Volume 39 Number 2 June 2015 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Special Issue:

Bioinspired Optimization

Guest Editors:

Jurij Šilc

Aleš Zamuda

Editorial Boards

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Higher Education, Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Editors' Introduction to the Special Issue on "Bioinspired Optimization"

The possibly changing and uncertain environment attracts and retains the fittest members of biological populations, which accumulate experience and improve, from adapting and competing among themselves. Their material of experience is exchanged and propagated from iteration to iteration according to the laws of nature. Relying on elementary activities of individuals, societies of these biological populations exhibit complex emergent behaviors.

Assemblies of genes, insects, bird flocks, and many other fascinating natural phenomena have been a rich source of inspiration in computer algorithms design for decades. Specifically, optimization is an area where these techniques are studied and exercised with particular practical success.

As a result bioinspired optimization algorithms (evolutionary algorithms, genetic algorithms, evolution strategies, evolutionary programming, genetic programming, ant colony optimization, particle swarm optimization, artificial immune systems, etc.) were designed to overcome the drawbacks of traditional algorithms in demanding application scenarios including those where little, if any, information is available to assist problem solving. The emerging challenges inspire new methods to be delivered and existing ones being introduced for specific tasks.

This special issue of Informatica – an International Journal of Computing and Informatics includes selected extended versions of student papers presented during:

- the Fifth International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2012) held in Bohinj, Slovenia, on 24–25 May 2012 and
- the Student Workshop on Bioinspired Optimization Methods and their Applications (BIOMA 2014), held in Ljubljana, Slovenia, on 13 September 2014.

After the selection and approval of the reviewing committee, this special issue presents seven valuable contributions. They were contributed by 21 co-authors coming from five countries (Germany, Romania, Slovenia, Turkey, and United Kingdom)

The first paper is entitled *Differential Evolution Control Parameters Study for Self-Adaptive Triangular Brushstrokes* and contributed by Aleš Zamuda and Uroš Mlakar. This work describes a lossy image representation where a reference image is approximated by an evolved image, constituted of variable number of triangular brushstrokes. Experimental results show the viability of the proposed encoding and optimization results with statistical tests that confirm the improved performance with the self-adaptation of the control parameters over the fixed control parameters.

The second paper, *Parallel Implementation of Desirability Function-Based Scalarization Approach for Multiobjective Optimization Problems*, contributed by Okkes Tolga Altinoz, Eren Akca, Asim Egemen Yilmaz, Anton Duca, and Gabriela Ciupriana, presents the results obtained for the parallel CUDA implementation of the previously proposed desirability-based scalarization approach for the solution of the multi-objective optimization problems. The CUDA implemented approach allows for roughly 20 times speedup compared to sequential implementation, provided a suitable number of solutions to be evaluated is given.

The third paper is *Using a Genetic Algorithm to Produce Slogans*, by Polona Tomašič, Gregor Papa, and Martin Žnidaršič. This paper describes a new solution based on the use of linguistic resources and evolutionary computing for invention of slogans. The approach utilizes a tool to check grammatical mistakes in trial solutions. A real case data is also studied, where slogans for Sentinel company are evolved.

The fourth paper, entitled *Comparing Evolutionary Operators and Search Spaces in the Construction of Facial Composites*, by Joseph James Mist, Stuart James Gibson, and Christopher John Solomon, addresses three experiments concerning the use of interactive evolutionary algorithms in the creation of facial composites. The approach was validated by roughly 20 participants using and assessing and it, thereby generating face-spaces, using different search algorithms, and assessing the comparison of different algorithms.

The fifth paper in this special issue is *Heuristics for Optimization of LED Spatial Light Distribution Model*, by David Kaljun, Darja Rupnik Poklukar, and Janez Žerovnik. This work presents a genetic algorithm and several versions of local search heuristics for optimization of a model of LED and secondary lens combination with symmetric spatial light distribution. They give a parameter and mechanisms combination study on the lighting task challenged. The yielding hybrid approach outperformed the standard genetic algorithm, and also outperformed a local search when inspected closely.

The sixth paper is entitled *Implicit and Explicit Averaging Strategies for Simulation-Based Optimization of a Real-World Production Planning Problem* and contributed by Juan Esteban Diaz and Julia Handl. This paper explores the impact of noise handling strategies on optimization performance in the context of a real-world production planning problem. Since the stochastic nature of the fitness values may impact on optimization performance, authors proposed explicit and implicit averaging strategies to address this issue. They show that under increased levels of fitness variability, a hybrid strategy starts to outperform pure implicit and explicit averaging strategies for evaluation of a real-world production planning problem.

Finally, the seventh paper *Data Mining-Assisted Parameter Tuning of a Search Algoritm* contributed, by Jurij Šilc, Katerina Taškova, and Peter Korošec, deals

with the problem of tuning the performance of a meta-heuristic search algorithm with respect to its parameters. The principle challenge here is how to provide meaningful settings for an algorithm, obtained as result of better insight in its behavior. They apply their approach in learning a model for the DASA algorithm and give some conclusions on the suggested parameters tuning based on the knowledge obtained, such as number of ants and the evaporation factor.

We would like to thank the authors of the papers for their individual contributions and all anonymous dedicated reviewers for their criticism and time to help us making final decisions. Without their valuable and strong support, we could not have made this special issue successful.

As Guest Editors, we hope the readers will find the Special Issue interesting and informative, as well as that the papers will stimulate further progress in the field of ‟Bioinspired Optimization”.

Jurij Šilc
Aleš Zamuda
Guest Editors

Differential Evolution Control Parameters Study for Self-Adaptive Triangular Brushstrokes

Aleš Zamuda and Uroš Mlakar
Faculty of Electrical Engineering and Computer Science, University of Maribor
Smetanova ulica 17, SI-2000 Maribor, Slovenia
E-mail: ales.zamuda@um.si, uros.mlakar@um.si

Keywords: differential evolution, evolutionary computer vision, evolutionary art, image-based modeling, self-adaptation, triangular brushstrokes

Received: December 1, 2014

This paper proposes a lossy image representation where a reference image is approximated by an evolved image, constituted of variable number of triangular brushstrokes. The parameters of each triangle brush are evolved using differential evolution, which self-adapts the triangles to the reference image, and also self-adapts some of the control parameters of the optimization algorithm, including the number of triangles. Experimental results show the viability of the proposed encoding and optimization results on a few sample reference images. The results of the self-adapting control parameters for crossover and mutation in differential evolution are also compared to results with keeping these parameters constant, like in a basic differential evolution algorithm. Statistical tests are furthermore included to confirm the improved performance with the self-adaptation of the control parameters over the fixed control parameters.

Povzetek: V članku je predlagana izgubna predstavitev slike, kjer je referenčna slika aproksimirana z evoluirano sliko, ki je sestavljena iz spremenljivega števila potez trikotniškega čopiča. Parametre vsake poteze čopiča optimiramo s pomočjo diferencialne evolucije, ki samoprilagaja trikotniške poteze na referenčno sliko in prav tako samoprilagaja nekatere krmilne parametre samega optimizacijskega algoritma, vključno s številom trikotnikov. Rezultati poizkusov kažejo primernost predlagane metode in rezultati optimizacije so prikazani za več izbranih referenčnih slik. Rezultati samoprilagodljivih krmilnih parametrov za diferecialno evolucijo so primerjani tudi z rezultati, kjer so ti parametri nespremenljivi, kot je to primer pri osnovnem algoritmu diferencialne evolucije. Dodatno so podani še statistični testi, ki nadalje potrjujejo izboljšanje kakovosti pristopa ob samoprilagajanju krmilnih parametrov v primerjavi s pristopom z nespremenljivimi krmilnimi parametri.

1 Introduction

In this paper, evolvable lossy image representation utilizing an image compared to its evolved generated counterpart image, is proposed. The image is represented using a variable number of triangular brushstrokes [7], each consisting of triangle vertices coordinates and color parameters. These parameters for each triangle brush are evolved using differential evolution [13, 4], which self-adapts the control parameters, including the proposed self-adaptation for the number of triangles to be used. Experimental results show the viability of the proposed encoding and evolution convergence for lossy compression of sample images. Since this paper is an extended version of [8], new additional results are included, where the experiments results with fixed control parameters for differential evolution are included to check and demonstrate the self-adaptation mechanism influence on results. The results show clear superiority of the proposed approach with the self-adaptive control parameters over the approach where its control parameters are fixed.

The approach presented is built upon and compared with [7], by addressing and also extending the original challenge. Namely, the challenge introduced in [7] uses triangles in trying to build an approximate model of an image [7]. The triangle is an efficient brush shape for this challenge, since it covers more pixels than a single point, and also allows overlaying and blending of colors over several regional surface pixels, which lines can not. Also, an arbitrary triangle shape is less constrained than any further point-approximated shape, and also other shapes can be built by combining several triangles. Instead of genetic programming in [7], in this paper differential evolution is used with a fixed size tree-like chromosome vector, which is cut-off self-adaptively to form codon and anti-codon parts of the chromosome. Also, our approach uses a modified challenge, where we can reconstruct the model for the reference image solely using the evolved model without using the reference image, whereas the [7] needs the reference image when drawing pixels to the canvas in deciding which pixels match the reference image for accepting them into the evolved canvas. Also, in this paper the triangle brushstroke encoding differs and is proposed especially designed for an efficient DE encoding.

In the following section, related work is presented, then the proposed approach is defined. In Section 4, the experimental results are reported. Section 5 concludes the paper with propositions for future work.

2 Related Work

In this section, related work on evolutionary computer vision, evolutionary art, image representation, and evolutionary optimization using differential evolution, are presented. These topics are used in the proposed method, defined in the next section.

2.1 Image-Based Modeling, Evolutionary Computer Vision, and Evolutionary Art

Image-based approaches to modeling include processing of images, e.g., two-dimensional, from which after segmentation certain features are extracted and used to represent a geometrical model [10]. For art drawings modeling, automatic evolutionary rendering has been applied [2, 12]. Heijer and Eiben evolved pop art two-dimensional scalable vector graphics (SVG) images [6] and defined genetic operators on SVG to evolve representational images using SVG, and also to evolve new images, different from source images, leading to new and surprising images for pop-art. Bergen and Ross [3] interactively evolved vector graphics images using genetic algorithm, where solid-coloured opaque or translucent geometric objects or mosaic tile effects with bitmap textures were utilized; they considered the art aspect of the evolved image and multiple possible outcomes due to evolution stochastics and concluded to investigate vector animation of the vectorized image.

In [14] animated artwork is evolved using an evolutionary algorithm. Then, Izadi et al. [7] evolved triangular brushstrokes challenge using genetic programming for two-dimensional images, using unguided and guided searches on a three or four branch genetic program, where roughly 5% similarity with reference images was obtained on average per pixel. In this paper, we build upon and compare our new approach with [7], by addressing and also extending this challenge. After extending the challenge, we optimize it using DE, which is described in the next section.

2.2 Evolutionary Optimization Using Differential Evolution

Differential evolution (DE) [13] is a floating-point encoding evolutionary algorithm for continuous global optimization. It has been modified and extended several times with various versions being proposed [5]. DE has also been applied to remote sensing image subpixel mapping [18], image thresholding [11], and for image-based modeling using evolutionary computer vision to reconstruct a spatial procedural tree model from a limited set of two dimensional images [16, 15]. DE mechanisms were also compared to other algorithms in several studies [17]. Neri and Tirronen in their survey on DE [9] concluded that, compared to the other algorithms, a DE extension called jDE [4], is superior to the compared algorithms in terms of robustness and versatility over a diverse benchmark set used in the survey. Therefore, we choose to apply jDE in this approach.

The original DE has a main evolutionary loop where a population of vectors is computed within each generation. For one generation, counted as g, each vector $\mathbf{x}_i$, $\forall i \in \{1, \ldots, NP\}$ in the current population of size NP, undergoes DE evolutionary operators, namely the mutation, crossover, and selection. Using these operators, a trial vector (offspring) is produced and the vector with the best fitness value is selected for the next generation. For each corresponding population vector, mutation creates a mutant vector $\mathbf{v}_{i,g+1}$ (*'rand/1'* [13]):

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F(\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}), \tag{1}$$

where the indexes r_1, r_2, and r_3 are random and mutually different integers generated in from set $\{1, \ldots, NP\}$, which are also different from i. F is an amplification factor of the difference vector, mostly within the interval $[0, 1]$. The term $\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}$ denotes a difference vector, which is named the amplified difference vector after multiplication with F. The mutant vector $\mathbf{v}_{i,g+1}$ is then used for recombination, where with the target vector $\mathbf{x}_{i,g}$ a trial vector $u_{i,j,g+1}$ is created, e.g., using binary crossover:

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1}, & \text{if } rand(0,1) \leq CR \\ & \text{or } j = j_{\text{rand}}, \\ x_{i,j,g} & \text{otherwise,} \end{cases} \tag{2}$$

where CR denotes the crossover rate, $\forall j \in \{1, \ldots, D\}$ is a j-th search parameter of D-dimensional search space, $rand(0, 1) \in [0, 1]$ is a uniformly distributed random number, and j_{rand} is a uniform randomly chosen index of the search parameter, which is always exchanged to prevent cloning of target vectors. The original DE [13] keeps the control parameters fixed, such as $F = 0.5$ and $CR = 0.9$ throughout optimization.

However, the jDE algorithm, which is a modification of the original DE, self-adapts the F and CR control parameters to generate the vectors $\mathbf{v}_{i,g+1}$ and $\mathbf{u}_{i,g+1}$, corresponding values F_i and CR_i, $\forall i \in \{1, \ldots, NP\}$ are updated prior to their use in the mutation and crossover mechanisms:

$$F_{i,g+1} = \begin{cases} F_{\text{l}} + rand_1 \times F_{\text{u}} & \text{if } rand_2 < \tau_1, \\ F_{i,g} & \text{otherwise,} \end{cases} \tag{3}$$

$$CR_{i,g+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2, \\ CR_{i,g} & \text{otherwise,} \end{cases} \tag{4}$$

where $\{rand_1, \ldots, rand_4\} \in [0, 1]$ are uniform random floating-point numbers and $\tau_1 = \tau_2 = 0.1$. Finally, the selection operator evaluates and compares the trial to current

vector and propagates the fittest:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{if } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \quad (5)$$

3 Differential Evolution for Self-Adaptive Triangular Brushstrokes

In this section, the encoding aspect, genotype-phenotpye rendering, and evaluation mechanisms of the proposed approach are defined.

3.1 Encoding Aspect

We encode an individual compressed image into a DE vector as follows. A DE vector $\mathbf{x} = (x_1, x_2, \ldots, x_{8T^{\max}}, F_i, CR_i, T_i^{\mathrm{L}}, T_i^{\mathrm{U}})$ is composed of floating-point scalar values packed sequentially as $\{x_j : \forall j \in \{1, \ldots, D+4\}\}$, starting with a triangles-coding part of length $D = 8T^{\max}$, and the rest are the self-adaptive control parameters of the vector to be used during the DE. The self-adaptive control parameters part of the $\mathbf{x}$ vector encodes and uses the scaling factor F and crossover rate CR as in the jDE [4]; then the T_i^{L}, $T_i^{\mathrm{U}} \in \{1, \ldots, T^{\max}\}$ control parameters follow.

The self-adaptive T_i^{L} and T_i^{U} control parameters determine index-wise triangles encoded in the vector $\mathbf{x}$ to be used for rendering the evolved image, i.e., the portion of $\mathbf{x}$ to render an image is $\{x_j : \forall j \in \{T_i^{\mathrm{L}}, \ldots, T_i^{\mathrm{U}}\}\}$.

In this paper, we propose to have the whole vector represent a triangle set, organized similar to serializing a tree as a linear vector in visiting nodes by depth-first search. However, the leaf nodes are mostly exposed to being cut-off, whereas the root node is encoded in the middle of the vector and the near-root nodes are therefore more protected in being retained, since they are more anchored due to cut-offs mostly around the codon edges. After being included into a new trial vector, all nodes have an equal probability of having their triangle data changed.

In this way, the T_i^{L} and T_i^{U} allow us to render only a sub-portion of the triangles set, similarly to taking an inseparable portion of a GP tree traversal as in [7]. This gives us an arbitrary length render set, and keeps the crossover of anti-codon to help us find the number of triangles $T_i \in \{1, \ldots, T^{\max}\}$, which is more suitable for image approximation:

$$T_i = \begin{cases} T_i^{\mathrm{U}} - T_i^{\mathrm{L}} + 1 & \text{if } T_i^{\mathrm{L}} < T_i^{\mathrm{U}} \\ (T^{\max} - T_i^{\mathrm{L}}) + T_i^{\mathrm{U}} & \text{otherwise.} \end{cases} \quad (6)$$

The T_i^{L} and T_i^{U} are updated similarly to the F_i control parameter:

$$T_{i,g+1}^{\mathrm{L}} = \begin{cases} \lfloor rand_1^{\mathrm{L}} \times T^{\max} \rfloor & \text{if } rand_2^{\mathrm{L}} < \tau^{\mathrm{L}}, \\ T_{i,g}^{\mathrm{L}} & \text{otherwise,} \end{cases} \quad (7)$$

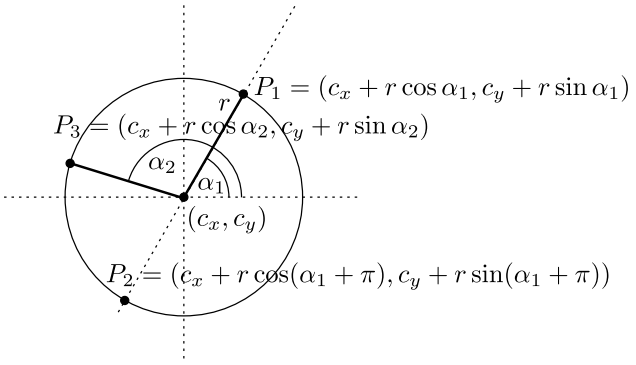


Figure 1: The triangle brush definition and the circumscribed circle.

$$T_{i,g+1}^{\mathrm{U}} = \begin{cases} \lfloor rand_1^{\mathrm{U}} \times T^{\max} \rfloor & \text{if } rand_2^{\mathrm{U}} < \tau^{\mathrm{U}}, \\ T_{i,g}^{\mathrm{U}} & \text{otherwise,} \end{cases} \quad (8)$$

where $\tau^{\mathrm{L}} = \tau^{\mathrm{U}} = \tau_1 = 0.1$ of the jDE.

3.2 Genotype-Phenotype Rendering

A DE vector $\mathbf{x}_i, \forall i \in \{1, \ldots, NP\}$ encoded using floating-point numbers $x_{i,j}, \forall j \in \{1, \ldots, D+4\}$ constituting a genotype is rendered into a phenotype image $\mathbf{z}_i = \{\mathbf{z}_{i,x,y}\}$ of R_{x} width and R_{y} height in pixels, to be compared against a reference image $\mathbf{z}^*$ as follows.

The triangle brushstrokes (Figure 1) are represented as $(c_x, c_y, r, \alpha_1, \alpha_2, b^{\mathrm{Y}}, b^{\mathrm{Cb}}, b^{\mathrm{Cr}})$, where $c_x \in [0, \ldots, R_{\mathrm{x}})$, $c_y \in [0, \ldots, R_{\mathrm{y}})$, and $r \in [0, R_{\mathrm{x}}/\sqrt{T_{\max}}]$ define the circumscribed circle center and radius for the triangle to be rendered; $\alpha_1 \in [1°, 360°)$ and $\alpha_2 \in [1°, 180°)$ define the vertices of this triangle on its circumscribed circle; and $b^{\mathrm{Y}} \in [16, 236)$, $b^{\mathrm{Cb}} \in [16, 241)$, and $b^{\mathrm{Cr}} \in [16, 241)$ are the color components of the brush for the triangle contained pixels.

The triangles' vertices coordinates encoded by i-th DE vector construct T_i triangles, each triangle $\mathbf{T}_k = (c_{x,k}, c_{y,k}, r_k, \alpha_{1,k}, \alpha_{2,k}), \forall k \in \{1, \ldots, T_i\}$ ($\mathbf{T}_k$ being packed as $\mathbf{x}_i = \{x_{i,j}\}$, $j = 8k + m$, $m \in \{1, \ldots, 8\}$), defining the vertices of a triangle $P_{1,k}$, $P_{2,k}$, and $P_{3,k}$:

$$P_{1,k} = \lfloor\, (c_{x,k} + r_k \cos\alpha_{1,k}, \; c_{y,k} + r_k \sin\alpha_{1,k}) \,\rfloor, \quad (9)$$

$$P_{2,k} = \lfloor\, (c_{x,k} + r_k \cos(\alpha_{1,k} + \pi), \; c_{y,k} + r_k \sin(\alpha_{1,k} + \pi)) \,\rfloor, \quad (10)$$

$$P_{3,k} = \lfloor\, (c_{x,k} + r_k \cos\alpha_{2,k}, \; c_{y,k} + r_k \sin\alpha_{2,k}) \,\rfloor. \quad (11)$$

The brush color $\mathbf{b}_k^{\mathrm{YCbCr}} = (b_k^{\mathrm{Y}}, b_k^{\mathrm{Cb}}, b_k^{\mathrm{Cr}})$ is first transformed into RGB color model as $\mathbf{b}_k^{\mathrm{RGB}} = (b_k^{\mathrm{R}}, b_k^{\mathrm{G}}, b_k^{\mathrm{B}})$ ($b_k^{\mathrm{R}}, b_k^{\mathrm{G}}, b_k^{\mathrm{B}} \in [0, 255]$), where:

$$b_k^{\mathrm{R}} = \lfloor 1.164(b_k^{\mathrm{Y}} - 16) + 1.596(b_k^{\mathrm{Cr}} - 128) \rfloor \quad (12)$$

$$b_k^{\mathrm{G}} = \lfloor 1.164(b_k^{\mathrm{Y}} - 16) - 0.813(b_k^{\mathrm{Cr}} - 128) - 0.391(b_k^{\mathrm{Cb}} - 128) \rfloor \quad (13)$$

$$b_k^{\mathrm{B}} = \lfloor 1.164(b_k^{\mathrm{Y}} - 16) + 2.018(b_k^{\mathrm{Cb}} - 128) \rfloor \quad (14)$$

For each triangle T_k, a solid color is rendered without antialiasing over the triangle brush area rasterizing [1] with a transparency factor of $1/T_i$:

$$\mathbf{b}_k = \left\lfloor \frac{255}{T_i} \mathbf{b}_k^{\mathrm{RGB}} \right\rfloor . \quad (15)$$

This is analogous to blending the triangle as a part-transparent layer within the evolved image $\mathbf{Z}_i = \sum_k \mathbf{z}_{k,x,y}$ and computes R, G, and B color layers for the pixels of the i-th individual:

$$\begin{aligned} \mathbf{z}_{k,x,y} &= \sum_{\mathbf{T}_k \text{ over } (x,y)} \mathbf{b}_{k,x,y} \\ &= \sum_{\mathbf{T}_k \text{ over } (x,y)} \left\lfloor \frac{255}{T_i} \mathbf{b}_{k,x,y}^{\mathrm{RGB}} \right\rfloor , \end{aligned} \quad (16)$$

where $\mathbf{T}_k$ over (x, y) denotes each triangle being rendered over the pixel (x, y) such that $\mathbf{b}_{k,x,y}$ contains the rendered pixels of a brushstroke. Triangles defined possibly over the edges of image canvas are drawn by clipping away pixels outside of the canvas area.

