuda zone (August 2010), http://www.nvidia.com/object/cuda_home_new.html
33. Myklebust, G., Solheim, J.G., Steen, E.: Wavefront implementation of self organizing maps on renns. In: International Conference on Digital Signal Processing, Limassol, Cyprus, pp. 268–273 (1995)
34. Mann, R., Haykin, S.: A parallel implementation of Kohonen's feature maps on the warp systolic computer. In: Proc. IJCNN-90-WASH-DC, Int. Joint Conf. on Neural Networks, vol. II, pp. 84–87. Lawrence Erlbaum, Hillsdale (1990)
35. Openshaw, S., Turton, I.: A parallel kohonen algorithm for the classification of large spatial datasets. Comput. Geosci. 22(9), 1019–1026 (1996), doi:http://dx.doi.org/10.1016/S0098-30049600040-4
36. Wu, C.-H., Hodges, R.E., Wang, C.-J.: Parallelizing the self-organizing feature map on multiprocessor systems. Parallel Computing 17(6-7), 821–832 (1991), doi:10.1016/S0167-8191(05)80069-9, http://www.sciencedirect.com/science/article/B6V12-4GMBN2V-N/2/214aee0cc02fb79d05751b5ded84b784
37. Nordström, T.: Designing parallel computers for self organizing maps. In: Fourth Swedish Workshop on Computer System Architecture (1992)
38. Valova, I., Szer, D., Gueorguieva, N., Buer, A.: A parallel growing architecture for self-organizing maps with unsupervised learning. Neurocomput. 68, 177–195 (2005), http://dx.doi.org/10.1016/j.neucom.2004.11.025
39. Valova, I., MacLean, D., Beaton, D.: Identification of patterns via region-growing parallel som neural network. In: Fourth International Conference on Machine Learning and Applications, pp. 853–858 (2008), http://doi.ieeecomputersociety.org/10.1109/ICMLA.2008.50
40. Weigang, L., da Silva, N.: A study of parallel neural networks. In: International Joint Conference on Neural Networks, IJCNN 1999, vol. 2, pp. 1113–1116 (1999), doi:10.1109/IJCNN.1999.831112

41. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer (2000)
42. Flexer, A.: On the use of self-organizing maps for clustering and visualization, Intell. Intell. Data Anal. 5(5), 373–384 (2001)
43. Openmp: Api specification for parallel programming (2010), `http://openmp.org/wp/`
44. Lee, S., Min, S.-J., Eigenmann, R.: Openmp to gpgpu: a compiler framework for automatic translation and optimization. In: PPoPP 2009: Proceedings of the 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 101–110. ACM, New York (2009), doi:`http://dx.doi.org/http://doi.acm.org/10.1145/1504176.1504194`
45. Kirk, D., Mei Hwu, W.: Programming Massively Parallel Processors: A Hands-on Approach. Morgan Kaufmann (2010)

Author Index