The initialization of a genotype is such that the $c_x, c_y, \alpha_1, \alpha_2, b^{\mathrm{Y}}, b^{\mathrm{Cb}}, b^{\mathrm{Cr}}, T_i^{\mathrm{L}}$, and T_i^{U} are initialized uniform randomly to integer values within their respective definition intervals, while r is kept as a floating-point. All parameters are however evolved as floating-point scalar values in DE.

3.3 Evaluation

Evaluation of the phenotype image $\mathbf{Z}_i$ to be compared against a reference image $\mathbf{Z}^*$ is as follows. A reference image $\mathbf{Z}^*$ is represented as RGB-encoded colored pixels integer values in layers $\mathbf{Z}^* = \{(z_{x,y}^{\mathrm{R}}, z_{x,y}^{\mathrm{G}}, z_{x,y}^{\mathrm{B}})\}$.

To obtain a difference assessment value, the following comparison metric is used for comparing an evolved image $\mathbf{Z} = \mathbf{Z}_i$ to $\mathbf{Z}^*$:

$$f(\mathbf{Z}) = 100\times \left(\frac{\sum_{y=0}^{R_{\mathrm{y}}-1} \sum_{x=0}^{R_{\mathrm{x}}-1} | z_{x,y}^{*\mathrm{R}} - z_{x,y}^{\mathrm{R}} |}{255 \times R_{\mathrm{x}} R_{\mathrm{y}}} + \frac{\sum_{y=0}^{R_{\mathrm{y}}-1} \sum_{x=0}^{R_{\mathrm{x}}-1} | z_{x,y}^{*\mathrm{G}} - z_{x,y}^{\mathrm{G}} |}{255 \times R_{\mathrm{x}} R_{\mathrm{y}}} + \frac{\sum_{y=0}^{R_{\mathrm{y}}-1} \sum_{x=0}^{R_{\mathrm{x}}-1} | z_{x,y}^{*\mathrm{B}} - z_{x,y}^{\mathrm{B}} |}{255 \times R_{\mathrm{x}} R_{\mathrm{y}}} \right) . \quad (17)$$

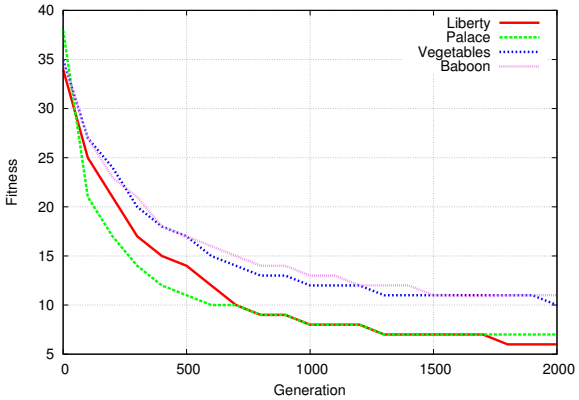


Figure 2: Fitness convergence, for best runs of each test image.

4 Experiments

The following experiments assess the viability of the approach on different control parameters, each with several independent runs. The parameter sets are as follows: the DE population size $NP = \{25, 50, 100\}$ and $T_{\max} = \{10, 20, \ldots, 150\}$, thereby for each run RNi={0, 1, …, 51} this counts for total of 45 parameter sets, i.e., 2340 independent runs. The NP and $T_{\max}$ are fixed during one run. The maximum number of function evaluations (MAXFES) used is same as with [7], MAXFES is 10^5. For image rendering, basic GDI+ is used.

4.1 Obtained Results

The obtained fitness values at the MAXFES termination of 10^5, over different parameters of $T_{\max}$ and NP, are seen in Tables 1 and 2. The best values obtained overall for an image are marked in bold underlined text font. The fitness convergence graphs for these best runs are seen in Figure 2, where after the initialization, the fitness is roughly below 40 (i.e., 40% similarity with reference), then drops below 15 for all test images and even further to slightly above 6 for two of them.

The convergent obtained results depend on the MAXFES used being same as with [7], but also NP and $T_{\max}$, as reported below. From Tables 1 and 2, we choose to report further evolved images up to MAXFES of 10^6 with all images. The best approximated images after MAXFES of 10^6 are shown in the Figure 3 which shows the evolution of the four images. In each line of Figure 3, the best fitting vectors upto MAXFES of 10^6 in generations $g = \{0, 100, 200, 400, 700, 1200, 2000\}$, and the final generation, are shown, then the rightmost the corresponding reference image. Figure 4 shows for each test image, dynamics of the number of triangle brushes in current best vector during generations, displaying varying convergent best T_i values across images.

Our approach searches for a representative image model and the values obtained such as 6.77, can roughly be compared to the 4.83 of [7]. Such representation of the problem

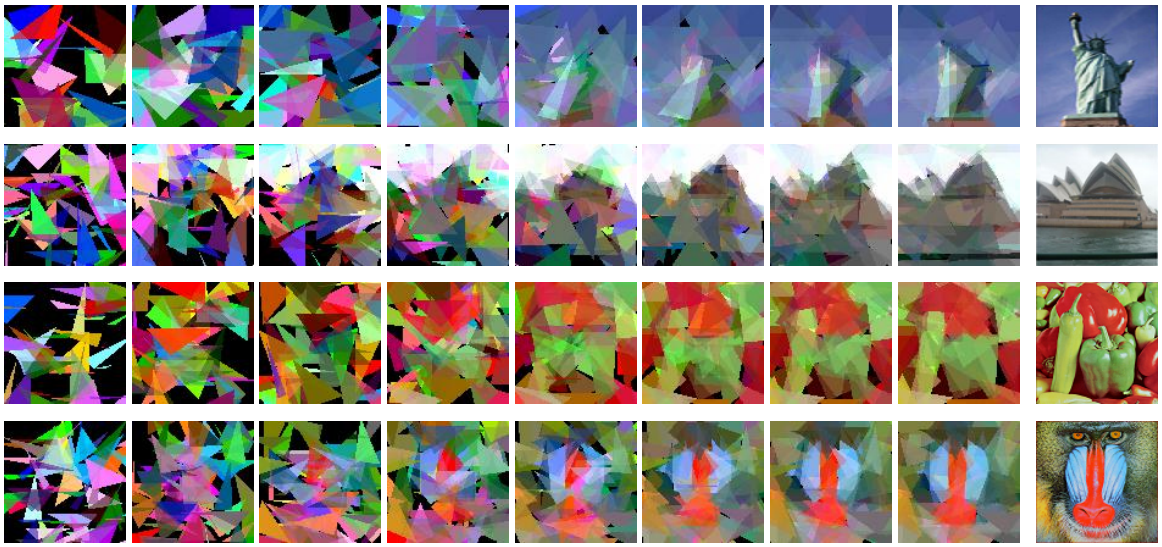


Figure 3: The evolved and the reference images (self-adaptive F and CR).

also makes our NP parameter have higher value, since we have no guided search and the problem is therefore more general. Also, our approach does not use a dynamically reallocatable morphable variable-size tree structure as in genetic programming encoding, inspite it rather uses a fixed size vector and limits its brushstrokes set by two simple bounds, making the approach faster for execution.

For comparison purposes and since this paper is an extended version of [8], following additional comparison is included. The algorithm is run again with fixed control parameters $F = 0.5$ and $CR = 0.9$ in DE, all other settings are kept same as with the proposed above approach.

Further, the results in Tables 1 and 2 are statistically tested using t-test with $alpha = 0.001$, against the null hypothesis, that the results obtained with fixed control parameters $F = 0.5$ and $CR = 0.9$ in DE, do not statistically differ. The symbol † with the values in bold text font signifies that the self-adaptive F and CR parameters approach results are significantly better and the symbol ‡ with values in italicized text font signifies that the fixed parameters approach results are significantly better. Comparing the statistics on the varied NP and $T_{\max}$ settings, DE with changing F and CR is 164 times better, 13 times worse, and 3 times with no significant performance difference, compared to the DE with $F = 0.5$, $CR = 0.9$.

The Figure 5, the best DE run with $F = 0.5$, $CR = 0.9$, nonetheless still shows self-adaptation of the T_i parameter – this is an additional indicator that the performance difference lines in the changing of the F and CR control parameters, which, compared to fixed values, improve the approach performance if they are self-adaptive.

Visually, the performance difference is observed from the rendered images in Figure 6, showing superiority of the proposed approach with self-adaptive control parameters over the approach using fixed control parameters. The Figure 7 shows fitness convergence of the best evaluated vector of the best DE run with $F = 0.5$, $CR = 0.9$, this

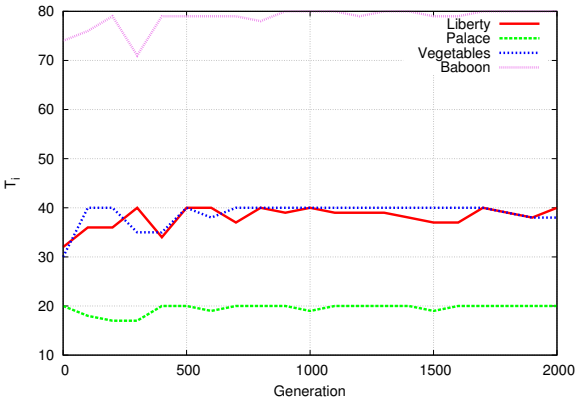


Figure 4: Number of brushstrokes in best vector, for best runs of each test image, self-adaptive F and CR parameters.

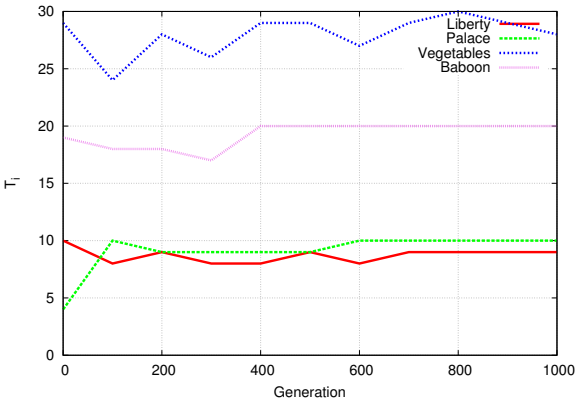


Figure 5: Number of brushstrokes in best vector, for best runs of each test image, $F = 0.5$, $CR = 0.9$.

Table 1: Obtained fitness over $T_{\max}$ and NP: test instances `Liberty` and `Palace`

		`Liberty`				`Palace`			
NP	$T_{\max}$	Best	Worst	Average	STD	Best	Worst	Average	STD
25	10	8.29	11.99	**9.93096**†	0.8233	8.69	13.69	**10.1362**†	0.9655
25	20	8.03	13.14	**10.0935**†	1.0845	7.83	11.5	**9.12173**†	0.8092
25	30	8.41	13.74	**10.0525**†	1.1712	7.52	11.1	**8.97942**†	0.7992
25	40	8.13	12.81	**10.4408**†	1.1416	7.34	11.36	**8.91788**†	0.8922
25	50	8.49	13.37	**10.6767**†	1.1768	7.65	12.53	**8.87442**†	0.9788
25	60	7.95	14.65	**10.9858**†	1.4284	7.9	11.88	**8.99673**†	0.8761
25	70	8.28	14.21	**11.4075**†	1.3630	7.79	13.17	**9.50327**†	1.0482
25	80	8.72	15.89	**11.7554**†	1.6330	7.97	12.34	**9.43558**†	0.9765
25	90	8.84	16.24	**12.1342**†	1.6608	8.41	13.54	**9.82**†	1.2756
25	100	9.01	16.74	**12.4798**†	1.7521	8.62	12.96	**9.83635**†	0.8869
25	110	8.07	16.78	**12.7412**†	1.7849	9.01	14.42	**10.4119**†	1.2468
25	120	9.67	16.14	**12.8467**†	1.7359	8.93	15.13	**10.3858**†	1.3149
25	130	10.16	17.96	**13.2692**†	1.7193	9.02	14.2	**10.2858**†	1.0292
25	140	9.29	17.99	**13.7029**†	1.7886	8.29	13.51	**10.7779**†	1.0299
25	150	10.82	18.56	**14.0373**†	1.6573	9.89	14.91	**11.1206**†	1.0586
50	10	7.51	9.69	**8.45077**†	0.4198	7.43	11.84	**8.68058**†	0.8825
50	20	6.78	8.99	**7.80173**†	0.4987	**7.1**	11.39	**8.79173**†	0.9592
50	30	6.89	9.17	**7.81788**†	0.5119	7.53	12.58	**9.75654**†	1.1186
50	40	**6.77**	9.87	**8.0375**†	0.6578	8.27	12.24	**10.0575**†	0.9537
50	50	7.08	10.61	**8.39923**†	0.7056	7.97	13.14	**10.3338**†	1.1009
50	60	7.15	10.4	**8.67115**†	0.7472	8.59	12.49	**10.7817**†	1.0754
50	70	7.46	10.9	**9.1025**†	0.8666	7.58	12.8	**10.7744**†	1.1086
50	80	7.6	11.4	**9.47981**†	0.8689	9.15	13.11	**11.3802**†	1.0178
50	90	8.05	12.65	**9.67346**†	0.9115	9.97	13.41	**11.5227**†	0.9315
50	100	8.75	11.75	**10.0152**†	0.7824	8.55	13.62	**11.4356**†	0.9923
50	110	8.93	13.63	**10.6356**†	0.9682	9.32	13.77	**12.0712**†	0.9579
50	120	9.22	13.01	**10.7502**†	0.9840	9.77	14.21	**12.429**†	0.8972
50	130	9.42	12.59	**11.0527**†	0.7707	11.37	14.07	**12.7387**†	0.6134
50	140	9.99	13.39	**11.5719**†	0.7815	9.69	15.5	**12.9317**†	0.9708
50	150	10.2	14.56	**12.2633**†	1.0702	9.58	15.36	**12.8092**†	1.1717
100	10	7.1	9.12	**7.98596**†	0.4241	7.91	13.88	**10.9573**†	1.8019
100	20	6.85	9.77	**7.83962**†	0.5360	8.86	14.59	**12.1117**†	1.2862
100	30	7.15	11.8	**8.49077**†	1.1563	9.59	16.15	**12.9098**†	1.0589
100	40	7.22	13	**8.86327**†	1.1092	9.65	14.97	**13.2477**†	1.1543
100	50	7.41	12.75	**9.34846**†	1.3939	11.01	15.52	**13.8606**†	0.9750
100	60	8.06	12.97	**9.77731**†	1.1539	11.5	16.14	**14.1856**†	1.1234
100	70	8.67	13.28	**10.1954**†	1.3722	10.77	16.32	**14.3629**†	1.1713
100	80	8.73	14.48	**11.0929**†	1.4093	10.98	17.06	**14.9348**†	1.1679
100	90	9.04	14.92	**11.3594**†	1.3483	11.1	16.8	**15.104**†	1.2586
100	100	9.4	16.13	**11.6604**†	1.4952	10.8	17.62	15.36	1.2330
100	110	10.17	15.68	**12.3365**†	1.5685	13.01	17.86	*16.0202*‡	0.9744
100	120	10.26	15.45	**12.3358**†	1.5076	11.07	17.99	*15.6113*‡	1.6455
100	130	10.22	16.19	**13.2212**†	1.6108	12.33	18.37	*16.4085*‡	1.3168
100	140	11.42	16.65	**13.7808**†	1.5502	11.64	18.35	*16.1229*‡	1.4990
100	150	11.35	18.68	**14.6113**†	1.9726	10.11	18.34	*16.2929*‡	2.0056

Table 2: Obtained fitness over $T_{\max}$ and NP: test instances `Vegetables` and `Baboon`

		Vegetables				Baboon			
NP	$T_{\max}$	Best	Worst	Average	STD	Best	Worst	Average	STD
25	10	14.13	17.21	**15.7269**†	0.7148	15.02	18.59	*16.38*‡	0.7128
25	20	12.56	18.03	**14.5658**†	0.9850	13.44	17.12	**15.3815**†	0.8129
25	30	12.33	15.98	**13.9215**†	0.8475	12.99	19.03	**15.0204**†	1.1150
25	40	11.62	16.21	**13.674**†	1.0436	11.99	16.85	**14.4342**†	1.0135
25	50	12.16	17.08	**13.88**†	1.0726	11.39	17.62	**14.4573**†	1.2299
25	60	11.64	17.88	**13.6438**†	1.2155	11.74	17.51	**14.8038**†	1.2229
25	70	11.29	17.15	**13.9056**†	1.3790	11.88	17.9	**14.6267**†	1.3495
25	80	11.61	16.6	**14.0871**†	1.3881	12.11	17.13	**14.3606**†	1.2815
25	90	11.63	17.96	**14.1062**†	1.4428	11.93	19.41	**14.6644**†	1.5269
25	100	11.34	17	**14.4533**†	1.4694	11.7	18.77	**14.7642**†	1.7438
25	110	11.74	19.66	**14.6085**†	1.7664	12.02	19.11	**15.0046**†	1.7605
25	120	12.26	17.91	**14.7737**†	1.5726	12.2	18.5	**15.6467**†	1.6086
25	130	12.1	19.75	**14.6338**†	1.9283	13.01	19.5	**15.4254**†	1.5505
25	140	11.94	19.01	**14.7635**†	1.6282	12.64	19.37	**15.8235**†	1.8458
25	150	12.82	18.7	**14.6487**†	1.3015	13.13	20.17	**15.7952**†	1.6923
50	10	13.03	15	**14.0723**†	0.4674	13.86	16.52	*14.9192*‡	0.5494
50	20	11.66	13.26	**12.4644**†	0.3184	11.8	14.54	**13.271**†	0.5569
50	30	11.12	13.59	**12.2425**†	0.6528	11.59	13.62	**12.5506**†	0.5732
50	40	**10.94**	14.1	**12.1848**†	0.6656	11.1	13.84	**12.3137**†	0.6090
50	50	11.04	13.92	**12.2946**†	0.7609	11.34	14.36	**12.4075**†	0.6304
50	60	11.29	15.86	**12.5506**†	0.9222	11.25	14.1	**12.3662**†	0.6161
50	70	11.18	15.21	**12.6104**†	0.8682	11.54	14.57	**12.5437**†	0.6510
50	80	11.32	15.26	**12.8619**†	0.7658	**11.07**	15.56	**12.9473**†	0.8087
50	90	11.84	15.28	**13.0077**†	0.8038	11.32	16.2	**12.857**†	1.0291
50	100	11.72	15.8	**13.5058**†	0.9565	11.85	15.72	**13.2658**†	0.7972
50	110	12.02	15.92	**13.5204**†	0.8750	11.98	15.56	**13.4275**†	0.7805
50	120	11.9	16.87	**13.829**†	1.1151	12.43	15.66	**13.5106**†	0.7265
50	130	12.51	15.97	**14.094**†	0.8855	12.64	16.32	**14.085**†	0.8259
50	140	12.16	17.07	**14.8198**†	1.2154	12.54	16.31	**14.15**†	0.8865
50	150	13.11	17.98	**14.9838**†	1.2072	13.08	18	**14.8765**†	1.0178
100	10	12.56	16.19	**13.9815**†	0.8083	13.49	16.19	*14.5367*‡	0.5672
100	20	11.84	16.45	**13.4704**†	1.0483	12.02	15.87	*13.8244*‡	0.8747
100	30	11.83	17.64	**13.9133**†	1.3335	12	15.76	*13.7206*‡	0.9727
100	40	12.01	17.95	**14.6354**†	1.3660	11.63	17.01	*13.6467*‡	1.3582
100	50	11.87	17.35	**14.9156**†	1.4272	11.99	17.48	*14.1658*‡	1.5554
100	60	12.32	18	**15.21**†	1.5119	12.12	17.46	*14.5021*‡	1.4517
100	70	12.13	18.05	**15.6513**†	1.2457	12.12	17.16	**14.3881**†	1.3782
100	80	12.9	18.86	**16.2008**†	1.4121	12.13	17.56	**14.8656**†	1.4214
100	90	12.32	20.04	**16.3233**†	1.7789	12.25	18.66	**15.2558**†	1.5144
100	100	12.98	20.55	**16.7275**†	1.7119	13.09	18.42	**15.5398**†	1.5064
100	110	13.76	20.18	**17.2896**†	1.5242	13	19.62	**15.84**†	1.6164
100	120	13.12	20.62	**17.626**†	1.5807	13.34	19.58	**16.4725**†	1.5223
100	130	13.52	20.12	17.9052	1.3516	13.84	19.6	**16.9367**†	1.7362
100	140	14.08	20.52	**18.216**†	1.6975	14.3	21	**17.4387**†	1.7372
100	150	14.97	21.19	19.1221	1.2128	14.75	21.13	**17.9488**†	1.6872

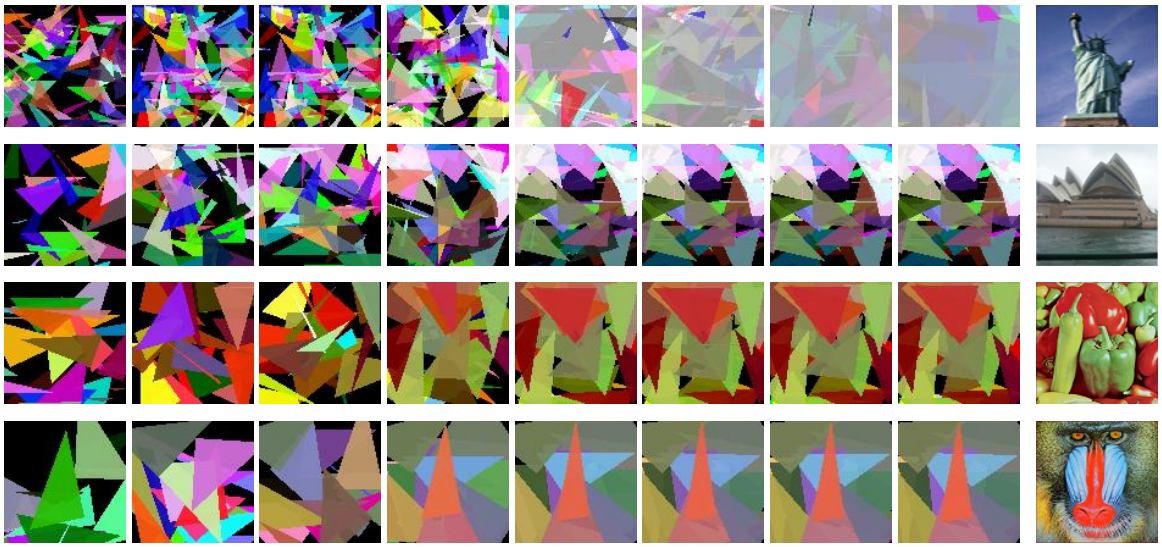


Figure 6: The evolved and the reference images, $F = 0.5$, $CR = 0.9$.

time with $NP = 100$ and therefore maximum generation number of 1000. The attained values tend to converge towards $T_{\max}$, but results are worse since the different $T_{\max}$, seen from Figures 4 and 5.

5 Conclusion

This paper presents an evolvable lossy image representation, approximating an image by comparing it to its evolved generated counterpart image. The image is represented using a variable number of triangular brushstrokes, each consisting of a triangle position and color parameters. These parameters for each triangle brush are evolved using differential evolution, which self-adapts the control parameters for mutation and crossover. Also, the proposed DE extension splits the DE vector in the codon and anticodon parts, where the triangles material is used only from the codon part, adjusting the genetic tree center and its borders, together with the number of triangle brushstrokes to be rendered. Experimental results show the viability of the proposed encoding and evolution convergence for the lossy representation of reference images, where fitness is displayed dependent on the population size, maximal number of function evaluations allowed, maximal number of triangles used in image representation, and different input reference images. While analyzing the NP and $T^{\max}$, moreover in this paper, we have shown that the self-adaptive jDE control parameters handling mechanism is preferable to the fixed control parameters mechanism from the original DE.

Future work can include increasing MAXFES, addressing different encoding aspects, evolutionary operators, control-parameters update, Euclidean distance for colors comparison, and more case studies on input images with different properties.

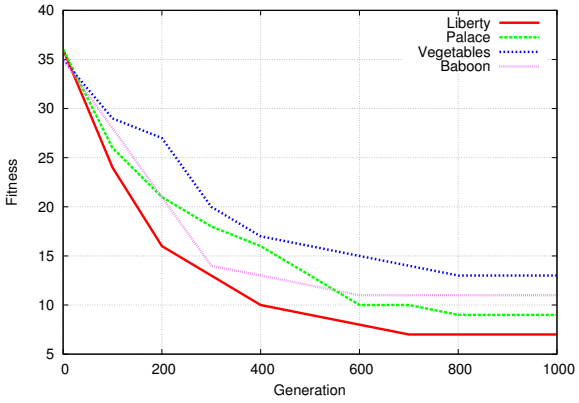


Figure 7: Fitness convergence, for best runs of each test image, $F = 0.5$, $CR = 0.9$.

Acknowledgement

This work is supported in part by Slovenian Research Agency, project P2-0041.

References

[1] B. D. Ackland, N. H. Weste (1981) The edge flag algorithm – a fill method for raster scan displays, *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 41–48.

[2] P. Barile, V. Ciesielski, M. Berry, K. Trist, (2009) Animated drawings rendered by genetic programming, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 939–946.

[3] S. Bergen, B. J. Ross (2012) Automatic and interactive evolution of vector graphics images with genetic algorithms, *The Visual Computer*, vol. 28, no. 1, pp. 35–45.

[4] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer (2006) Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems, *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657.

[5] S. Das, P. N. Suganthan (2011) Differential Evolution: A Survey of the State-of-the-art, *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31.

[6] E. den Heijer, A. E. Eiben (2012) Evolving pop art using scalable vector graphics, *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, Springer, pp. 48–59.

[7] A. Izadi, V. Ciesielski, M. Berry (2011) Evolutionary non photo-realistic animations with triangular brushstrokes, *AI 2010: Advances in Artificial Intelligence*, Springer, pp. 283–292.

[8] U. Mlakar, J. Brest, A. Zamuda (2014) Differential Evolution for Self-adaptive Triangular Brushstrokes, *Proceedings of the Student Workshop on Bioinspired Optimization Methods and their Applications (BIOMA)*, pp. 105–116.

[9] F. Neri, V. Tirronen (2010) Recent Advances in Differential Evolution: A Survey and Experimental Analysis, *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106.

[10] L. Quan (2010) *Image-Based Modeling*, 1st edition, Springer.

[11] S. Rahnamayan, H. R. Tizhoosh (2008) Image thresholding using micro opposition-based Differential Evolution (Micro-ODE), *Proceedings of the World Congress on Computational Intelligence (WCCI)*, pp. 1409–1416.

[12] J. Riley, V. Ciesielski (2010) Fitness landscape analysis for evolutionary non-photorealistic rendering, *Proceedings of the Congress on Evolutionary Computation (CEC)*, pp. 1–9.

[13] R. Storn, K. Price (1997) Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, *Journal of Global Optimization*, vol. 11, pp. 341–359.

[14] K. Trist, V. Ciesielski, P. Barile (2010) Can't see the forest: Using an evolutionary algorithm to produce an animated artwork. *Arts and Technology*, Springer, pp. 255–262.

[15] A. Zamuda, J. Brest (2014) Vectorized procedural models for animated trees reconstruction using differential evolution, *Information Sciences*, vol. 278, pp. 1–21.

[16] A. Zamuda, J. Brest, B. Bošković, V. Žumer (2011) Differential Evolution for Parameterized Procedural Woody Plant Models Reconstruction, *Applied Soft Computing*, vol. 11, no. 8, pp. 4904–4912.

[17] K. Zielinski, R. Laur (2007) Stopping criteria for a constrained single-objective particle swarm optimization algorithm, *Informatica*, vol. 31, no. 1, pp. 51–59.

[18] Y. Zhong, L. Zhang (2012) Remote sensing image subpixel mapping based on adaptive differential evolution, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 5, pp. 1306–1329.

Parallel Implementation of Desirability Function-Based Scalarization Approach for Multiobjective Optimization Problems

O. Tolga Altinoz
Ankara University, Electrical and Electronics Engineering, Turkey
E-mail: taltinoz@ankara.edu.tr

Eren Akca
HAVELSAN A.S., Ankara, Turkey
E-mail: eren.akca@havelsan.com.tr

A. Egemen Yilmaz
Ankara University, Electrical and Electronics Engineering, Turkey
E-mail: aeyilmaz@eng.ankara.edu.tr

Anton Duca and Gabriela Ciuprina
Politehnica University of Bucharest, Romania
E-mail: anton.duca@upb.ro, gabriela@lmn.pub.ro

Keywords: parallel implementation, CUDA, particle swarm optimization

Received: December 1, 2014

Scalarization approaches are the simplest methods for solving the multiobjective problems. The idea of scalarization is based on decomposition of multiobjective problems into single objective sub-problems. Every one of these sub-problems can be solved in a parallel manner since they are independent with each other. Hence, as a scalarization approach, systematically modification on the desirability levels of the objective values of multiobjective problems can be employed for solving these problems. In this study, desirability function-based scalarization approach is converted into parallel algorithm and applied into seven benchmark problems. The performance of parallel algorithm with respect to sequential one is evaluated based on execution time on different graphical processing units and central processing units. The results show that even the accuracy of parallel and sequential codes are same, the execution time of parallel algorithm is up to 24.5-times faster than the sequential algorithm (8.25-times faster on average) with respect to the complexity of the problem.

Povzetek: Pristopi s skalarizacijo sodijo med najenostavnejše načine reševanja večkriterijskih problemov. Zamisel skalarizacije temelji na dekompoziciji večkriterijskih problemov v enokriterijske podprobleme, ki jih lahko rešujemo sočasno, saj niso medsebojno odvisni. Torej lahko uporabimo za reševanje večkriterijskih problemov sistematično spreminjanje nivoja zaželenosti ciljnih vrednosti teh problemov. V tej študiji smo implementirali vzporedni način skalarizacije na osnovi funkcije zaželenosti in ga aplicirali na sedmih tesnih problemih. Učinek vzporednega algoritma glede na zaporednega smo ovrednotili z ozirom na čas izvajanja na različnih grafično-procesnih in centralno-procesnih enotah. Vzporedna različica daje enako natančne rezultate in je tudi do 24,5-krat hitrejša od zaporedne (8,25-krat v povprečju), glede na zahtevnost problema.

1 Introduction

The problem for determining the best possible solution set with respect to multiple objectives is referred to as a multiobjective (MO) optimization problem. There are many approaches for the solution of these kinds of problems. The most straightforward approach, the so-called "scalarization" or "aggregation" is nothing but to combine the objectives in order to obtain a single-objective [1].

Scalarization approaches are the simplest methods for solving the multiobjective problems. The idea of scalarization is based on decomposition of multiobjective problems into single objective sub-problems. The solutions of these single objective sub-problems form the Pareto approximation set. However, since the number of sub-problems is much higher than the number of objectives in multiobjective problem, and each problem is desired to be solved by single objective optimization algorithm, the computa-

tion time of scalarization approaches is much higher such that it becomes unfeasible to be solved by scalarization approaches. For each sub-problem, a specific number of function evaluations must be performed by a single objective optimization algorithm. Hence, a bunch of function evaluations are evaluated for solving multiobjective optimization problem. Before development of powerful multi-objective optimization algorithms such as the Non-Dominated Sorting Genetic Algorithm (NSGA) [2], NSGA-II [3] or Vector Evaluated Genetic Algorithm (VEGA) [4], scalarization techniques were preferred to solve engineering optimization problems. After the development of successful multi-objective optimization algorithms, scalarization techniques were considered to be old-fashioned, and they were abandoned due to the necessary of much higher number of function evaluations to obtain approximately same performance as multiobjective optimization algorithms. However, with the aid of parallel architectures and devices, it is possible to reconsider and revisit the scalarization techniques since these techniques are usually suitable for parallelization.

One of the scalarization approaches for a-priori process is defined with the aid of a desirability function in this study. Desirability function is integrated to the particle swarm optimization algorithm in order to normalize the joint objective function values [5]. Then, geometric mean of the desirability levels of each objective is computed in order to obtain a single value. For each sub-problem, the shape of the desirability function is shrunk. Therefore the desirability level is changed and the optimization results are also varied. At the end of this method, a set of possible solutions are composed. This set contains both the dominated and the non-dominated solutions. If necessary, the programmer might run a posterior method like non-dominated sorting for selecting the non-dominated solutions, as well. However, in this study, the main focus is to obtain the possible solution set. In this study, with a similar motivation, we demonstrate how one of these techniques can be parallelized and present performance of the approach by implementing on the Graphic Processing Units (GPUs) via the Compute Unified Device Architecture (CUDA) framework.

This paper is organized as follows: Section 2 explains the desirability function-based scalarization approach in detail and Section 3 presents a parallel implementation of the proposed method. Section 4 gives the implementation environment, benchmark problems and performance evaluation of the proposed method. The last section presents the conclusion and future work off the proposed method.

2 Desirability Function-Based Scalarization Approach

In a general manner, the desirability functions can be applied in order to incorporate the decision maker's preferences without any modification of the single-objective optimization algorithm. The decision maker chooses a desirability function and corresponding level. At each steps/iterations of the algorithm, instead of objective values; desirability index is calculated. At the end of the algorithm only a single solution is ready for collected by the decision maker. Even this method uses the advantages of desirability functions (Desirability functions are explained in Section 2.1) decision maker has small control on final result since a solution is obtained on a region defined by the desirability function (Figures 3 and 4) instead of on a line like weighted sum approach. However, in this study, by defining a systematical reduction approach, our aim is not to include or incorporate the preference of the decision maker but to present a generalized multi-objective optimization method for obtaining many possible solution candidates, that proposed method is applied as a scalarization approach like weighted sum method. Therefore a systematic approach was previously proposed by changing the shape of desirability functions by three of the authors of this paper [6]. For N objective problem, N numbers of desirability functions are selected with respect to the boundaries of the problem. Next, desirability functions are divided into levels and each level corresponding to one of the single objective implementation. For example of two objective problem case which was investigated in this paper, two desirability functions are defined and they are divided into same level (let's say 10) per function. Since there are two desirability functions defined, there are 100 single objective implementations in total. The previous study [6] show that the performance of the desirability function is greatly depends on the number of the levels, in other words the number of the single objective evaluations. Also the results obtained in the previous study are showed that, it is acceptable for bi-objective problems. However, still the performance of the proposed approach is greatly depends on the number of levels, which increases the total number of computation time. Hence, in this study, the parallel cores of CPU and GPU are using as computation units for single objective optimization algorithms, and the total evaluation times are recorded for comparison. The aim of this paper is to show the applicability of the proposed method with the aid of parallel architectures of CPU and GPU.

2.1 Desirability Function

The desirability function idea was first introduced by Harrington in 1965 for the multi-objective industry quality control. After the proposition of the desirability function concept, Deringer and Suich [7] introduced two different desirability function formulations, which become the fundamental equations of desirability functions. These two desirability function definitions are given by (1), (2) and (3), which are called one-sided and two-sided, respectively.

The parameters given in equations are as follows: y is the input, for our case it is the objective function value, $h_{\min}$, $h_{\max}$ and h_{med} are the minimum, maximum and the median acceptable values for the domain of the two-sided desirability function.

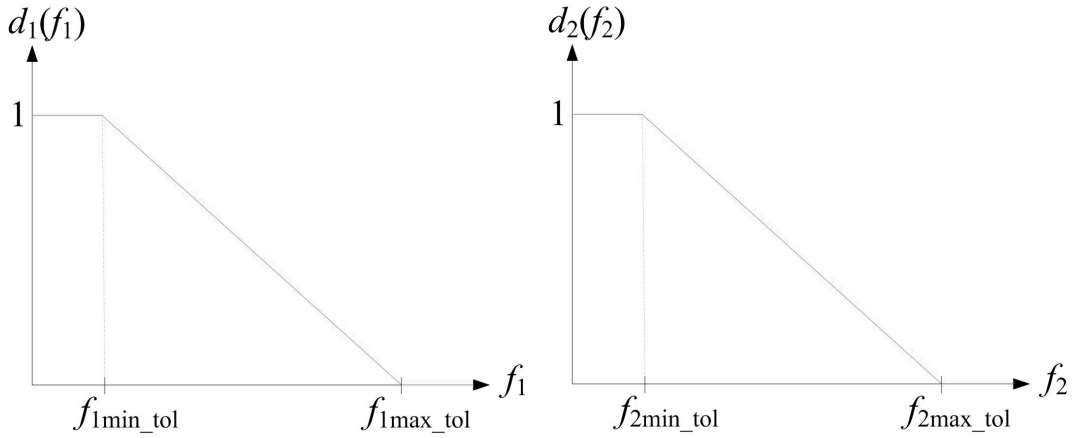


Figure 2: The linear desirability functions constructed for the bi-objective optimization problem.

$$d_1(y) = \begin{cases} 1, & y < h_{\min} \\ (\frac{y-h_{\max}}{h_{\min}-h_{\max}})^r, & h_{\min} < y < h_{\max} \\ 0, & y > h_{\max} \end{cases} \quad (1)$$

$$d_2(y) = \begin{cases} 0, & y < h_{\min} \\ (\frac{y-h_{\min}}{h_{\max}-h_{\min}})^r, & h_{\min} < y < h_{\max} \\ 1, & y > h_{\max} \end{cases} \quad (2)$$

$$d_3(y) = \begin{cases} 0, & y < h_{\min} \\ (\frac{y-h_{\min}}{h_{\text{med}}-h_{\min}})^t, & h_{\min} < y < h_{\text{med}} \\ (\frac{y-h_{\max}}{h_{\text{med}}-h_{\max}})^s, & h_{\text{med}} < y < h_{\max} \\ 0, & y > h_{\max} \end{cases} \quad (3)$$

The desirability level $d(y) = 1$ is the state for fully desirable, and $d(y) = 0$ is for a not-desired case. In this respect, d_1 one-sided desirability function is useful for minimization problem. The curve parameters are r, t and s. They are used in order to plot an arc instead of solid line, when desired. Curves plot in Figure 1 demonstrate the effects of the curve parameters and the graphs of the desirability functions.

2.2 Method of Desirability Function-Based Scalarization

The main idea beneath the desirability functions is as follows:

- The desirability function is a mapping from the domain of real numbers to the range set $[0, 1]$.
- The domain of each desirability function is one of the objective functions; and it maps the values of the relevant objective function to the interval $[0, 1]$.
- Depending on the desire about minimization of each objective function (i.e., the minimum / maximum tolerable values), the relevant desirability function is constructed.
- The overall desirability value is defined as the geometric mean of all desirability functions; this value is to be maximized.

Particularly, for a bi-objective optimization problem in which the functions f_1 and f_2 are to be minimized, the relevant desirability functions $d_1(f_1)$ and $d_2(f_2)$ can be defined as in Figure 2. The desirability functions are not necessarily defined to be linear; certainly, non-linear definitions shall also be made as described in [7].

Throughout this study, we prefer the linear desirability functions.

In [6], a method for extraction of the Pareto front was proposed by altering the shapes of the desirability functions in a systematical manner. Particularly by:

- Fixing the parameters $f_{1_{\text{max\_tol}}}$ and $f_{2_{\text{max\_tol}}}$ seen in Figure 2 at infinity, and
- Varying the parameters $f_{1_{\text{min\_tol}}}$ and $f_{2_{\text{min\_tol}}}$ systematically,

It is possible to find the Pareto front regardless of its convexity or concavity. This claim can be illustrated for the bi-objective case as follows: as seen in Figure 3, the parameters $f_{1_{\text{min\_tol}}}$ and $f_{2_{\text{min\_tol}}}$ determine the sector which is traced throughout the solution. The obtained solution corresponds to a point for which the geometric mean of the two desirability values. As seen in Figure 4, even in the case of concave Pareto front, the solution can be found without loss of generality. In other words, unlike the weighted-sum approach, the method proposed in [6] does not suffer from the concave Pareto fronts.

In [6], the applicability and the efficiency of the proposed scalarization approach was demonstrated via some multi-objective benchmark functions. Each single-objective problem (i.e., the scalarization scheme) was

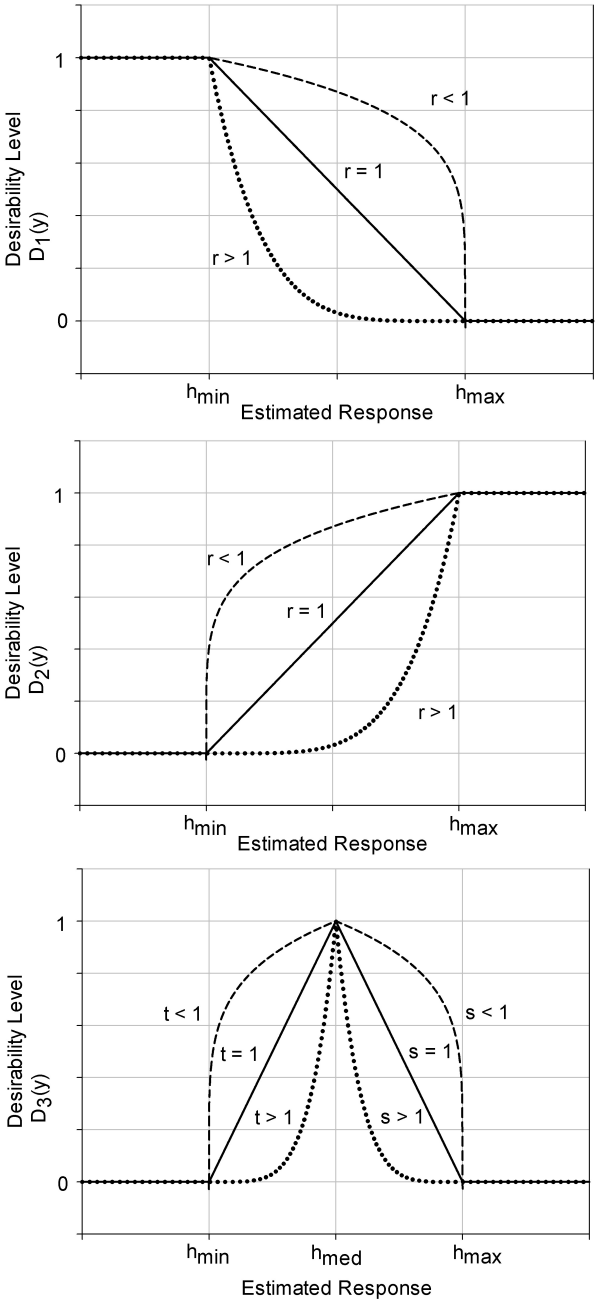


Figure 1: The graphical demonstration of the desirability functions.

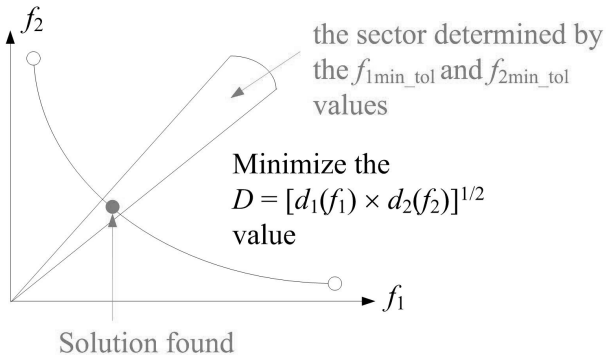


Figure 3: The solution via the desirability-function based approach for convex Pareto front.

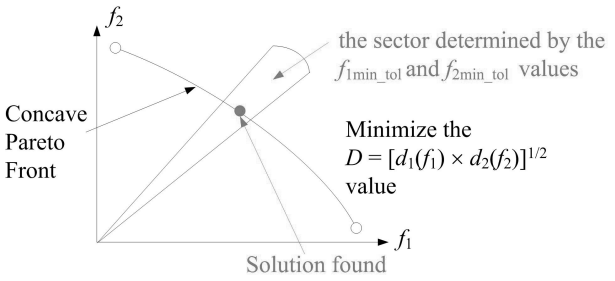


Figure 4: The solution via the desirability-function based approach for concave Pareto front.

solved with Particle Swarm Optimization. Despite no explicit demonstration or proof, it was claimed that:

- There were no limitations about the usage of Particle Swarm Optimization; i.e., any other heuristic algorithm could be incorporated and implemented.
- The proposed method can be easily parallelizable.

In this study, we demonstrate the validity of these claims by performing a parallel implementation on GPUs via the CUDA framework. The next section is devoted to the implementation details.

3 Parallel Multiobjective Optimization with GPU

This section is dedicated to explaining the steps and idea of parallelizing the Desirability function-based scalarization approach with the aid of CUDA library.

3.1 Fundamentals of CUDA Parallel Implementation

The researchers familiar with the programming languages used to desire a programming language or framework letting them write parallel codes easily. For this purpose in 2007, NVidia [8] introduced a software framework called CUDA. By means of this, a sequential function code can

be converted to a parallel kernel by using the libraries and some prefix expressions. By this way, the programmers do not need to learn a new programming language. They are able to use their previous know-how related to C/C++, and enhance this knowledge with some basic expressions introduced by CUDA. However, without the knowledge about the CUDA software and the parallel architecture hardware, it is not possible to write efficient codes.

CUDA programming begins with the division of the architectures. It defines the CPU as host and GPU as device. The parallel programming actually is the assignment of duties to parallel structure and collection of the results by CPU. In summary, the codes are written for CPU on C/C++ environment, and these codes include some parallel structures. These codes are executed by the host. Host commands device for code executed. When the code is executed by the device, the host waits until the job is finished, then a new parallel duty can be assigned, or results from the finished job can be collected by the host. Thus, the device becomes a parallel computation unit. Hence, parallel computing relies on the data movement between host and device. Eventhough both host and device are very fast computation units, the data bus is slower. Therefore, in order to write an efficient program, the programmer must keep his/her code for minimum data transfer between the host and the device.

The GPU has stream multiprocessors (SMs). Each SM has 8 stream processors (SPs), also known as cores, and each core has a number of threads. In tesla architecture there are 240 SPs, and on each SP has 128 threads, which is the kernel execution unit. The bodies of threads are called groups. The groups are performed collaterally with respect to the core size. If the GPU architecture has two cores, then two blocks of threads are executed simultaneously. If it has four cores, then four blocks are executed collaterally.

Host and device communicate via data movement. The host moves data to the memory of the GPU board. This memory is called global memory which is accessed from all threads and the host. The host has also access to constant and texture memories. However, it cannot access the shared memory, which is a divided structure assigned for every block. The threads within the block can access their own shared memory. The communication of the shared memory is faster than the global memory. Hence, a parallel code must contain data transfers to shared memory more often, instead of global memory.

In this study, random numbers are needed to execute the algorithm. Hence, instead of the rand() function of the C/C++ environment, CURAND library of the CUDA pack has been employed. In addition, the CUDA Event is preferred for accurate measurement of the execution time. In the next section, the parallel implementation of desirability function-based scalarization was explained in detailed.

3.2 Parallel Implementation of Desirability Function-Based Scalarization

The main idea of our parallel implementation throughout this study is illustrated in Figure 5.

Each scalarization scheme is handled in a separate thread; after the relevant solutions are obtained, they are gathered in a centralized manner to constitute the Pareto front from which the human decision maker picks a solution according to his/her needs. This approach ensures that the number of solutions found that can be found in parallel is limited by the capability of the GPU card used.

As stated before, we implemented the Particle Swarm Optimization Algorithm for verification of the aforementioned claims. The parallel CUDA implementation was compared to the sequential implementation on various GPUs and CPUs.

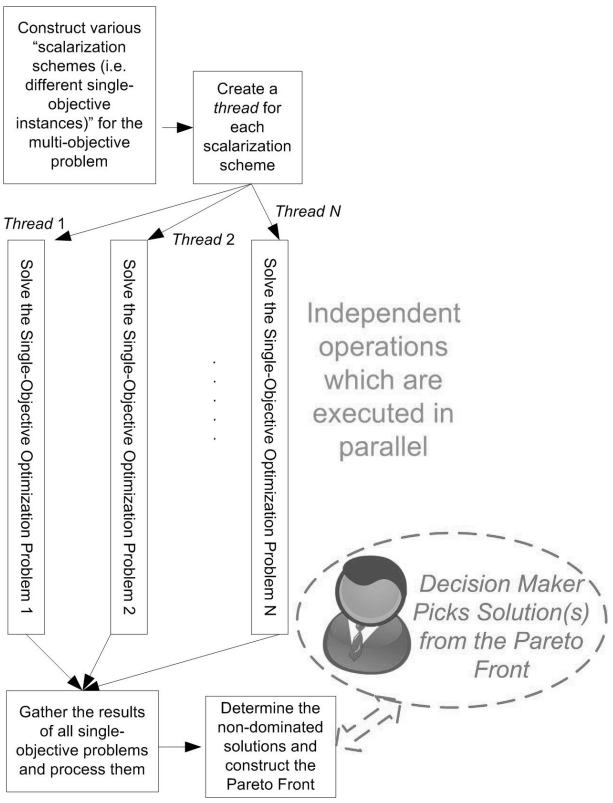


Figure 5: The parallel CUDA implementation of the desirability-function based approach.

It was seen that both implementations (sequential and parallel CUDA) were able to find the same solutions but in different elapsed times. As seen in Figure 6, if the number of Pareto front solutions increase, the advantage of the parallel CUDA increases dramatically.

Figure 6 presents parallel implementation of scalarization approach for the weighted sum method. The simple convex problem is selected and defined in (4) and (5) as a test bed for present the performance of the parallelization method for scalarization.

$$f_1(x) = x^2 \tag{4}$$

$$f_2(x) = (x-2)^2 \tag{5}$$

According to Figure 6, the performance of high and mid-level GPU cards are approximately 10-times faster than sequential implementation. The results obtained in Figure 6 yields the following conclusions:

- For a small number of Pareto solutions, CPU performs better against GPU
- After 64 solutions, parallel implementation presents better results than sequential code
- An old-fashion mobile GPU performs almost same as a relatively high level CPU.
- As the number of solution increases, the professional high level GPU devices perform more stable than general purpose GPUs.

4 Implementation, Results, and Discussion

The parallel desirability function-based scalarization approach was applied to solve seven benchmark problems. These problems are selected based on the complexity against execution time on computation unit. Since the average number of execution time is considered in the study, problems from simple calculation to problems with more branch and complex functions. In this section the benchmark problems and the results with respect to execution time is presented.

4.1 Benchmark Problems

In this study, ten benchmark problems [9] with different complexity and Pareto shape are selected to present the performance of the method. Table 1 gives the mathematical formulations of the problems. The performance comparison is performed not only on the accuracy of the results, but more importantly on the execution time. As given in Table 1 the complexity of the benchmark problems are given from simple to more complex problems. The reason behind is that as the complexity of the function is increased, the single processors have to accomplish much more calculations, and since the single processors on a GPU has lower capacity than CPU, it will be a good comparison for not only the number of solutions in solution space but also the problem complexity.

Table 1 presents as three columns. The first column gives the known-names of benchmark problems. The reader can be access amount of information about the function by searching by selecting keyword as function name. The second column is the mathematical formulation of the function. As the order of row increases the complexity of the function also increases. The last column is for the defines of the range of the decision variables.

4.2 Implementation Results

Table 2 presents the execution time comparison of CPU (Xeon E2620) and GPU (Tesla K20) for various numbers of levels from 8×8 to 100×100, number of single objective evaluations are 64 and 104 respectively. For low complex problems, until 225 numbers of levels (400 levels need for hard problems), the CPU outperforms GPU implementation with respect to execution time. It is reasonable since only small portion of cores on GPU can be used. But lower number of relatively very fast cores are finished the executions earlier than GPU. From 400 to $6,400$ levels, GPU computation time of parallel codes exceeds CPU time. At $6,400$ levels, the difference between CPU and GPU is at the peak grade. After that level, the advantage of GPU reduces. In other words, the GPU implementation acts more sequentially, since there are not any empty resources to execute parallel implementation. Among all of the problems, UF1 is the hardest for GPU implementation since the computation time is the longest for this problem. The main reasons are that: a) checking mechanism for even and odd parts that adds branch to the code, b) square of the trigonometric function. for GPU implementation branch are the time consuming programming codes such that in an if-else, both parts are evaluated by the architecture, that reduces the resources.

The average execution time of CPU is 8.25-times slower than average GPU execution time. The following results are obtained for comparison the execution time:

- For a small number of solutions, CPU outperforms GPU
- The increase on CPU execution time is proportional to the number of solutions. Hence, the execution time on CPU increases.
- The GPU implementations are much beneficial for overall comparison.
- For a very high number of solutions, the improvements obtained in GPU slowly decreases since GPU contains limited number of stream (multi)processors. At some point the improvements are not lower than $\approx$ 10-times on average.

5 Conclusion

In this study, desirability function-based scalarization approach is evaluated in a parallel fashion. Since the performance of sequential and parallel implementations are similar to each other, the execution time of these codes are compared based on different number of solutions. The results show that, for small number of solutions, parallel implementation is slower when compared to sequential implementation. But as the number of solution increases, the

Table 1: Multiobjective benchmark problems

Function name	*Mathematical description*	*Decision variable range*
ZDT1	$f_1(x) = x_1$ $f_2(x) = g(1 - \sqrt{\frac{f_1}{g}})$ $g = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	$0 \leq x_i \leq 1$
ZDT2	$f_1(x) = x_1$ $f_2(x) = g(1 - (\frac{f_1}{g})^2)$ $g = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	$0 \leq x_i \leq 1$
ZDT3	$f_1(x) = x_1$ $f_2(x) = g(1 - \sqrt{\frac{f_1}{g}} - \frac{x_1}{g}\sin(10\pi x_1))$ $g = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	$0 \leq x_i \leq 1$
UF1	$f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{i\in J_1}(x_i - \sin(6\pi x_1 + \frac{i\pi}{n}))^2$ $f_2(x) = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{i\in J_2}(x_i - \sin(6\pi x_1 + \frac{i\pi}{n}))^2$ $J_1 = \{i\mid i \text{ is odd and } 2 \leq i \leq n\}, J_2 = \{i\mid i \text{ is even and } 2 \leq i \leq n\}$	$0 \leq x_i \leq 1$ $-1 \leq x_{i-1} \leq 1$
UF2	$f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{i\in J_1} y_i^2$ $f_2(x) = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{i\in J_1} y_i^2$ $y_i = \begin{cases} x_i - (0.3x_1^2\cos(24\pi x_1 + \frac{4i\pi}{n}) + 0.6x_1)\cos(6\pi x_1 + \frac{i\pi}{n}), & i \in J_1 \\ x_i - (0.3x_1^2\cos(24\pi x_1 + \frac{4i\pi}{n}) + 0.6x_1)\sin(6\pi x_1 + \frac{i\pi}{n}), & i \in J_2 \end{cases}$	$0 \leq x_i \leq 1$ $-1 \leq x_{i-1} \leq 1$
UF3	$f_1(x) = x_1 + \frac{2}{\lvert J-1\rvert}((4\sum_{i\in J_1} y_i^2) - (2\prod_{i\in J_1}\cos(\frac{20y_i\pi}{\sqrt{i}})) + 2)$ $f_2(x) = 1 - \sqrt{x_1} + \frac{2}{\lvert J-2\rvert}((4\sum_{i\in J_2} y_i^2) - (2\prod_{i\in J_2}\cos(\frac{20y_i\pi}{\sqrt{i}})) + 2)$ $y_i = x_i - x_1^{0.5(1+\frac{3(i-2)}{n-2})}$	$0 \leq x_i \leq 1$
UF4	$f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{i\in J_1} h(y_i)$ $f_2(x) = 1 - x_1^2 + \frac{2}{\lvert J_2\rvert}\sum_{i\in J_2} h(y_i)$ $y_i = x_i - \sin(6\pi x_1 + \frac{i\pi}{n}),\ h(t) = \frac{t}{1+e^{2t}}$	$0 \leq x_i \leq 1$ $-2 \leq x_{i-1} \leq 2$

Table 2: Execution time comparison [seconds] of benchmark functions, where improvement, $impr$, is the scale factor shows how many times the GPU is faster than CPU, so that if $impr < 1$ means CPU is faster than GPU

# of levels for 2 desirability functions	Devices & $impr$	ZDT1	ZDT2	ZDT3	UF1	UF2	UF3	UF4	Average
	CPU	0.133	0.109	0.19	0.11	0.109	0.109	0.094	0.1220
8×8	GPU	0.433	0.4504	0.483	0.4917	0.4861	0.4906	0.408	0.4633
	$impr$	0.3072	0.2420	0.3934	0.2237	0.2242	0.2222	0.2304	**0.2633**
	CPU	0.221	0.153	0.291	0.222	0.199	0.197	0.168	0.2073
10×10	GPU	0.439	0.451	0.4848	0.4934	0.49	0.4914	0.405	0.4649
	$impr$	0.5034	0.3392	0.6002	0.4499	0.4061	0.4009	0.4148	**0.4450**
	CPU	0.446	0.333	0.576	0.42	0.418	0.413	0.372	0.4254
15×15	GPU	0.4424	0.4576	0.4904	0.499	0.4944	0.4967	0.409	0.4699
	$impr$	1.0081	0.7277	1.1746	0.8417	0.8455	0.8315	0.9095	**0.9055**
	CPU	0.8	0.564	0.997	0.717	0.706	0.728	0.811	0.7604
20×20	GPU	0.4281	0.442	0.4781	0.5	0.4977	0.5	0.4146	0.4658
	$impr$	1.8687	1.2760	2.0853	1.4340	1.4185	1.4560	1.9561	**1.6421**
	CPU	1.21	0.893	1.521	1.12	1.444	1.114	0.987	1.1841
25×25	GPU	0.4393	0.4573	0.491	0.5	0.4954	0.499	0.408	0.4700
	$impr$	2.7544	1.9528	3.0978	2.2400	2.9148	2.2325	2.4191	**2.5159**
	CPU	1.753	1.266	2.279	1.582	1.589	1.59	1.428	1.6410
30×30	GPU	0.4424	0.4566	0.4871	0.501	0.4973	0.4979	0.4132	0.4708
	$impr$	3.9625	2.7727	4.6787	3.1577	3.1953	3.1934	3.4560	**3.4880**
	CPU	3.162	2.186	4.094	2.794	2.854	2.757	2.508	2.9079
40×40	GPU	0.4451	0.453	0.4893	0.4999	0.4983	0.4991	0.4151	0.4714
	$impr$	7.1040	4.8256	8.3671	5.5891	5.7275	5.5239	6.0419	**6.1684**
	CPU	4.879	3.431	6.138	4.412	4.382	4.298	3.889	4.4899
50×50	GPU	0.4488	0.4639	0.4967	0.5119	0.5	0.501	0.4321	0.4792
	$impr$	10.8712	7.3960	12.3576	8.6189	8.7640	8.5788	9.0002	**9.3695**
	CPU	6.946	4.798	9.492	6.236	6.411	6.391	6.233	6.6439
60×60	GPU	0.4709	0.4864	0.518	0.5287	0.518	0.519	0.4587	0.5000
	$impr$	14.7505	9.8643	18.3243	11.7950	12.3764	12.3141	13.5884	**13.2876**
	CPU	9.52	6.764	11.959	8.566	8.548	8.562	7.592	8.7873
70×70	GPU	0.4995	0.5144	0.5417	0.5489	0.539	0.5435	0.4923	0.5256
	$impr$	19.0591	13.1493	22.0768	15.6058	15.8590	15.7534	15.4215	**16.7036**
	CPU	12.488	8.87	15.892	11.11	11.366	11.538	13.307	12.0816
80×80	GPU	0.6179	0.6321	0.6488	0.6388	0.635	0.6362	0.607	0.6308
	$impr$	20.2104	14.0326	24.4945	17.3920	17.8992	18.1358	21.9226	**19.1553**
	CPU	15.776	11.246	20.027	14.039	14.138	14.053	14.583	14.8374
90×90	GPU	0.8299	0.854	0.8749	0.8424	0.84	0.8432	0.8335	0.8454
	$impr$	19.0095	13.1686	22.8906	16.6655	16.8310	16.6663	17.4961	**17.5325**
	CPU	19.2579	13.863	24.504	17.252	19.219	17.74	15.49	18.1894
100×100	GPU	1.1157	1.149	1.1812	1.12	1.222	1.125	1.132	1.1493
	$impr$	17.2608	12.0653	20.7450	15.4036	15.7275	15.7689	13.6837	**15.8078**

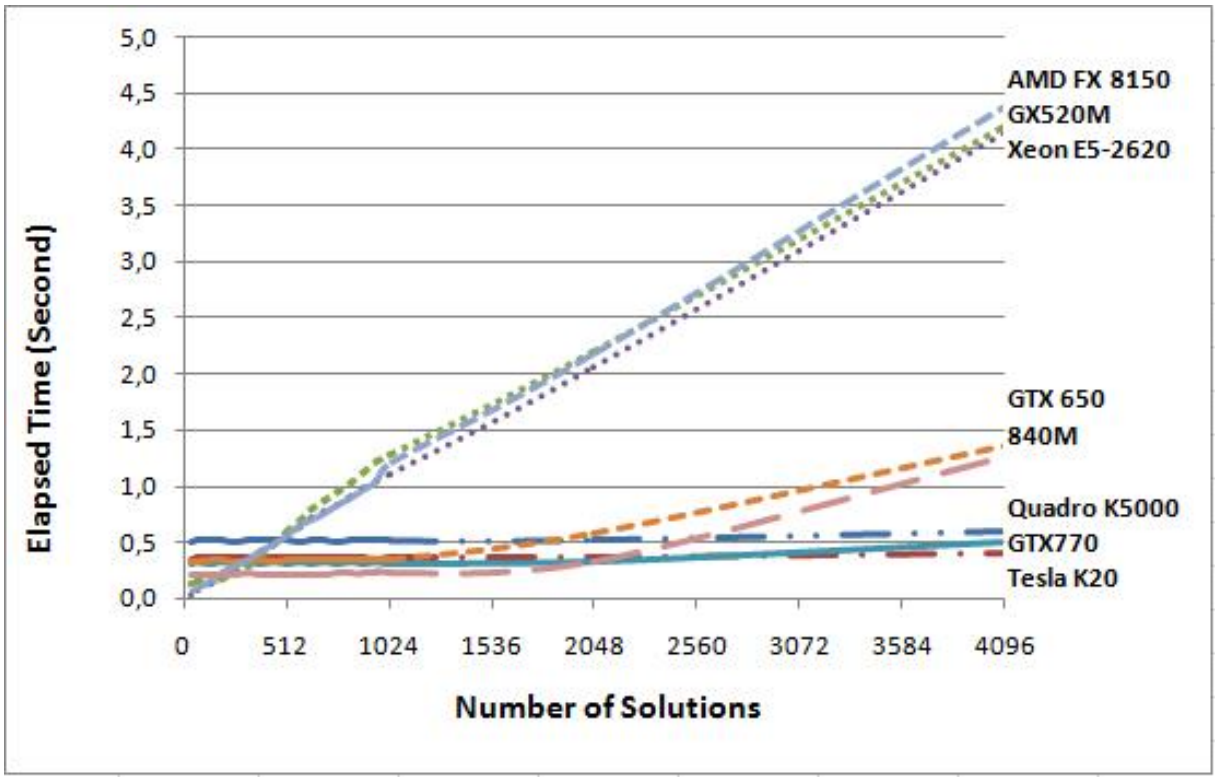


Figure 6: Comparison of the sequential Java and the parallel CUDA implementations.

GPU is almost 20-times faster than sequential implementation.

Acknowledgement

This study was made possible by grants from the Turkish Ministry of Science, Industry and Technology (Industrial Thesis – San-Tez Programme; with Grant Nr. 01568.STZ.2012-2) and the Scientific and Technological Research Council of Turkey - TÜBITAK (with Grant Nr. 112E168). The authors would like to express their gratitude to these institutions for their support.

References

[1] R. Marler, S. Arora (2009) Transformation methods for multiobjective optimization, *Engineering Optimization*, vol. 37, no. 1, pp. 551–569.

[2] N. Srinivas, K. Deb (1995) Multi-Objective function optimization using non-dominated sorting genetic algorithms, *Evolutionary Computation*, vol. 2, no. 3, pp. 221-–248.

[3] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-–197.

[4] J. D. Schaffer (1985) Multiple objective optimization with vector evaluated genetic algorithms, *Proceedings of the International Conference on Genetic Algorithm and their Applications*, pp. 93–100.

[5] J. Branke, K. Deb (2008) Integrating user preferences into evolutionary multiobjective optimization, *Knowledge Incorporation in Evolutionary Computing*, Springer, pp. 461–478.

[6] O. T. Altinoz, A. E. Yilmaz, G. Ciuprina (2013) A Multiobjective Optimization Approach via Systematical Modification of the Desirability Function Shapes, *Proceedings of the 8th International Symposium on Advanced Topics in Electrical Engineering*.

[7] G. Derringer, R. Suich (1980) Simultaneous optimization of several response variables,*Journal of Quality Technology*, vol. 12, no. 1, pp. 214–219.

[8] NVIDIA Corporation (2012) *CUDA dynamic parallelism programming*, NVIDIA.

[9] E. Ziztler, K. Deb, L. Thiele (2000) Comparison of multiobjective evolutionary algorithms: Empirical results, *Evolutionary Computation Journal*, vol. 8, no. 2, pp. 125–148.

Using a Genetic Algorithm to Produce Slogans

Polona Tomašič
XLAB d. o. o., Pot za Brdom 100, SI-1000 Ljubljana, Slovenia and
Jožef Stefan International Postgraduate School, Jamova cesta 39, SI-1000 Ljubljana, Slovenia
E-mail: polona.tomasic@xlab.si

Gregor Papa
Computer Systems Department, Jožef Stefan Institute, Jamova cesta 39, SI-1000 Ljubljana, Slovenia and
Jožef Stefan International Postgraduate School, Jamova cesta 39, SI-1000 Ljubljana, Slovenia
E-mail: gregor.papa@ijs.si

Martin Žnidaršič
Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, SI-1000 Ljubljana, Slovenia and
Jožef Stefan International Postgraduate School, Jamova cesta 39, SI-1000 Ljubljana, Slovenia
E-mail: martin.znidarsic@ijs.si

Keywords: genetic algorithm, slogan generation, computational creativity, linguistic resources

Received: December 1, 2014

Creative tasks, such as creation of slogans for companies, products or similar entities, can be viewed from the combinatorial perspective – as a search through the space of possible combinations. To solve such a combinatorial optimization problem, we can use evolutionary algorithms. In this paper, we present our solution for generation of slogans based on a genetic algorithm and linguistic resources. We also compare it to the unguided slogan generator.

Povzetek: Na kreativne naloge, kot je snovanje sloganov za podjetja in produkte, lahko gledamo s kombinatoričnega vidika – kot na iskanje v prostoru možnih kombinacij. Za reševanje tovrstnih kombinatoričnih optimizacijskih problemov lahko uporabljamo evolucijske algoritme. V tem članku predstavljamo rešitev za generiranje sloganov na podlagi genetskega algoritma in jezikovnih virov. Predstavljeno rešitev primerjamo tudi z generatorjem sloganov brez vodenja.

1 Introduction

Automated generation of slogans is a problem from the field of Computational Creativity [5]. There are very few studies dedicated to slogan generation. In fact, the only one we came across is the BRAINSUP framework [19], which is based on beam search through a carefully defined space of possible slogans. This space gets reduced by applying user specified constraints on keywords, domain, emotions, and other properties of slogans.

High quality slogans are often a result of group brainstorming. Several individuals present their ideas and the proposed slogans are then mixed into new slogans, and some new ideas emerge. This brainstorming process is similar to the evolution, from which we got the idea of using evolutionary algorithms for slogan generation. The initial slogans from brainstorming represent an initial population, mixing the best proposed slogans represents recombination, and new included ideas represent mutations. Evolutionary algorithms have already been applied to different natural language processing problems [2].

In this paper, we present our slogan generation procedure which is not influenced by the user in any way, apart from being provided with a short textual description of the target entity. The method is based on a genetic algorithm (GA) [3]. Genetic algorithms are the most traditional evolutionary algorithms and they ensure a good coverage of the search space. They have been successfully used for generating recipes [17], poetry [13] and trivial dialog phrases [16]. However, genetic algorithms have not been previously used for slogan generation. Our method follows the BRAINSUP framework in the initial population generation phase, and it uses a collection of heuristic slogan functions in the evaluation phase.

We tested our slogan generator and compared it to the random slogan generator. The statistical results are in favor of our method. However, even though the generated slogans can present a good starting point for brainstorming, their quality is not yet at the desired level.

The rest of the paper is organized as follows. In Section 2 we present the linguistic and semantic resources used in our solution. Section 3 provides a detailed description of the entire slogan generation process. It includes description of the evaluation functions and it clarifies the differ-

ence between the slogan generator and the unguided slogan generator. The performed experiments and the discussion of the results are presented in Section 4. The conclusions are drawn in Section 5.

2 Resources

Linguistic and semantic resources are a prerequisite for any kind of text generation. We use them at several steps of our method – for generation of initial population, mutation, and evaluation. Some are available as extended libraries for programming languages, others are available for download from the Internet, and some databases were created by ourselves. The origin of the data and the process is briefly described in the following paragraphs.

1. **Database of famous slogans:** it serves as a basis for the initial population generation and for comparison with generated slogans. It contains 5,249 famous slogans obtained from the Internet.

2. **Database of frequent grammatical relations between words in sentences:** for its acquisition we used the Stanford Dependencies Parser [14]. Stanford dependencies are triplets containing two words and the name of the relation between them. The parser also provides part-of-speech (POS) tags and phrase structure trees. To get representatives of frequent grammatical relations between words, we parsed 52,829 random Wikipedia pages, sentence by sentence, and obtained 4,861,717 different dependencies.

3. **Database of slogan skeletons:** slogan skeletons were obtained by parsing famous slogans with the Stanford Dependencies Parser. A slogan skeleton contains information about each position in the sentence – its POS tag and all its dependence relations with other words in the sentence. It does not contain any content words, only stop words. An example of a skeleton is shown in Figure 1.

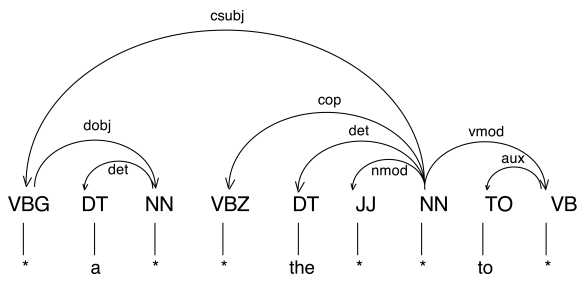


Figure 1: Example of a skeleton.

3 Slogan Generation

An input of our slogan generator is a short textual description about the target entity. It is the only required input from a user. It is used to obtain the name of the target entity and a set of keywords. An output is a list of generated slogans. The whole procedure is shown in Algorithm 1.

3.1 Extraction of the Keywords and the Main Entity

The most frequent non-negative words from the input text are selected as keywords. Negative words are detected using the Nodebox English Linguistics library [18]. The main entity is usually the name of the company and is obtained by selecting the most frequent entity in the whole text using the *nltk* library [4].

3.2 Generation of the Initial Population of Slogans

The procedure of generating the initial population of slogans is based on the BRAINSUP framework [19], with some modifications. It follows the steps in Algorithm 2. Skeletons are obtained from the database of slogan skeletons. Fillers are the words from the database of all grammatical relations between words in sentences that satisfy all predefined dependencies and POS tags. If there are any keywords in a set of all possible filler words, the algorithm assigns them higher priority for the selection phase. The main difference between our algorithm and the BRAINSUP method is in the selection of filler words. We don't consider any user specified constraints, while the BRAINSUP framework uses beam search in the space of all possible lexicalizations of a skeleton to promote the words with the highest likelihood of satisfying the user specifications. Thus using our method we can produce many different slogans from the same slogan skeleton, whereas BRAINSUP produces only one for given user specifications.

3.3 Evaluation of Slogans

An aggregated evaluation function is used to evaluate the slogans. It is composed of 9 different sub-functions, each assessing a particular feature of a slogan, with scores in the interval [0,1]. Parameter of the aggregation function is a list of 9 weights that sum to 1. They define the proportions of sub-functions in the overall score. In this subsection, we give a short description for every one of them.

3.3.1 Bigram Function

In order to work with 2-grams, we obtained the dataset of 1,000,000 most frequent 2-grams and 5,000 most frequent words in Corpus of Contemporary American English (COCA) [6]. We assume that slogans containing many frequent 2-grams, are more likely to be semantically coherent.

Algorithm 1: SloganGenerator

Input: A textual description of a company or a product T, Size of the population S_{P}, Maximum number of iterations $MaxIterations$, Crossover probability $p_{\mathrm{crossover}}$, Mutation probability p_{mutation}, Set of evaluation weights W.
Output: A set of generated slogans *S*.

$Keywords, Entity \Leftarrow$ GetKeywordsAndEntity(T)
$P \Leftarrow$ CreateInitialPopulation($S_{\mathrm{P}}, Keywords, Entity$)
Evaluate(P)
$Iteration \Leftarrow 0$
while $Iteration < MaxIterations$ **do**
  $Parents \Leftarrow$ ChooseParentsForReproduction(P)
  $Children \Leftarrow$ Crossover($Parents, p_{\mathrm{crossover}}$)
  $Children \Leftarrow$ Mutation($Children, p_{\mathrm{mutation}}$)
  $NewGeneration \Leftarrow$ DeleteSimilarSlogans($P, Children$)
  while Size($NewGeneration$) $< S_{\mathrm{P}}$ **do**
    AddRandomlyGeneratedSlogan($NewGeneration$)
  end while
  Evaluate($NewGeneration$)
  $P \Leftarrow S_{\mathrm{P}}$BestSlogans($NewGeneration$)
  $Iteration \Leftarrow Iteration + 1$
end while
$S \Leftarrow P$

Algorithm 2: CreateInitialPopulation

Input: Size of the population S_{P}, a set of target keywords K, and the target entity E.
Output: A set of initial slogans *S*.

$S \Leftarrow \emptyset$
while $S_{\mathrm{P}} > 0$ **do**
  $SloganSkeleton \Leftarrow$ SelectRandomSloganSkeleton()
  while not AllEmptySlotsFilled($SloganSkeleton$) **do**
    $EmptySlot \Leftarrow$ SelectEmptySlotInSkeleton($SloganSkeleton$)
    $Fillers \Leftarrow$ FindPossibleFillerWords($EmptySlot$)
    $FillerWord \Leftarrow$ SelectRandomFillerWord($Fillers$)
    FillEmptySlot($SloganSkeleton, FillerWord$)
  end while
  AddFilledSkeleton($S, SloganSkeleton$)
  $S_{\mathrm{P}} \Leftarrow S_{\mathrm{P}} - 1$
end while

3.3.2 Length Function

The length function is very strict, it assigns score 1 to slogans with less than eight words, and score 0 to longer ones. The threshold between 0 and 1 was set according to the results of the experiments, which showed that a large majority of the generated slogans that contained more than seven words were grammatically incorrect and semantically incoherent. Also, more than 90% of the famous slogans are less than eight words long. This function acts as an absolute constraint and that is why no values between 0 and 1 are allowed.

3.3.3 Diversity Function

The diversity function evaluates a slogan by counting the number of repeated words. The highest score goes to a slogan with no repeated words. If a slogan contains identical consecutive words, it receives score 0.

3.3.4 Entity Function

It returns 1, if slogan contains the main entity, and 0, if it doesn't.

3.3.5 Keywords Function

If one up to half of the words in a slogan belong to the set of keywords, the keywords function returns 1. If a slogan doesn't contain any keyword, the score is 0. If more than half of the words in the slogan are keywords, the score is 0.75.

3.3.6 Word Frequency Function

This function prefers slogans with many frequent words. A word is considered to be frequent, if it is among 5,000 most frequent words in COCA. The frequency score is obtained by dividing the number of frequent words by the number of all words in the slogan.

3.3.7 Polarity and Subjectivity Functions

Polarity of a slogan indicates whether slogan contains positive or negative words. For instance, the adjective “happy" is a positive word. In a similar way subjectivity of a slogan indicates whether slogan contains words that express the attitude of the author. For instance, the adjectives “good" and “bad" both represent the opinion of the author and are therefore subjective. The polarity and subjectivity scores are calculated based on the adjectives in the slogan, using the *sentiment* function from *pattern* package for Python [7].

3.3.8 Semantic Relatedness Function

This function computes the relatedness between all pairs of content words in a slogan. Stop words are not taken into account. Each pair of words gets a score based on the path distance between corresponding synsets (sets of synonyms) in WordNet [15]. The final score is the sum of all pairs' scores divided by the number of all pairs.

3.4 Production of a New Generation of Slogans

A list of all generated slogans is ordered descending with regard to the evaluation score. We use 10% elitism [8]. The other 90% of parent slogans are selected using a roulette wheel [11].

A new generation is built by pairing parents and performing the crossover function followed by the mutation function, which occur with probabilities $p_{\text{crossover}}$ and p_{mutation}, respectively. Offspring are then evaluated and compared to the parents, in order to remove very similar ones. If the number of the remaining slogans is smaller than the size of the population, some additional random slogans are generated using the method for creation of initial population. After that, slogans proceed into the next generation. These steps are repeated until the predefined number of iterations is achieved.

3.4.1 Crossover

We use two types of crossover functions, the *big* and the *small* one. Both inspect POS tags of the words in both parents, and build a set of possible crossover locations. Each element in the set is a pair of numbers. The first one provides a position of crossover in the first parent and the second one in the second parent. The corresponding words must have the same POS tag. Let the chosen random pair from the set be (p, r). Using the *big* crossover, the part of the first parent, from the p-th position forward, is switched with the part of the second parent, from the r-th position forward. For the *small* crossover only the p-th word in the first parent and the r-th word in the second parent are switched. Examples for the *big* and the *small* crossover are illustrated in Figure 2.

big:
We [PRP] bring [VBP] **good [JJ] things [NNS] to [DT] life [NN].**
Fly [VB] the [DT] **friendly [JJ] skies [NNS].**

→ We bring friendly skies.
Fly the good things to life.

small:
Just [RB] **do [VB]** it [PRP].
Drink [VB]more [JJR] milk [NN].

→ Just drink it.
Do more milk.

Figure 2: Examples for the *big* and the *small* crossover.[1]

3.4.2 Mutation

Two types of mutation are possible. Possible *big* mutations are: deletion of a random word; addition of an adjective in front of a noun word; addition of an adverb in front of a verb word; replacement of a random word with new random word with the same POS tag. *Small* mutations are replacements of a word with its synonym, antonym, meronym, holonym, hypernym or hyponym. A meronym is a word that denotes a constituent part or a member of something. The opposite of a meronym is a holonym – the name of the whole of which the meronym is a part. A hypernym is a general word that names a broad category that includes other words, and a hyponym is a subdivision of more general word.

Functions for obtaining such replacements are embedded into the Nodebox English Linguistics library and are based on the WordNet lexical database.

3.4.3 Deletion of Similar Slogans

Every generated slogan is compared to all its siblings and to all the evaluated slogans from the previous generation. If a child is identical to any other slogan, it gets removed. If more than half of child's words are in another slogan, the two slogans are considered similar. Their evaluation scores are being compared and the one with higher score remains in the population while the other one is removed. The child is also removed if it contains only one word or if it is longer than 10 words. Deletion of similar slogans prevents the generated slogans to converge to the initial ones. This has been checked by testing our method without the deletion of similar slogans phase.

[1] Slogans used in the examples were or still are official slogans of the following companies: General Electric, United Airlines, Nike, and BC Dairy Association.

3.5 Correction of Grammatical Errors

Crossover and mutation functions may cause grammatical errors in generated slogans. For instance, incorrect usage of determiners (e.g., “a apple" instead of “an apple"), sequence of incompatible words (e.g., “a the"), and others. Spelling mistakes were much less frequent.

In order to remove both types of errors in the final slogans, we tested different spell- and grammar checkers. One example of a spell-checker is Hunspell [12]. Its downside is that it works on one word at a time and does not take the word's context into account. As the majority of errors in slogans originated from grammar, we tested several grammar checkers. They, on the other hand, work on the sentence level rather than on the word level. Most of these grammar checkers are available as online services, and don't support API calls. One that does is python-ginger [10] – a Python package for fixing grammar mistakes. It comes with an unofficial Ginger [9] API. This tool corrects different types of grammatical mistakes. It is also used for contextual spelling correction. We used python-ginger only on final slogans, the ones that are displayed to the user, because the corrected slogan may not have the same structure anymore. Possible added words, or replacing a word with another one with different POS tag would cause errors while executing crossover, mutation and evaluation functions.

3.6 Unguided Slogan Generator

For the purpose of evaluation of our slogan generation method, we also implemented an unguided slogan generator (USG). This generator produces random slogans, such as the ones in the initial population. The only difference between our method and the unguided slogan generation method is in the production of a new generation. USG has no crossover and the mutation steps. Instead it produces a new generation using a method for creation of initial population. Thus children are independent of the previous generation. The algorithmic steps are shown in Algorithm 3.

4 Experiments

We tested the slogan generation method on different input texts and for different values of algorithm parameters. We analyzed the results of every iteration of the genetic algorithm to see how the slogans' scores changed and made further assessment of the generator by comparing its results with the results of the unguided slogan generator.

4.1 Experimental Setting

4.1.1 Input Text

In the presented experiments, we use a case of the Croatian provider of marine solutions, Sentinel. Sentinel is a control module that provides more security to boat owners, and is comprised of a network of sensors and a central information hub. It ensures the vessel is monitored at all times. The input text was obtained from the Sentinel's web-page [21].

4.1.2 Algorithm Parameters

Different combinations of weights of the evaluation function were tested on a set of manually evaluated slogans. We added one constraint – the weight of the keywords function had to be at least 0.2 in order to include keywords in the slogans. Without this constraint the computed weight for the keywords was almost zero. The comparison of the computed and the manually assigned scores showed that the highest matching was achieved with the following weights: [bigram: 0.25, length: 0.01, diversity: 0.01, entity: 0.1, keywords: 0.2, frequent words: 0.25, polarity: 0.01, subjectivity: 0.02, semantic relatedness: 0.15].

Probabilities for crossover and mutation were set to $p_{\text{crossover}} = 0.8$ and $p_{\text{mutation}} = 0.7$. The probability for mutation was set very high, because it affects only one word in a slogan. Consequently the mutated slogan is still very similar to the original one. Thus the high mutation probability does not prevent population from converging to the optimum solution. For the algorithm to decide which type of crossover to perform, we set probabilities for the *big*, the *small* and *both* crossovers to 0.4, 0.2 and 0.4, respectively. The mutation type is chosen similarly. Probabilities of the *big* and the *small* mutation were set to 0.8 and 0.2. These algorithm parameters were set according to the results of testing on a given input text, as their combination empirically leads to convergence.

We performed three experiments and for each of them we executed 20 runs of the algorithm using the same input parameter values. The difference between these three tests was in the size of the population (S_{P}) and the number of iterations (N_{It}). Those were chosen according to the desired number of all evaluations ($\approx 6,800$ NoE), and the NoE was set according to the desired execution time for one run of the algorithm – approximately 2 hours.

1. S_{P}: 25, N_{It}: 360
2. S_{P}: 50, N_{It}: 180
3. S_{P}: 75, N_{It}: 120

4.1.3 Comparison with the Unguided Slogan Generator

For comparison, we performed three experiments with the unguided slogan generator. For each of them we executed 20 runs of the algorithm using the same input parameter values as in the experiments with slogan generator. The initial populations were also identical. The number of iterations were again chosen so as to match the number of all evaluations ($\approx 6,800$ NoE) in the experiments with slogan generator:

1. S_{P}: 25, N_{It}: 300
2. S_{P}: 50, N_{It}: 150
3. S_{P}: 75, N_{It}: 100

Algorithm 3: UnguidedSloganGenerator

Input: A textual description of a company or a product T, Size of the population S_{P}, Maximum number of iterations $MaxIterations$, Set of evaluation weights W.
Output: A set of generated slogans S.

$Keywords, Entity \Leftarrow$ GetKeywordsAndEntity(T)
$P \Leftarrow$ CreateInitialPopulation($S_{\mathrm{P}}, Keywords, Entity$)
Evaluate(P)
$Iteration \Leftarrow 0$
while $Iteration < MaxIterations$ **do**
  $Children \Leftarrow$ CreateInitialPopulation($S_{\mathrm{P}}, Keywords, Entity$)
  $NewGeneration \Leftarrow$ DeleteSimilarSlogans($P, Children$)
  while Size($NewGeneration$) $< S_{\mathrm{P}}$ **do**
    AddRandomlyGeneratedSlogan($NewGeneration$)
  end while
  Evaluate($NewGeneration$)
  $P \Leftarrow S_{\mathrm{P}}$BestSlogans($NewGeneration$)
  $Iteration \Leftarrow Iteration + 1$
end while
$S \Leftarrow P$

In USG, children in new generations are frequently identical to parents, and therefore need no evaluation (we already have the scores of the parents). We wanted to compare the two generators based on the number of evaluations, not the number of iterations. For our slogan generator to reach the same number of evaluations as the unguided slogan generator, it needs to perform more iterations of genetic algorithm. That is why the numbers of iterations in SG and USG differ.

4.2 Results and Discussion

Comparing the statistical results of the initial and final populations of slogans, there were no major differences between the 20 runs of the algorithm on the same input data for all 6 experiments. The number of evaluations for each run is approximately $6,800$.

Statistics of average initial slogans' scores are in Table 1. The numbers are the same for both generators. Average final slogans' scores are in Table 2. The average minimum score is much higher using the unguided slogan generator (USG). This is because in our slogan generator (SG) many slogans get deleted in the deletion phase of the algorithm. Consequently some new random slogans are automatically included in a new generation, and they can have very low evaluation scores. However, SG has higher maximum slogan scores. This suggests that the usage of crossover and mutation functions actually increases the slogan scores. The average score of the 10 best slogans is higher using the SG.

Numbers in both tables show that average slogans' scores increased a lot from the initial population to the final one. Figures 3 and 4 show the relation between average slogan scores and the number of performed evaluations in a genetic algorithm using SG and USG. Using the USG causes the scores to increase immensely already in the first few iterations of the genetic algorithm. After that, they do not increase much anymore. In SG slogans' scores increase a little bit slower, but at some point they exceed the USG scores.

From the two graphs in Figures 3 and 4 one might conclude that the unguided slogan generator is at least as good as our developed slogan generation method. However, the numbers are calculated on slogans from a whole generation. In practice we don't expect the user to go through all 75 final slogans, but only a few. Thus only the best few slogans from the final list are important. Table 3 shows the average scores for the 10 best final slogans. In this case the slogan generator outperforms the unguided slogan generator.

In the following two lists, there are examples of slogans for one specific run of the algorithm. The first list contains 10 best-rated initial slogans and the second one contains 10 best-rated final slogans for the case when the size of the population was set to 50. Evaluation scores are in the brackets. The final slogans list contains the corrected versions of slogans using the Ginger API.

Initial Population:

1. Former offices for all its members houses. (0.692)
2. The lowest water to play Sentinel build. (0.664)
3. Land routes to better places. (0.663)
4. The Sentinel performance is topic. (0.662)
5. On day to perform. (0.642)
6. The side take in region. (0.639)
7. Even now right as not. (0.638)
8. A precise application consists with a pair. (0.632)
9. Draft the choice of allowing. (0.629)
10. The initiative in pursuing systems and weapons. (0.623)

Table 1: Comparison of average initial slogans' scores for population sizes 25, 50 and 75.

Size of the population	*Minimum*	*Maximum*	*Average*	*Median*	*Standard deviation*
25	0.000	0.713	0.287	0.359	0.257
50	0.000	0.740	0.289	0.302	0.257
75	0.000	0.730	0.274	0.295	0.251

Table 2: Comparison of average final slogans' scores using our slogan generator (SG) and the unguided slogan generator (USG) for population sizes 25, 50 and 75

Size of the population	*Minimum*	*Maximum*	*Average*	*Median*	*Standard deviation*
25 (SG)	0.578	0.906	0.801	0.823	0.088
50 (SG)	0.511	0.927	0.793	0.807	0.090
75 (SG)	0.488	0.939	0.773	0.791	0.094
25 (USG)	0.763	0.840	0.795	0.796	0.021
50 (USG)	0.723	0.837	0.767	0.761	0.032
75 (USG)	0.707	0.840	0.750	0.743	0.036

Table 3: Comparison of average scores of 10 best final slogans, using our slogan generator (SG) and the unguided slogan generator (USG) for population sizes 25, 50 and 75

Size of the population	*Minimum*	*Maximum*	*Average*	*Median*	*Standard deviation*
25 (SG)	0.833	0.906	0.869	0.870	0.023
50 (SG)	0.877	0.927	0.895	0.892	0.019
75 (SG)	0.871	0.939	0.902	0.902	0.023
25 (USG)	0.799	0.840	0.816	0.813	0.013
50 (USG)	0.804	0.837	0.819	0.813	0.011
75 (SG)	0.801	0.840	0.818	0.814	0.013

Final Slogans:

1. Enjoy the part like water for sentinel. (0.958)
2. Enjoy a take of routine on sentinel. → Enjoy a track of routine on sentinel. (0.958)
3. Make all safety in safe for sentinel. → Make all safety in safe for a sentinel. (0.958)
4. Demand and enjoy the use in sentinel. → Demand and enjoy the ease in sentinelthe sentinel. (0.958)
5. Write a base for demand on sentinel. (0.948)
6. Demand the of potential as sentinel. (0.945)
7. Enjoy a sentinel performance show. (0.922)
8. Themes for head on sentinel. (0.913)
9. Contents with application on sentinel. → Contents with application of sentinel. (0.913)
10. Make the sentinel performance plays. (0.897)

The analysis of initial populations and final slogans in all runs of experiments shows that the majority of slogans are semantically incoherent and have grammatical errors. However, slogans produced with the unguided slogan generator seemed more structured and semantically coherent. This is understandable, since the crossover and mutation functions in our slogan generator affect the sentence structure a lot. The percentage of corrected final slogans is also in favor of the unguided slogan generator: 24.6% of final slogans produced with USG got corrected with the Ginger API, while the percentage of corrected final slogans for SG is 33.9%. But we need to take into account the fact that Ginger API does not work without mistakes. Some of the corrections are strange or unnecessary (e.g., see example 4 in the final slogans list).

5 Conclusion

The proposed slogan generation method works and could be potentially useful for brainstorming. It produces slogans solely from the textual description of the target entity. No other user specifications are needed. The genetic algorithm ensures higher slogan scores with each new iteration. Our method outperforms the unguided slogan generator whose best 10 final slogans have significantly lower average scores. The unguided slogan generator also needs more than six times more time to produce and evaluate the same number of slogans as our slogan generator.

The evaluation function is inherently hard to formalize and seems not yet fully aligned with human evaluation. The definitions of evaluation sub-functions need further improvement in order to increase the quality of slogans, not only their scores.

The current algorithm is suitable only for production of slogans in English. The lack of resources and different language properties would require a lot of work in order to adapt our algorithm to another language.

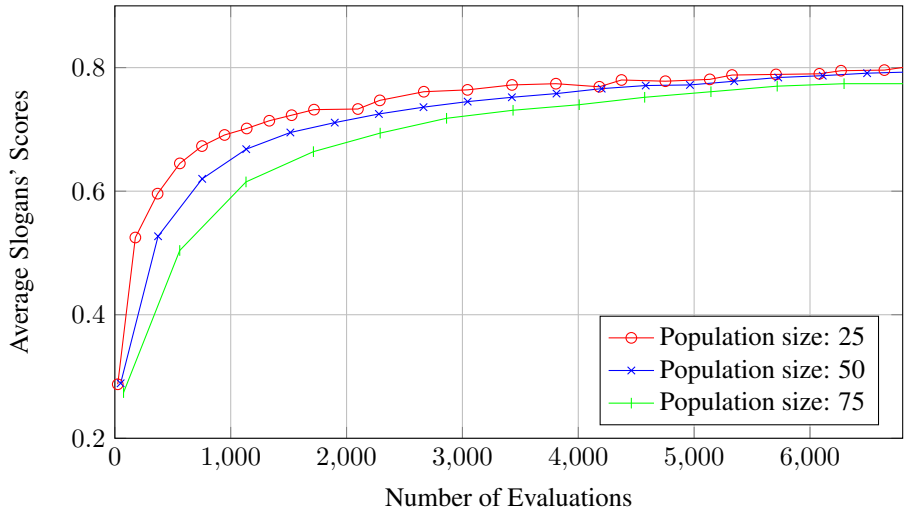


Figure 3: Slogan generator: average scores of slogans in a relation to the number of evaluations.

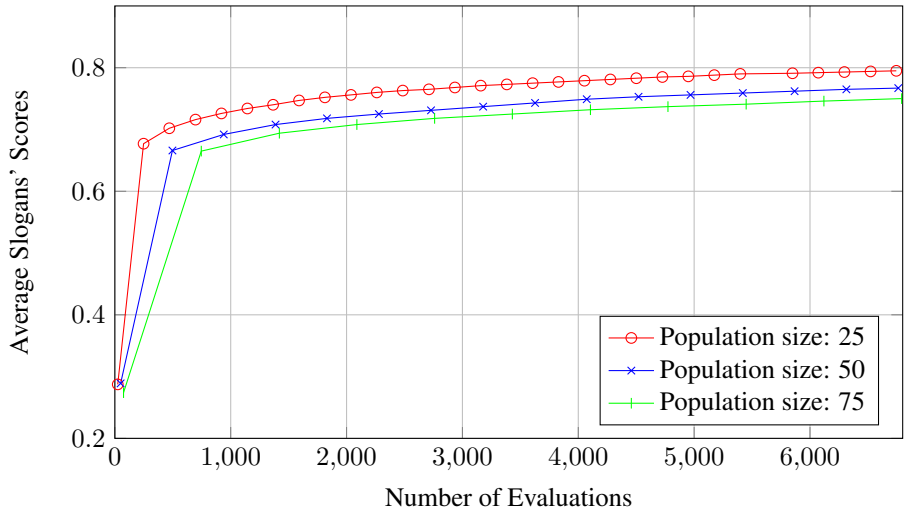


Figure 4: Unguided slogan generator: average scores of slogans in a relation to the number of evaluations.

Following are some ideas for the future work that would improve the quality of slogans. One is detecting and correcting grammatical errors already during the generation phase. New weights for the evaluation could be computed periodically with semi-supervised learning on manually assessed slogans. The parallelization of GA [1] might provide gains in performance. Also, the GA parameters could be adaptively calculated during the optimization process [20].

Acknowledgement

This research was partly funded by the European Union, European Social Found, in the framework of the Operational Programme for Human Resources Development, by the Slovene Research Agency and supported through EC funding for the project ConCreTe (grant number 611733) and project WHIM (grant number 611560) that acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission.

References

[1] E. Alba, J. M. Troya (1999) A survey of parallel distributed genetic algorithms, *Complexity*, vol. 4, pp. 31–52.

[2] L. Araujo (2009) How evolutionary algorithms are applied to statistical natural language processing, *Artificial Intelligence Review*, vol. 28, pp. 275–303.

[3] T. Bäck (1996) *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press.

[4] S. Bird, E. Klein, E. Loper (2009) *Natural language processing with Python*, O'Reilly Media.

[5] S. Colton, R. Mantaras, O. Stock (2009) Computational Creativity: Coming of age, *AI Magazine*, vol. 30, no. 3, pp. 11–14.

[6] M. Davies, N-grams data from the Corpus of Contemporary American English (COCA), `www.ngrams.info`, downloaded on April 15, 2014.

[7] T. De Smedt, W. Daelemans (2012) Pattern for Python, *Journal of Machine Learning Research*, vol. 13, pp. 2063–2067.

[8] D. Dumitrescu, B. Lazzerini, L. C. Jain, A. Dumitrescu (2000) *Evolutionary Computation*, CRC Press.

[9] Ginger, `www.gingersoftware.com/grammarcheck`, accessed on October 17, 2014.

[10] Ginger API, `github.com/zoncoen/python-ginger`, accessed on October 17, 2014.

[11] J. H. Holland (1992) *Adaption in Natural and Artificial Systems*, MIT Press.

[12] Hunspell, `hunspell.sourceforge.net`, accessed on October 20, 2014.

[13] R. Manurung, G. Ritchie, H. Thompson (2012) Using genetic algorithms to create meaningful poetic text, *Journal of Experimental & Theoretical Artifcial Intelligence*, vol. 24, pp. 43–64.

[14] M. Marneffe, B. MacCartney, C. Manning (2006) Generating typed dependency parses from phrase structure parses, *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pp. 449–454.

[15] G. A. Miller (1995) WordNet: A Lexical Database for English, *Communications of the ACM*, vol. 38, pp. 39–41.

[16] C. S. Montero, K. Araki (2006) Is it correct?: Towards web-based evaluation of automatic natural language phrase generation, *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL)*, pp. 5–8.

[17] R. G. Morris, S. H. Burton (2012) Soup over bean of pure joy: Culinary ruminations of an artifcial chef, *Proceedings of the International Conference on Computational Creativity (ICCC)*, pp. 119–125.

[18] NodeBox, `nodebox.net/code/index.php/Linguistics`, accessed on October 17, 2014.

[19] G. Özbal, D. Pighin, C. Strapparava (2013) BRAINSUP: Brainstorming Support for Creative Sentence Generation, *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pp. 1446–1455.

[20] G. Papa (2013) Parameter-less algorithm for evolutionary-based optimization, *Computational Optimization and Applications*, vol. 56, pp. 209–229.

[21] Sentinel, `sentinel.hr`, accessed on October 10, 2014.

Comparing Evolutionary Operators, Search Spaces, and Evolutionary Algorithms in the Construction of Facial Composites

Joseph James Mist, Stuart James Gibson and Christopher John Solomon
School of Physical Sciences, University of Kent, Canterbury, United Kingdom
E-mail: jm441@kent.ac.uk, s.j.gibson@kent.ac.uk, c.j.solomon@kent.ac.uk

Keywords: interactive evolutionary algorithm, facial composite

Received: December 1, 2014

Facial composite construction is one of the most successful applications of interactive evolutionary computation. In spite of this, previous work in the area of composite construction has not investigated the algorithm design options in detail. We address this issue with four experiments. In the first experiment a sorting task is used to identify the 12 most salient dimensions of a 30-dimensional search space. In the second experiment the performances of two mutation and two recombination operators for interactive genetic algorithms are compared. In the third experiment three search spaces are compared: a 30-dimensional search space, a mathematically reduced 12-dimensional search space, and a 12-dimensional search space formed from the 12 most salient dimensions. Finally, we compare the performances of an interactive genetic algorithm to interactive differential evolution. Our results show that the facial composite construction process is remarkably robust to the choice of evolutionary operator(s), the dimensionality of the search space, and the choice of interactive evolutionary algorithm. We attribute this to the imprecise nature of human face perception and differences between the participants in how they interact with the algorithms.

Povzetek: Kompozitna gradnja obrazov je ena izmed najbolj uspešnih aplikacij interaktivnega evolucijskega računanja. Kljub temu pa do zdaj na področju kompozitne gradnje niso bile podrobno raziskane možnosti snovanja algoritma. To vprašanje smo obravnavali s štirimi poskusi. V prvem je uporabljeno sortiranje za identifikacijo 12 najbolj izstopajočih dimenzij 30-dimenzionalnega preiskovalnega prostora. V drugem primerjamo učinkovitost dveh mutacij in dveh rekombinacijskih operaterjev za interaktivni genetski algoritem. V tretjem primerjamo tri preiskovalne prostore: 30-dimenzionalni, matematično reducirani 12-dimenzionalni in 12-dimenzionalni prostor sestavljen iz 12 najpomembnejših dimenzij. Na koncu smo primerjali uspešnost interaktivnega genetskega algoritma z interaktivno diferencialno evolucijo. Rezultati kažejo, da je proces kompozitne gradnje obrazov izredno robusten glede na izbiro evolucijskega operatorja(-ev), dimenzionalnost preiskovalnega prostora in izbiro interaktivnega evolucijskega algoritma. To pripisujemo nenatančni naravi percepcije in razlikam med interakcijami uporabnikov z algoritmom.

1 Introduction

Consider a situation in which a person witnesses a crime being committed by an unknown perpetrator. In the interests of identifying and subsequently locating the perpetrator, a facial image is often created from the witnesses' memory of the event. The traditional method is for the witness to select, from a database, individual facial features which a composite system operator then combines to form a likeness to the perpetrator called a *facial composite*. However, psychological research has shown that people generally recognise faces as whole objects (holistically) as opposed to recognising faces as collections of individual features [24, 6]. Also, people find it difficult to recall faces from memory and describe them whereas recognising an individual from a photograph of their face is a relatively easy task. Holistic methods for facial composite construction have been developed that account for these facets of human memory. EFIT-V [26] and EvoFIT [7] are commercial systems based on these principles that were developed in the early 2000s. EFIT-V is now used by over 75% of police constabularies in the UK and by many other law enforcement agencies in countries around the world.

The holistic method represents faces as points in a multidimensional search space. In our work, we refer to such a search space as a *face-space* due to its conceptual similarity to the notion of face-space in cognitive psychology research [25]. The key idea is to navigate from an initial starting point navigate to a unique region of face-space that corresponds to a facial likeness of the perpetrator.

The dimensions of face-space are determined by the principle components (PCs) of a training set of face images [5]. Each PC represents a unique holistic aspect of facial appearance and accounts for a proportion of the statistical image variance within the training set. The PCs are ordered by decreasing variance such that the first PC accounts for more variation than the second PC which accounts for more variation than the third PC etc. Faces not included in the training set, such as a perpetrator's face, may also be ap-

proximated by a weighted sum of the PCs.

To produce a likeness of a perpetrator, some process for searching the face-space is required. A simple approach is to use a bank of sliders in which each slider corresponds to a single PC. This method has been used in a workable composite system [3] but has two drawbacks: it is unlikely that any one slider will produce a change in facial appearance that maps to a simple semantic description (e.g. thin face) and the number of permutations of for the bank of sliders becomes cognitively prohibitive even for a relatively small number of PCs.

An alternative, less demanding, method for locating a face in a face-space is to use an iterative process whereby generated faces are assessed by the witness according to their similarity to the perpetrator. This method is implemented in EFIT-V and EvoFIT using *interactive evolutionary algorithms* (IEAs). In IEAs the fitness function evaluation, standard in evolutionary algorithms (EAs), is replaced by subjective human evaluation. IEAs are suitable for tasks requiring human assessment of solutions in which input values are difficult to optimise individually because of interaction between input values and because of the noisy and imprecise nature of human interaction. Takagi [23] provides many examples of tasks that IEAs have been applied to, including the fitting of hearing aids, graphic art, and industrial design.

Genetic algorithms (GAs) were introduced by Holland in 1973 [12]. GAs can be used to solve problems requiring binary, integer, and real valued inputs and are easy to implement. For these reasons, interactive genetic algorithms (IGAs) are a popular choice of IEA. IGAs were used in the implementation of EFIT-V and EvoFIT and have also been applied to tasks such as image filtering [15] and product design [4].

The use of human evaluation places limitations on an IEA which are not usually present in an EA. Fatigue will limit the number of individuals (faces) a user is willing to evaluate. Fatigue also limits the granularity of the scale upon which individuals can be rated. For example, a scale of 1–100 is overly burdensome whereas a simple "good" or "not good" decision is less so [28]. It is a demanding task for users to assign absolute fitness scores to individuals, which limits the number of individuals that a user can be expected to evaluate. An alternative approach that enables users to evaluate more individuals, albeit generally less thoroughly, is to allow the user to compare individuals to each other. For example, individual "A" could be better than, as good as, or worse than individual "B". The latter approach to evaluation is used in the IEAs implemented for comparison in this work.

When using an EA to solve a problem, care is taken to choose an appropriate algorithm, operators, and parameter values. In most cases it is feasible to perform many runs, comparing different algorithm design options and parameter values to see which yield the best result. Such comparisons are prohibitively difficult when working with IEAs because of the limitations placed by human evaluation.

In an effort to make these comparisons, mathematical models of human evaluation, which we refer to here as *virtual users*, have been used in place of human participants when optimising aspects of IEAs. These virtual users are effectively EAs implemented with limitations that model those imposed by human evaluation. Virtual users were used in the early development of EFIT-V and EvoFIT to choose effective IGAs, set population sizes, mutation rates, and selection pressures [19, 11, 8, 9].

It is difficult to judge the usefulness of the virtual user approach as there is virtually no work evaluating design decisions at the parameter/operator level of algorithm design that use human participants. An experiment conducted by Breukelaar et al. [2] used a colour matching task to compare the use of three fixed step size and one variable step size mutation parameters in an interactive evolution strategy. The work concluded that using variable step size enabled colour matches to be achieved quicker than using fixed step sizes. Oinuma et al. [18] compared four recombination operators in a face beautification task and concluded that a novel recombination method introduced in the paper performed better than existing recombination methods. These results were not confirmed using statistical analysis and therefore it is not known whether the observed differences were due to genuine differences between the operators or if they were due noise in the data gathered. More robust testing of design decisions using human participants is required to gauge whether the comparison of parameter values and operators is useful or whether differences between users generally renders any differences between the design options irrelevant.

EFIT-V uses a face-space model determined by 60 PCs [21] whereas the number of PCs used in EvoFIT is harder to discern but [9] and [10] imply that the maximum possible number of PCs is used. The question of the optimal number of PCs to use does not appear to have been addressed since the earliest work in the development of EFIT-V and EvoFIT. The imperfect nature of human face recognition implies that the number of dimensions used in holistic facial composite systems could be reduced significantly without any perceived loss in image accuracy. If the number of PCs to be used is reduced then the most obvious PCs to retain are those which account for the most statistical variation in the training set. These PCs may not necessarily, however, be those that account for the most perceptual variation. In this paper we ask if human evaluation should play a role in selecting those PCs that are used to create a face-space of reduced dimensionality.

It is reasonable to expect that the difference between algorithms is more significant than the difference between operators. Differential evolution (DE) is a relatively recent metaheuristic algorithm having been introduced by Storn and Price in 1997 [22]. Examples of applications for interactive differential evolution (IDE) include forensic image segmentation [17] and optimising optical illusions [16].

Work on comparing IEAs is as scant as that for comparing operators and parameter settings. Kurt et al. [13] compared a number of biologically inspired metaheuristic

algorithms, including IDE and IGAs, for facial composite construction. It was found that IDE required fewer evaluations create a composite but the recognition rate of the IDE composites was lower than for the other algorithms. Lee and Cho [14] compared an IDE algorithm to an IGA and to a direct input manipulation method for an image enhancement task and found that participants generally favoured the IDE algorithm for usability. In neither of these experiments was a statistical comparison between the algorithms undertaken and so it is unknown whether these results are reliable.

In this work we construct a 12-dimensional "human reduced" face-space using human evaluation of the differences between pairs of faces drawn from a larger 30-dimensional face-space. We then compare two mutation operators and two recombination operators in an IGA using a task in which participants create facial composites from memory. In the third experiment the performance of searches using the human-reduced face-space, developed in the first experiment, is compared to that of the larger 30-dimensional face-space and a "mathematically reduced" 12-dimensional face-space using the same facial composite task. In the final experiment, we compare an IGA to an IDE algorithm.

2 Theory

2.1 Face-space model

A face-space model was constructed that captures the natural variation of shape and texture (the shading and colour) of human faces. The training set of photographs used to build our face-space model consists of 27 male and 63 female faces of various ages. The model building process starts with manually placing 190 land mark points on each photograph to delineate the key facial features at, for example, at the corners of the eyes, the bottom of the chin, and the outline of the eyebrows. The face shape of each subject in the training set is hence defined by a 380 dimensional vector containing the x-y coordinates of 190 land mark points.

The face shapes are aligned, using the Procrustes method, and the mean face shape $\bar{\mathbf{s}}$ calculated. Principal components analysis (PCA) is used to reduce the 380-dimensional shape model to a smaller number of dimensions. Any face shape $\mathbf{s}$ can then be approximated as $\widehat{\mathbf{s}}$ by the shape model using

$$\widehat{\mathbf{s}} = \mathbf{P}_s \mathbf{b}_s + \bar{\mathbf{s}} \tag{1}$$

where $\mathbf{P}_s$ are the PCs of the shape model ordered from most important (the PCs which account for the most variance in the data) to least important and $\mathbf{b}_s$ is a vector of parameters that determine how the shape PCs are combined to make the face shape.

In order to create the texture model that encodes the image pixel values, each photograph in the training set is partitioned using its land mark points and Delaunay triangulation. Piecewise affine transforms are used to warp each training image to the mean face shape thereby forming shape normalised texture patterns. PCA is then used to find a texture model of much fewer dimensions than the original pixel space of the normalised texture patterns. As with the face shapes, any face texture $\mathbf{g}$ may be approximated using

$$\widehat{\mathbf{g}} = \mathbf{P}_g \mathbf{b}_g + \bar{\mathbf{g}}. \tag{2}$$

where $\mathbf{P}_g$ are the PCs of the face texture ordered from the most important to least important and $\mathbf{b}_s$ are parameters that determine how the texture PCs are combined to make the face texture. Finally, a face-space model is created from the combined shape and texture models using PCA to further reduce the number of dimensions. Thus, the appearance model parameters, $\mathbf{c}$, of any face can be approximated as $\widehat{\mathbf{c}}$ using

$$\widehat{\mathbf{c}} = \mathbf{Q}^T \begin{bmatrix} w\mathbf{b}_s \\ \mathbf{b}_g \end{bmatrix} \equiv \mathbf{Q}^T \begin{bmatrix} w\mathbf{P}_s^T(\widehat{\mathbf{s}} - \bar{\mathbf{s}}) \\ \mathbf{P}_g^T(\widehat{\mathbf{g}} - \bar{\mathbf{g}}) \end{bmatrix} \tag{3}$$

where $\mathbf{Q}$ are the appearance PCs of the training set ordered from the most important to the least important and w scales the shape parameters such that equal significance is assigned to shape and texture.

New faces can be generated by setting the values of an n-dimensional parameter vector $\mathbf{c}$ and performing the above process in reverse. Starting with the extraction of $\mathbf{b}$

$$\mathbf{b} = \sum_{i=1}^{n} \mathbf{q_i} c_i \tag{4}$$

where $\mathbf{q}_i$ is the i-th column of matrix $\mathbf{Q}$ in Equation 3. The shape and texture parameters $\mathbf{b}_s$ and $\mathbf{b}_g$ are extracted from $\mathbf{b}$ and are used in Equations 1 and 2 to find the shape parameters $\mathbf{s}$ and texture parameters $\mathbf{g}$. The pixel intensities in $\mathbf{g}$ are rearranged into a two-dimensional (or three-dimensional for colour images) array of pixels which then form an intermediate face image with mean face shape. Aspects of the edge of the face image which are due to the land marking process have a dominant unwarranted effect on the perception of the face. To counter this effect the generated face texture is inserted and blended into a softened background. The resulting image is subsequently warped according to the shape parameters, $\mathbf{s}$, to form the final face image.

It is important to note that there are many features which cannot be reproduced using this method. Apart from obvious highly distinctive features such as birthmarks and scars, more mundane high frequency features such as beards and hair cannot be effectively rendered. In commercial software these features are added separately using overlays and drawing packages.

2.2 The interactive algorithms used

The IEAs used in this work both used the same representation for the genotypes: n-dimensional real valued vectors

where n is the number of dimensions of the face-space being used.

A larger population requires more processing time to generate the composites and imposes a greater cognitive burden on the user whereas a smaller population size means that a greater number of generations is required to achieve a satisfactory composite. EvoFIT uses a population size of 18, EFIT-V uses uses a population size of 9. We used a population size of 9 for both the IGA and IDE because this number of images could be displayed at a reasonable scale and also limits the cognitive demands placed on the user when comparing faces.

The IGA used in this work is very similar to that developed by Frowd [8]. Only three levels of fitness evaluation are allowed: preferred (best), selected, and not selected. Every generation exactly one individual is chosen as the preferred individual. This individual is carried unaltered into the following generation. Eight new individuals are needed to populate each generation. Each new individual has two parents and so a mating pool of sixteen individuals is required.

Stochastic universal sampling (SUS) [1] is used to select the parents to go into the mating pool. In SUS a "wheel" bearing a superficial similarity to a roulette wheel, is constructed based on the fitness values of individuals in the previous generation. In the IGA used in this work, each selected individual is assigned an equal sized section of the wheel except for the preferred individual which is assigned a double sized wedge. To select the parents, a "spinner" comprising sixteen equally spaced arms is spun and for every arm that "comes to rest" on a particular section the individual corresponding to that section is added once to the parent pool.

Once the parent pool is filled, individuals are drawn from the pool in pairs to undergo recombination to form new individuals. Uniform crossover and arithmetic crossover recombination operators are used in our experiments. In our implementation of uniform crossover there is equal chance that the offspring will inherit each gene from either parent. In our implementation of arithmetic crossover the value of each gene in an offspring is the mean of the values for that gene in the parents.

After a new individual is created using recombination it undergoes mutation. We used Gaussian addition and Gaussian replacement mutation operators in our experiments. In Gaussian addition, the mutated gene value c_i' is given by

$$c_i' = c_i + \sigma_i \cdot m \cdot r_i \tag{5}$$

where σ_i is the standard deviation (SD) of the data on the i-th PC, m is the mutation factor set by the user on the interface, and r_i is a random number from the Gaussian distribution $N(0,1)$. Gaussian replacement is the name given in this paper to an analogous method to the uniform mutation operator. In uniform mutation, each gene c_i in an offspring's genotype will be replaced, with probability p_m, by a uniformly distributed random value c_i' such that $c_i' \in [\text{Lower limit}, \text{Upper limit}]$. The Gaussian replacement operator is similar except that c_i' is a random number taken from $N(0,1)$ and multiplied by the SD of the data on the i-th PC. c_i' has the further restriction that it is bounded by a hyperrectangle which designates the edge of the face-space, that is $c_i' \in [-2.5, 2.5]$ SDs. This was done to reduce the likelihood of implausible faces or faces exhibiting image artefacts. The mutation probability is set by the mutation slider and is restricted to the range $[0, p_{\max}]$ where $p_{\max} = 5/$ (the dimensionality of the face-space).

The IDE algorithm used is an adaptation of basic DE as presented by Price et al. [20]. In DE each member of the population is the main parent of exactly one offspring. This main parent is the *target vector* and the offspring is known as the *trial vector*. Three other parents are used to generate each trial vector; the *base vector* and two *difference vectors*. Once the trial vectors have been generated each is compared to its target vector. If the trial vector is found to be fitter than its target vector then the trial vector takes the place of target vector in the population.

The first step in creating a trial vector is to create a *mutant vector* according to

$$\mathbf{x}_{mutant} = \mathbf{x}_{base} + F(\mathbf{x}_{diff1} - \mathbf{x}_{diff2}) \tag{6}$$

where $\mathbf{x}_{base}$ is the base vector, $\mathbf{x}_{diff1}$ and $\mathbf{x}_{diff2}$ are the difference vectors, and F is the mutation scale factor which is usually constrained to the range $(0,1)$. The second step is to cross the mutant vector with the target vector to create the trial vector according to

$$x_{i,trial} = \begin{cases} x_{i,mutant} & \text{if } r_i < Cr \\ x_{i,target} & \text{otherwise} \end{cases} \tag{7}$$

where Cr is the crossover probability and r_i is a random number drawn from a uniform distribution in the range $(0,1)$. To ensure that $\mathbf{x}_{trial} \neq \mathbf{x}_{target}$, if $\mathbf{x}_{trial} = \mathbf{x}_{target}$ one random position i in $\mathbf{x}_{trial}$ would be set such that $x_{i,trial} = x_{i,mutant}$. A virtual user was used to find optimal values of F and Cr for the IDE implemented in this work, as some values of F and Cr can lead to, for example, premature convergence. The optimal values were found to be $F = 0.6$ and $Cr = 0.5$. Preliminary testing with human evaluation confirmed that these values were suitable.

The target, base, and difference vectors were chosen to be different members of the population. Each vector was used as the base vector exactly once per generation. The order for the base vectors was determined using the random permutation method. The difference vectors for each trial vector were chosen at random from the population excluding the trial vector's target and base vectors.

3 Software for Experiments 2, 3, and 4

We developed software using Matlab that generates faces from our face-space model using input values determined using IEAs. The IEAs were designed and built specifically for this work.

Figure 1: Screenshot of the interface for the IGA

A screenshot of the interface developed for the IGA is given in Figure 1. For every generation the participant would choose, using the left mouse button, exactly one preferred composite face that best resembled the target face they were trying to recreate. Additionally, if the participant thought that any of the other faces were a good likeness, they had the option of selecting these using the right mouse button. Anywhere from zero to eight faces could be selected in this way. A green border was placed around the face the participant preferred, a yellow border for those faces the participant thought were also good, and a black border for those faces that were not selected. Once they were satisfied that they had selected the best match, and any other matches they considered to be good, the participant would go to the next generation by pressing the 'Next' button. The participant would repeat the process until they thought no further improvement was possible, at which point they would click on the 'Finish' button.

A mutation slider was included so that participants could adjust the value of the mutation parameter. For the experiments reported in this paper, the mutation slider was decremented by 0.03 per generation by the software (the slider's range was $[0, 1]$). A 'Back' button was included which enabled the participant to go back to the previous generation and make alternative selections or adjust the mutation slider if they were not satisfied with the current generation. This design decision was based on comments from participants in earlier experiments who expressed a desire for such functionality when the population as a whole was worse than that of the previous generation.

Screenshots of the interface developed for the IDE algorithm are given in Figures 2 and 3. In every generation the participant would look for a satisfactory match to the target face within the population. If a satisfactory face was apparent the participant could select it and click the 'Finish' button. If no such face was apparent they would click the 'Next' button to generate the trial vectors and their corresponding faces (Figure 2). The faces generated from the trial vectors would be compared to those generated from their target vectors on a pairwise basis (Figure 3). From each pair of faces, the participant was asked to click on the face which most closely resembled the target and then click on the 'OK' button. Once the participant had completed the nine pairwise comparisons the new population of individuals was presented to them. At this stage the participant could continue or finish. The participant also had the option of redoing the pairwise comparisons if they thought that the current population was generally worse than that of the previous generation by pressing the 'Redo' button.

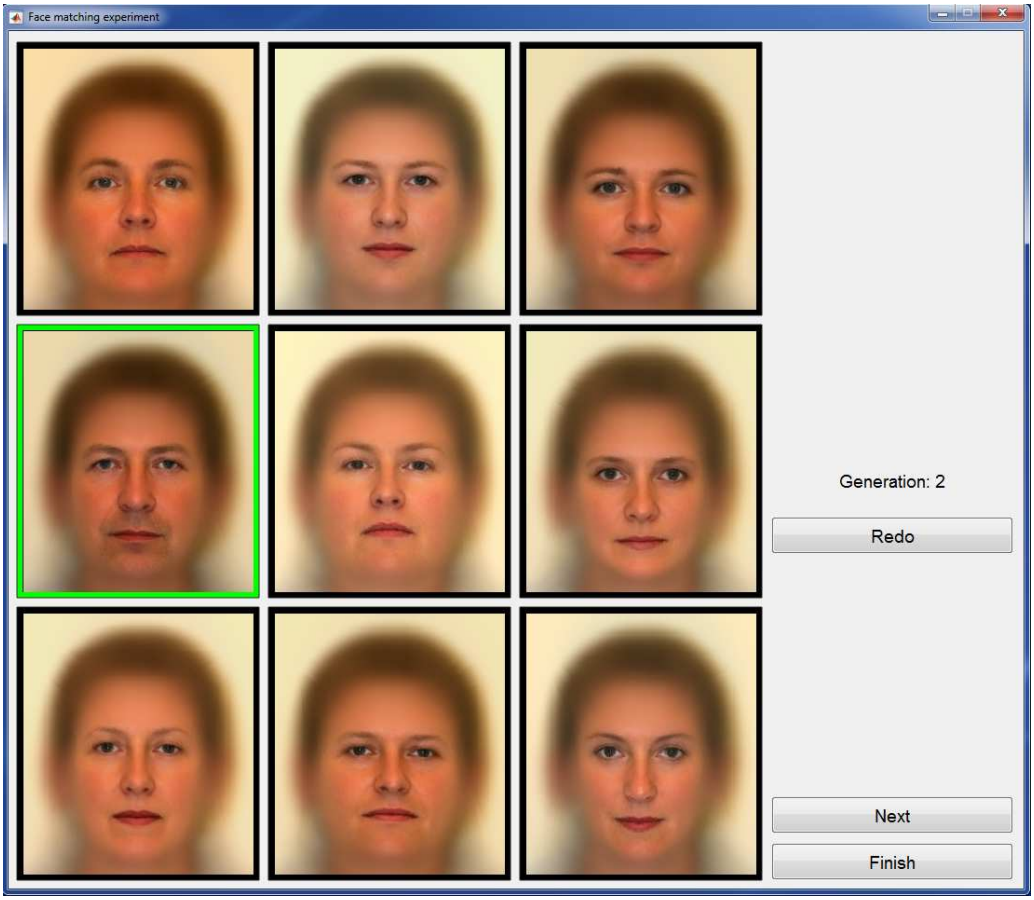


Figure 2: Screenshot of the main interface for the IDE algorithm

Figure 3: Screenshot of the pairwise selection interface for the IDE algorithm

(a) Faces generated at ±3 SDs on the 1-st PC

(b) Faces generated at ±3 SDs on the 30-th PC

Figure 4: The pairs of faces at ±3 SDs on the 1-st and 30-th PCs

4 Experiment 1: Identifying the most perceptually significant PCs

4.1 Method

In the first experiment 32 participants performed a face sorting task to determine which 12 of the first 30 PCs, derived using PCA, are perceptually most significant. Accordingly, thirty pairs of faces were generated from the first 30 PCs. Each pair of faces was constructed from points at ±3 SDs along one of the PCs. If we form a 'large' 30-dimensional face-space in which a face's representation is given by $\mathbf{c} = [c_1, c_2, \ldots, c_i, \ldots, c_{30}]$ then each pair of points $(\mathbf{c}_{+k}, \mathbf{c}_{-k})$ representing a pair of faces has the face-space coordinates

$$c_{\pm i} = \begin{cases} \pm 3 \quad \text{SDs} & \text{if } i = k \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The pairs of faces from the 1-st and 30-th PCs are shown in Figure 4

The faces were printed in their respective pairs on matt photographic paper. Each pair was 5.8 cm high by 10.2 cm wide. There are three reasons why the task was limited to 30 pairs of faces: 30 pairs of faces fit comfortably on a desk's surface, the differences between each pair of faces becomes smaller for higher order PCs, and the difficulty of the task increases with the number of pairs.

At the start of the experiment the pairs of faces were arranged randomly in a grid six pairs high by five pairs wide. The participants were instructed to group the 12 pairs of faces which "exhibited the most within pair dissimilarity". Once the participants had done this they were instructed to sort the 12 pairs of faces from the most similar to the least similar. In preliminary testing, it was observed that the degree of dissimilarity between pairs of faces became very hard to discern beyond the 12 most dissimilar pairs. Consequently, 12 dimensions were used for the human reduced face-space.

4.2 Results

A pair of faces was awarded 12 marks when judged to be the most dissimilar by a participant. Similarly, the second most dissimilar pair was awarded 11 marks, the third 10 marks and so on until the 12 most dissimilar face pairs had been accounted for. The marks were summed over all of the participants to obtain the aggregated rank order of face pairs and hence the perceptual ordering of PCs. The 12 most perceptually significant PCs were found, in order, to be 1, 2, 3, 5, 15, 7, 4, 14, 13, 6, 18, and 9. These are the PCs that were used to build the human reduced face-space. It can be seen that 8 of the 12 PCs in the human reduced face-space are in the first 12 PCs of the larger 30-dimensional face-space.

5 Experiment 2: Comparison of recombination and mutation operators

5.1 Method

In this experiment 15 participants were used to compare two recombination operators (uniform crossover and arithmetic crossover) and two mutation operators (Gaussian replacement and Gaussian addition).

The 12-dimensional human reduced face-space was used in this experiment. This face-space was chosen because it was thought that a face-space constructed using fewer dimensions may lead to a face match more quickly than one constructed using many dimensions and thus induce less fatigue in the participants. It was not thought that choice of face-space would affect the relative performances of the different recombination and mutation operators. Testing each combination of recombination and mutation operator required $2 \times 2 = 4$ runs per participant. Each participant also did a practice run at the start of the experiment in order to gain familiarity with the task and the interface. The initial population was the same for every run of the experiment and was designed to be roughly evenly distributed across the human reduced face-space. In an attempt to achieve this, K-means clustering was used. To generate the initial population, 1000 points were generated using a 12-dimensional uniform distribution with the limits

being at ± 2.5 SDs on each axis. The points were grouped into nine clusters using K-means clustering via Matlab's *kmeans* function. The centroids of the nine clusters were used as the genotypes for the initial population of faces.

At the start of each run the participants were given 10 seconds to study the target face which they then tried to recreate from memory using the IGA facial composite process. The target face was not shown to the participants again until the end of the run. The target faces were chosen to be equidistant from the centre of the human reduced face-space. At the end of every run, participants were shown the composite they had just created and were asked to rate its similarity to the target on a scale from 1 to 10. Composites were then displayed side-by-side with their corresponding target faces and in each case the participant provided an additional similarity score. The purpose of the without target comparison was to gauge how well the composites matched the faces held in the minds of the participants; in reality witnesses would not have an image of the perpetrator to compare their composites to. The with target comparison was included as a slightly less subjective measure of how good the composites were.

Three sets of objective data were gathered: the time taken to create the composites, the number of generations it took to create the composites, and the number of times the Back button was used. The time taken, and the number of generations, were used as indicators of how quickly the participants were able to attain face matches. The use of the 'Back' button was recorded to provide an indication of how often the searches were producing a generation that was worse than the proceeding one.

5.2 Results

Table 1 comprises the means and standard deviations of the following measured variables: number of generations, time taken, number of times the Back button was used, participant rating of their composite without reference to the target, and participant rating of the their composite with reference to the target. Each of the measured variables were subjected to aligned rank transform (ART) with two-way ANOVA [27]; having two mutation operators (Gaussian addition and Gaussian replacement) and two recombination operators (uniform crossover and arithmetic crossover) (Table 2). The differences between the mutation operators, and the differences between the recombination operators, were not significant for any of the measured variables. The interaction between the operators, that is the effect of using any particular mutation/recombination operator pair, was not significant.

6 Experiment 3: Comparison of Face-Spaces

6.1 Method

In this experiment 21 participants were used to compare three face-spaces: a face-space constructed from the first 30 PCs of the PCA analysis (the large face-space), a face-space constructed from the first 12 PCs (the mathematically reduced face-space), and a face-space constructed form the 12 most perceptually important PCs identified in the first experiment (the human reduced face-space).

As the results of the second experiment showed no significant difference between the operators on any of the recorded measures, arithmetic crossover and Gaussian addition were arbitrarily chosen as the operators for this experiment.

As there were only three test conditions (large face-space, human reduced face-space, and mathematically reduced face-space) each participant performed two runs for each condition, equal to $2 \times 3 = 6$ runs in total. Each participant also performed an additional practice run at the start of the experiment.

The initial populations for each of the face-spaces were generated using the same method as that used in Experiment 2. The target faces were chosen to be equidistant from the centre of the 30-dimensional face-space. They were also chosen such that they could not be represented exactly in the two 12-dimensional face-spaces. This was done to model the error in reconstruction associated with using a low-dimensional face-space.

6.2 Results

The measured variables were the same as those for Experiment 2. The means and standard deviations of the measured variables for each of the face-spaces are presented in Table 3.

Performing Friedman's test on each of the measured variables showed that the differences between the face-spaces were not significant for any of the measured variables (number of generations: $\chi^2(2) = 2.11, p = 0.349$, number of times the 'Back' button was used: $\chi^2(2) = 0.54, p = 0.765$, time taken: $\chi^2(2) = 2.14, p = 0.343$, without comparison rating: $\chi^2(2) = 2.37, p = 0.306$, and with comparison rating: $\chi^2(2) = 0.71, p = 0.700$).

7 Experiment 4: Comparison of IGA and IDE

7.1 Method

In this experiment 22 participants were used to compare an IGA to an IDE algorithm.

As the results of the second experiment showed no significant difference between the operators on any of the recorded measures, arithmetic crossover and Gaussian addition were arbitrarily chosen as the operators used for this experiment. As the results of the third experiment showed no significant difference between the face-spaces, the human reduced face-space was used in this experiment.

Table 1: Means (standard deviations) of the dependent variables in the comparison of mutation and recombination operators in the creation of facial composites

Mutation Recombination	Gauss. replacement uniform	Gauss. replacement arithmetic	Gauss. addition uniform	Gauss. addition arithmetic
Generations	10.6 (5.10)	12.5 (8.64)	11.5 (4.73)	9.73 (2.49)
Back count	0.73 (1.33)	0.47 (0.74)	0.87 (1.41)	0.47 (0.64)
Time taken	195s (91.5s)	222s (155s)	220s (71.1s)	188s (66.2s)
Without rating	6.27 (1.22)	5.47 (2.00)	6.07 (1.03)	6.07 (1.49)
With rating	4.40 (2.10)	5.07 (2.19)	4.60 (2.41)	4.40 (2.32)

Table 2: ART with two-way ANOVA of the dependent variables in the comparison of mutation and recombination operators in the creation of facial composites

	Mutation		Recombination		Interaction	
Variable	$F(1,56)$	p-value	$F(1,56)$	p-value	$F(1,56)$	p-value
Generations	0.025	0.874	0.041	0.840	0.826	0.367
Back count	0.153	0.670	0.368	0.547	0.055	0.816
Time taken	0.427	0.516	0.553	0.460	0.851	0.360
Without comparison rating	0.132	0.718	0.510	0.478	0.771	0.384
With comparison rating	0.425	0.517	0.214	0.645	0.571	0.529

Table 3: Means (standard deviations) of the dependent variables in the comparison of the large, human reduced and mathematically reduced face-spaces in the creation of facial composites

Face-space	Generations	Back count	Time taken	Without target rating	With target rating
Large	10.7 (4.73)	0.50 (0.55)	205s (80.3s)	5.81 (1.13)	4.10 (1.25)
Human reduced	9.38 (4.31)	0.36 (0.42)	186s (91.8s)	6.02 (1.08)	3.95 (1.33)
Mathematically reduced	10.5 (4.75)	0.48 (0.56)	193s (85.6s)	5.86 (1.16)	4.12 (1.82)

There were two test conditions (IGA and IDE) hence we had each participant perform two runs using each condition, equal to $2 \times 2 = 4$ runs in total. Each participant also performed two practice runs at the start of the experiment, one for each of the IEAs.

The initial populations were generated using the same method as that used in Experiment 2. The target faces were chosen to be equidistant from the centre of the human reduced face-space.

7.2 Results

The measured variables were the same as those for Experiments 2 and 3 but the use of the IGA's "Back" button was compared to the use of the IDE's "Redo" button. The means and standard deviations of the measured variables for each of the algorithms are presented in Table 4.

Performing exact calculations for Wilcoxon's signed-rank test on the measured variables showed that the differences between the face-spaces were not significant for any of the measured variables (number of generations: $p = 0.571$, number of times the "Back"/"Redo" button was used: $p = 0.625$, time taken: $p = 0.305$, without comparison rating: $p = 0.553$, and with comparison rating: $p = 0.520$).

The participants were also asked which of the two IEAs they preferred as it was possible to differentiate between the IEAs because of the difference between the interfaces. The IGA was preferred by 6 of the 22 participants, 14 preferred IDE and 2 stated no preference. Performing exact calculations for the sign test showed that this difference was not significant: $p = 0.115$. Those who preferred IDE often stated that they found it easier to compare two faces at a time than nine, which they found made the composite process easier.

8 Conclusion

A human reduced face-space for use with an IEA in the creation of facial composites was derived from a higher dimensional PCA based face-space. The performances of searches for faces in the human reduced face-space were compared to those of a mathematically reduced face-space and to the larger face-space. Searches performed using an IGA with two different mutation operators and two different recombination operators were compared. Searches performed using the IGA were compared to those performed using IDE.

The prioritisation of the PCs with regards to human evaluation was found to be similar to the numerical ordering returned by PCA itself. The human reduced face-space was found to share 8 of its 12 dimensions with the mathematically reduced face-space. We note that our data set comprised images captured under conditions of controlled pose, lighting and facial expression. If this were not the case, one might expect greater differences between the perceptual and numerical orderings of PCs. This is because users can filter out variability due to lighting, pose, and camera angle; something that selecting the most significant PCs mathematically does not account for.

No significant differences in the performances of the searches conducted using the different operators were detected, nor were any significant differences found between the performances of the IEAs. The difficulty and uncertain nature of creating a facial composite render any difference in the performances of the operators or the IEAs insignificant. This observation calls into question the utility of using virtual users or even testing with human users to aid in making algorithmic design decisions; and lends strength to the idea that it is safe to make these decisions based on the judgement of the people implementing an IEA. Our work also brings into doubt the validity of conclusions in prior work based on experiments with virtual users or where statistical analysis has been omitted.

No significant differences in the performances of the searches conducted in the different face-spaces was observed. Again this is likely to be due to the imperfect nature of face recall and recognition. This result implies that it is possible to reduce the dimensionality of the face-space without any loss of performance. It also shows that using the mathematical ordering of the PCs is acceptable when truncating the face-space and it is unlikely to be necessary to make allowances for human perception.

References

[1] J. E. Baker (1987) Reducing bias and inefficiency in the selection algorithms, *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 14–21.

[2] R. Breukelaar, M. Emmerich, T. Bäck (2006) On Interactive Evolution Strategies, *Lecture Notes in Computer Science*, vol. 3907, pp. 530–541.

[3] R. Brunelli, O. Mich (1996) SpotIt! an interactive identikit system, *Graphical Models and Image Processing*, vol. 58, no. 5, pp. 399–404.

[4] F. Cluzel, B. Yannou, M. Dihlmann (2012) Using evolutionary design to interactively sketch car silhouettes and stimulate designer's creativity, *Engineering Applications of Artificial Intelligence*, vol. 25, no. 7, pp. 1413–1424.

[5] T. F. Cootes, G. J. Edwards, C. J. Taylor (1998) Active appearance models, *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 484–498.

[6] G. Davies, D. Christie (1982) Face recall: An examination of some factors limiting composite production accuracy, *Journal of Applied Psychology*, vol. 67, no. 1, pp. 103–109.

Table 4: Means (standard deviations) of the dependent variables in the comparison of the IGA and IDE algorithm

Algorithm	Generations	Back/Redo count	Time taken	Without target rating	With target rating
IGA	5.05(2.50)	0.07(0.23)	150s(74.4s)	6.39(1.41)	5.00(1.74)
IDE	5.34(2.19)	0.14(0.32)	161s(55.3s)	6.55(1.21)	4.68(1.74)

[7] C. D. Frowd, P. J. B. Hancock, EvoFIT, www.evofit.co.uk, Accessed 27/04/2015.

[8] C. D. Frowd (2001) EvoFIT: A Holistic, Evolutionary Facial Imaging System. PhD thesis, Department of Psychology, University of Stirling.

[9] C. D. Frowd, P. J. B. Hancock, D. Carson (2004) EvoFIT: a holistic, evolutionary facial imaging technique for creating composites, *Transactions in Applied Perception*, vol. 1, no. 1, pp. 19–39.

[10] C. D. Frowd, J. Park, A. McIntire, V. Bruce, M. Pitchford, S. Fields, M. Kenirons, P. J. Hancock (2008) Effecting an improvement to the fitness function. How to evolve a more identifiable face, *Proceedings of the ECSIS Symposium on Bio-inspired Learning and Intelligent Systems for Security*, pp. 3–10.

[11] S. J. Gibson, C. J. Solomon, A. Pallares Bejarano (2003) Synthesis of photographic quality facial composites using evolutionary algorithms, *Proceedings of the British Machine Vision Conference*, pp. 221–230.

[12] J. H. Holland (1973) Genetic algorithms and the optimal allocation of trials, *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105.

[13] B. Kurt, A. S. Etaner-Uyar, T. Akbal, N. Demir, A. S. Kanlikilicer, M. C. Kus, F. H. Ulu (2006) Active appearance model-based facial composite generation with interactive nature inspired heuristics, *Lecture Notes in Computer Science*, vol. 4105, pp. 183–190.

[14] M.-C. Lee, S.-B. Cho (2012) Interactive differential evolution for image enhancement application in smart phone, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp. 2411–2416.

[15] J. J. Mist, S. J. Gibson (2013) Optimization of weighted vector directional filters using an interactive evolutionary algorithm, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1691–1694.

[16] Z. S. Mohamad, A. Darvish, S. Rahnamayan (2011) Eye illusion enhancement using interactive differential evolution, *Proceedings of the IEEE Symposium on Differential Evolution*, pp. 135-141.

[17] H. Mushtaq, S. Rahnamayan, A. Siddiqi (2015) Color Separation in Forensic Image Processing Using Interactive Differential Evolution, *Journal of Forensic Sciences*, vol. 60, no. 1, pp. 212–218.

[18] J. Oinuma, K. Arakawa, H. Harashima (2014) Evaluation of genetic algorithm for interactive evolutionary face image beautifying system, *Proceedings of the 6th International Symposium on Communications, Control and Signal Processing*, pp. 594-597.

[19] A. Pallares-Bejarano (2006) Evolutionary Algorithms for Facial Composite Synthesis. PhD thesis, School of Physicial Sciences, University of Kent.

[20] K. Price, R. M. Storn, J. A. Lampinen (2006) *Differential evolution: a practical approach to global optimization*, Springer Science & Business Media.

[21] C. J. Solomon, S. J. Gibson, J. J. Mist (2013) Interactive evolutionary generation of facial composites for locating suspects in criminal investigations, *Applied Soft Computing*, vol. 13, no. 7, pp. 3298–3306.

[22] R. Storn, K. Price (1997) Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces, *Journal of global optimization*, vol. 11, no. 4, pp. 341–359.

[23] H. Takagi (2001) Interactive Evolutionary Computation: Fusion of the Capabilities for EC Optimization and Human Evaluation, *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296.

[24] J. W. Tanaka, M. J. Farah (1993) Parts and wholes in face recognition, *Quarterly Journal of Experimental Psychology*, vol. 46A, pp. 225–245.

[25] T. Valentine (1991) A unified account of the effects of distinctiveness, inversion and race in face recognition, *Quarterly Journal of Experimental Psychology*, vol. 43A, pp. 161–204.

[26] Visionmetric, EFIT-V, www.visionmetric.com, Accessed 27/04/2015.

[27] J. O. Wobbrock, L. Findlater, D. Gergle, J. J. Higgins (2011) The aligned rank transform for nonparametric factorial analyses using only anova procedures, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 143–146.

[28] D.-M. Yoon, K.-J. Kim (2012) Comparison of scoring methods for interactive evolutionary computation based image retouching system, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 617–618.

Heuristics for Optimization of LED Spatial Light Distribution Model

David Kaljun and Darja Rupnik Poklukar
Faculty of Mechanical Engineering, University of Ljubljana, Aškerčeva 6, 1000 Ljubljana, Slovenia
E-mail: david.kaljun@fs.uni-lj.si

Janez Žerovnik
Faculty of Mechanical Engineering, University of Ljubljana, Aškerčeva 6, 1000 Ljubljana, Slovenia and
Institute of Mathematics, Physics and Mechanics, Jadranska 19, Ljubljana, Slovenia
E-mail: janez.zerovnik@fs.uni-lj.si

Keywords: local search, iterative improvement, steepest descent, genetic algorithm, Wilcoxon test, least squares approximation

Received: December 1, 2014

Recent development of LED technology enabled production of lighting systems with nearly arbitrary light distributions. A nontrivial engineering task is to design a lighting system or a combination of luminaries for a given target light distribution. Here we use heuristics for solving a problem related to this engineering problem, restricted to symmetrical distributions. A genetic algorithm and several versions of local search heuristics are used. It is shown that practically useful approximations can be achieved with majority of the algorithms. Statistical tests are performed to compare various combinations of parameters of genetic algorithms, and the overall results of various heuristics on a realistic dataset.

Povzetek: Napredek tehnologije LED je omogočil izdelavo osvetljevalnih sistemov s skoraj poljubno porazdelitvijo svetlobe. Netrivialna inženirska naloga je, kako načrtovati osvetljevalni sistem ali kombinacijo svetilk za dano ciljno porazdelitev svetlobe. V sestavku predstavljamo uporabo hevrističnih algoritmov za reševanje te naloge, kjer predpostavljamo, da je porazdelitev svetlobe osno simetrična. Izkaže se, da lahko dobimo praktično uporabne rešitve z algoritmi za lokalno optimizacijo, z genetskimi algoritmi in s hibridnimi algoritmi, ki povezujejo obe ideji. Za izbiro parametrov genetskih algoritmov in za primerjavo različnih algoritmov na izbranem vzorcu realnih podatkov so uporabljeni statistični testi.

1 Introduction

Even the most simply stated optimization problems such as the traveling salesman problem are known to be NP-hard, which roughly speaking means that there is no practical optimization algorithm provided the famous P≠NP conjecture is correct [26]. From practical point of view, knowing that the problem is computationally intractable implies that we may use heuristic approaches. It is well known that best results are obtained when a special heuristics is designed and tuned for each particular problem. This means that the heuristics should be based on considerations of the particular problem and perhaps also on properties of the most likely instances. On the other hand, it is useful to work within a framework of some (one or more) metaheuristcs which can be seen as a general strategies to attack an optimization problem. Metaheuristics in contrast to heuristics often make fewer assumptions about the optimization problem being solved, and so they may be usable for a variety of problems, while heuristics are usually designed for particular problem or even particular type of problem instances. Compared to optimization algorithms, metaheuristics do not guarantee that a globally optimal solution can be found on some class of problems. We say that the heuristics search for so called near optimal solutions because in general we also have no approximation guarantee. Several books and survey papers have been published on the subject, for example [25].

Most studies on metaheuristics are experimental, describing empirical results based on computer experiments with the algorithms. As experiments provide only a sample that may in addition be biased for a number of reasons, it is often hard to draw any firm conclusions from the experimental results, even when statistical analysis is applied (see, c.f. [4] and the references there). Some theoretical results are also available, often proving convergence of a particular algorithm or even only showing the possibility of finding the global optimum.

Perhaps the most natural and conceptually simple metaheuristics is local search. In the search space of feasible solutions that is usually regarded as a "landscape", the solutions with extremal values of the goal functions are to be found. In order to speak about local search on the landscape, a topology is introduced, usually via definition of a neighborhood structure. It defines which feasible solutions can be obtained in "one step" from a given feasible solution. It is essential that the operation is computationally

cheap and that the new value of the goal function is provided. There are two basic variants of the local search, iterative improvement and best neighbor (or steepest descent). As the names indicate, starting from initial feasible solution, iterative improvement generates a random neighbor, and moves to the new solution based on the difference in goal function. The procedure stops when there has been no improvement for sufficiently long time. On the other hand, best neighbor heuristics considers all neighbors and moves to the new solution with best value of the goal function. If there is no better neighbor, the current solution is clearly a local optima. Note that given a particular optimization problem, often many different neighborhood structures can be defined giving rise to different local search heuristics. Recently, there has been some work on the heuristics that use and switch among several neighborhoods [21].

In fact, most metaheuristics can be seen as variations or improvement of the local search [1]. Examples of popular metaheuristics that can be seen as variations of local search include iterated local search, simulated annealing [17], threshold accepting [7], tabu search [11], variable neighborhood search [21], and GRASP (Greedy Randomized Adaptive Search Procedure) [8]. The other type of search strategy has a learning component added to the search, aiming to improve the obvious drawback of the local search, complete lack of memory. (An exception is the tabu search that successfully introduces a short time memory.) Metaheuristics motivated by idea of learning from past searches include ant colony optimization [6, 28, 10, 9, 19], evolutionary computation [3] and its special case, genetic algorithms, to name just a few. It is however a good question in each particular case whether learning does indeed mean an improvement [29], namely a successful heuristic search must have both enough intensification and diversification.

Genetic algorithms (GA) are optimization and search techniques based on the natural evolution principles. The basic idea is to allow a population composed of many individuals to evolve under specified selection rules to a point where some of the population individuals reach or at least get close to the optimal solution. The method was developed by John Holland, and popularized by one of his students, David Goldberg, who was able to solve a difficult problem involving the control of gas-pipeline transmission for his dissertation. Since the early days of GA, many versions of evolutionary based algorithms have been tried with varying degrees of success. Nevertheless there are some advantages of GA worth noticing [12, 23]. GA is able to work with continuous or discrete variables, does not require derivative information, it simultaneously searches from a wide sampling of the cost surface, deals with a large number of variables, is well suited for parallel computers, optimizes variables with extremely complex cost surfaces (they can jump out of a local minimum), provides a list of optimum variables not just a single solution and works well with numerically generated data, experimental data, or analytical functions.

In this paper, a comprehensive experimental study of several heuristics on an industrial problem is carried out. It extends and upgrades previous published work on the subject, in particular by introducing a statistically based comparison of the algorithms. The results of algorithms are statistically tested in order to determine significant differences between them. Another extension of previous work is the genetic algorithm parameter tunning, presented bellow. Previous related work is the following. The suitability of the model and practical applicability have been shown in [14]. Attempting to improve and speed up the optimization, different metaheuristics have been implemented and compared. The conference paper [13] reports results of a comparison of local search with a naive genetic algorithm. A hybrid genetic algorithm was proposed in [15].

Here we implement and run two versions of genetic algorithm, a standard genetic algorithm (SGA) and a hybrid genetic algorithm (HGA) where we infuse a short local search as an evolution rule in hope to enhance the population. As the initial experiment was run on various computers, and consequently the results on various computers slightly differed because of different environments and in particular different random generators. The new experiment therefore repeated the complete experiment, this time on the same computer, a standard home PC with a Intel Core I7-4790K @ 4.4 GHz processor. The experiment was run in parallel on 6 threads. In addition, part of the code was rewritten to make it more machine independent. Furthermore, statistical tests on the experimental results were applied thus providing ground for tuning the parameters of genetic algorithms and for comparison of various algorithms' performance on the dataset considered.

The rest of the paper is organized as follows. In the next section we provide background of the engineering application. Section 3 provides the analytical model and the optimization problem that is addressed. In Section 4, overview of the experimental study is given. Details of local search heuristics and genetic algorithms used are given in Sections 5 and 6. Section 7 elaborates tuning of parameters for the genetic algorithms. Main experiment, comparison of local search, standard and hybrid genetic algorithms is presented in Section 8. The paper ends with a summary of conclusions and ideas for future work, Section 9.

2 Motivation – the Engineering Problem

The mass production of high power - high efficacy white Light Emitting Diodes (LEDs), introduced a revolution in the world of illumination. The LEDs at the basics enable lower energy consumption, never before seen design freedoms and of course endless possibilities on the design of optics systems. The latter in turn enables the optics designer to build a lighting system that delivers the light to the environment in a fully controlled fashion. The many possible designs lead to new problems of choosing the optimal or at least near optimal design depending on possibly

different goals such as optimization of energy consumption, production cost, and, last but not least, the light pollution of the environment. Nevertheless the primary goal or challenge of every luminaire design process is to design a luminaire with an efficient light engine. A light engine consists of a source, which are LEDs, and the appropriate secondary optics. The choice of the secondary optics is the key in developing a good system while working with LEDs. For designing such a system nowadays technology provides two options. The first option is to have the know-how and the resources to design a specific lens to accomplish the task. However, the cost of resources coupled with the development and production of optical elements may be enormous. Therefore a lot of manufactures are using the second option, that is to use ready made of the shelf lenses. These lenses are produced by several specialized companies in the world that offer different types of lenses for all of the major brands of LEDs. The trick here is to choose the best combination of lenses to get the most efficient system. The usual current practice in development process is a trial and error procedure, where the developer chooses a combination of lenses, and then simulates the system via Monte Carlo ray-tracing methods. The success heavily depends on the engineers' intuition and experience but also needs sizable computation resources for checking the proposed design by simulation. In contrast to that, we believe that using analytical models and optimization tools may speed up the design and also at the same time possibly improve the quality of solutions. The first step towards this ambitious goal is to investigate an analytical model and its use for representing single ready made lenses. For this purpose we adopt an analytical model presented by Moreno and Sun [22] and use heuristic methods based on this model to provide good approximations.

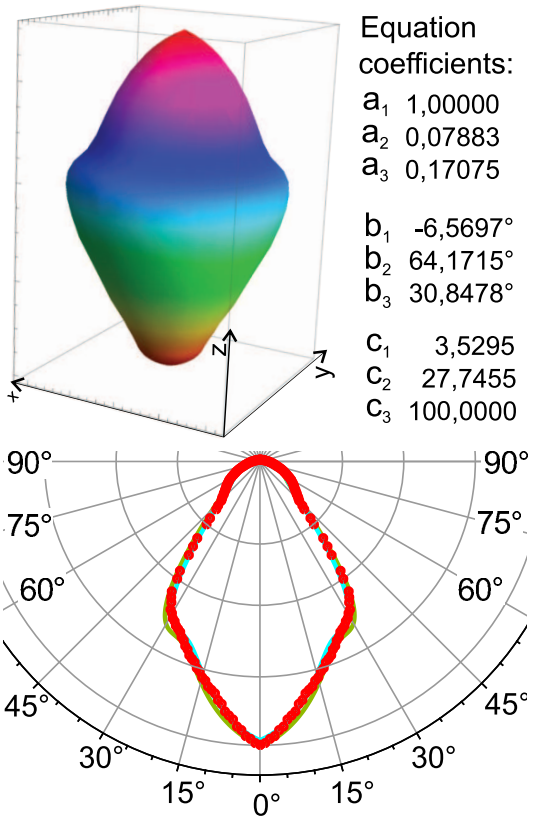


Figure 1: Fitting results on the C13353 lens with the 3D represenation.

3 Analytical Model and Problem Definition

With so many different LED's that have different beam patterns and many different secondary optics that can be placed over these LED's to control the light distribution, finding the right combination of a LED - lens combo is presumably a very complicated and challenging task. Consequently, providing a general analytical model for all of them is also likely to be a very challenging research problem. Here we therefore restrict attention to LED-lens combinations that have symmetrical spatial light distributions. In other words, the cross section of the surface which represents the spatial distribution with a section plain that is coincident with the vertical axis of the given coordinate system is alike at every azimuthal angle of offset. This yields an analytical model in two dimensions, so it describes a curve rather a surface. To produce the desired surface, we just revolve the given curve around the central vertical axis with the full azimuthal angle of 360°.

In [14], a normalizing parameter I_{max} is introduced in addition to the parameters of the original model [22] as this simplifies (unifies) the range intervals of the other three parameters: $a = [0, 1]$, $b = [0, 90]$ and $c = [0, 100]$, for all test lenses. The model used is based on the expression

$$I\left(\Phi; \mathbf{a}, \mathbf{b}, \mathbf{c}\right) = I_{max} \sum_{k=1}^{K} a_k * \cos(\Phi - b_k)^{c_k} \quad (1)$$

Assume that we have measured values $I_m(\Phi_i)$ at angles Φ_i, $i = 1, 2, \ldots, N$. The goodness of fit is, as usual, defined to be minimizing the root mean square error (RMS), or, formally [22, 24]:

$$\mathrm{RMS}\left(\mathbf{a}, \mathbf{b}, \mathbf{c}\right) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left[I_m(\Phi_i) - I(\Phi_i, \mathbf{a}, \mathbf{b}, \mathbf{c})\right]^2} \quad (2)$$

For a sufficiently accurate fit, the RMS value must be less than 5% [22, 24]. On the other hand, current standards and technology allow up to 2% noise in the measured data. Therefore, the target results of the fitting algorithms are at less than 5% RMS error, but at the same time there is no practical need for a solution with less than 1% or 2% RMS error.

We will assume that all data is written in form of vectors v= (polar angle [Φ], intensity [I]). In reality, measured

photometric data from the lens manufacturers is available in one of the two standard coded formats. These are the IESNA photometric digital format *.ies [27] used primarily in the USA and the European format EULUMDAT *.ldt [2]. The data in the two standard formats can easily be converted into a list of vectors. In addition, due to the parameter I_{max} each dataset will be normalized during the preprocessing so that in each instance the maximal intensity of the vectors will be 1, and the normalizing value I_{max} is given as additional input value to the algorithms.

The problem can formally be written as:

INPUT: I_{max} and a list of vectors v= (polar angle [Φ], intensity [I])
TASK: Find parameters (a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3, c_3) that minimize the RMS error (2).

4 Overview of the Experimental Study

Although the minimization problem defined above is conceptually simple, it is on the other hand likely to be computational hard. In other words, it is a min square error approximation of a function for which no analytical solution is known.

The experiment was set-up to test the algorithms performance on different real life LED-lens combinations.

We have chosen a set of real available lenses to be approximated. The set was taken from the online catalogue of one of the biggest and most present manufacturer in the world Ledil Oy Finland [18]. The selection from the broad spectrum of lenses in the catalogue was based on the decision that the used LED is of the XP-E product line from the manufacturer Cree [5]. And the demand that the lenses have a symmetric spatial light distribution. We have preserved the lens product codes from the catalog, so the reader can find the lens by searching the catalog for the code from the first column in tables below, c.f. Table 1.

All of the chosen lenses were approximated with all algorithms. To ensure that algorithms' results could be compared the target error was set to 0% and the runtime was defined in terms of basic steps that is defined as a generation of a feasible solutions in the local search and an adequate operation for genetic algorithms. This implies that the wall clock runtime was also roughly the same for all algorithms. Details are given below.

In the experiment and in the study, we address the optimization problem as a discrete optimization problem. Natural questions that may be asked here is why use heuristics and why discrete optimization heuristics on a continuous optimization problem. First, application of an approximation method is justified because there is no analytical solution for best approximation of this type of functions. Moreover, in order to apply continuous optimization methods such as the Newton method, usually we would need a good approximation in order to assure convergence. Therefore a method for finding good starting solution before running fine approximation based on continuous optimization methods is needed. However, in view of the at least 2% noise in the data, these starting solutions may in many cases already be of sufficient quality! Nevertheless, it may be of interest to compare the two approaches and their combination in future work, although it is not of practical interest for the engineering problem regarded here.

When considering the optimization problem as a discrete problem, the values of parameters to be estimated will be $a_\star \in [0, 0.001, 0.002, \ldots, 1]$, $b_\star \in [-90, -89.9, -89.8, \ldots, 90]$, and $c_\star \in [0, 1, 2, \ldots, 100]$. Hence, the discrete search space here consists of $N_t = 1000^i * 1800^i * 100^i \sim 5,83 * 10^{24}$ tuples $t = (a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3)$.

In the experiments, all the heuristics were tested on all instances of the dataset, a long run and a short run. The long run is defined to be 4 million steps that are defined to be equivalent of one iteration of a basic local search heuristics, in other words it is the number of feasible solutions generated by the iterative improvement. The time for other heuristics is estimated to be comparable, and will be explained in detail later. Short runs are one million and two hundred thousand steps long and the long runs have four million steps. The long run CPU time per algorithm and lens was measured to be 16 minutes on the processor Intel Core I7-4790K @ 4.4 GHz and 16 GB of RAM. The code is not fully optimized. The overall runtime of the experiment was substantially lowered by use of parallelization. We ran the experiment on 6 of the 8 available CPU threads.

5 Local Search Heuristics

First we discuss the specific local search type heuristics. As the original problem is a continuous optimization problem, compared to discrete optimization, there are even more possibilities to define a neighborhood for the local search based heuristics. In fact, the neighborhoods we use can be seen as variable neighborhoods though they are all similar. Below we define two neighborhoods that were implemented.

We have started our experiments with two basic local search algorithms, steepest descent (SD) and iterative improvement (IF), where in both cases the neighborhoods were defined in the same way. We call this neighborhood fixed step size neighborhood. The third local search algorithm (IR) is iterative improvement using a second type of neighborhood with random step size. Roughly speaking, given a step size and direction as before, we randomly make a step in the direction chosen and the step is at most as long as in the fixed size neighborhood search. Of course, there may be other neighborhoods that would be worth consideration. The main reason for not extending the selection of neighborhoods is simply the fact that they already gave us results of sufficient quality. The local search type heuris-

tics used here are explained in more detail below.

5.1 Steepest Descent (SD)

The *steepest descent* (SD) algorithm begins with the initialization of the initial function parameter values that are $a_1 = a_2 = a_3 = 0.5$, $b_1 = b_2 = b_3 = 0$, and $c_1 = c_2 = c_3 = 1$. Next it initializes the search step values which are for $da = 0.01$, for $db = 1$ and for $dc = \frac{I_{max}}{10}$ giving the 512 neighbors of the initial solution: $(a_1 \pm da, b_1 \pm db, c_1 \pm dc, a_2 \pm da, b_2 \pm db, c_2 \pm dc, a_3 \pm da, b_3 \pm db, c_3 \pm dc)$. If there are several neighbors with better RMS value, the search moves to the neighbor with minimal RMS value (if there are more minimal neighbors, one of them is chosen, all with the same probability). If none of the 512 is better than the current solution, a new set of neighboring solutions is generated, this time with a step size of $d_{n+1} = d_n + d_0$. This is repeated until $n = 10$. If there still is no better solution the search stops, the initial step value is multiplied by 0.9 and the search resumes from the current solution with a smaller initial step. The algorithm stops when the number of generated solutions reaches T_{max}.

5.2 Iterative Improvement – Fixed Neighborhood (IF)

The *iterative improvement with fixed neighborhood* (IF) algorithm initializes the same neighborhood as SD. Instead of considering all 512 neighbors, the algorithm generates a neighbor randomly, and immediately moves to that neighbor if its RMS value is better than the current RMS value. If no better neighbor is found after 1000 trials, it is assumed that no better neighbor exists. As above, the algorithm changes the size of the step value and continues the search in the same manner as SD algorithm does. The algorithm stops when the number of generated solutions reaches T_{max}.

5.3 Iterative Improvement – Variable Neighborhood (IR)

The *iterative improvement with a variable neighborhood* (IR) algorithm begins as the previous two algorithms. It initializes the same initial function parameter values but a different neighborhood which has the search step value within a range, rather than a static fixed value. The ranges are for $da_1 = da_2 = da_3 = \{-0.1, -0.099, -0.098, \ldots, 0.1\}$, for $db_1 = db_2 = db_3 = \{-9, -8.9, -8.8, \ldots, 9\}$ and dc_1 = dc_2 = dc_3= $\{-10, -9, -8, \ldots, 10\}$ It begins generating solutions, using the step range around the initial solution and calculating their RMS error. As soon as it generates a better solution, it stops, shifts the focus on that solution, resets the step range to the initial value, and continues the search in the neighborhood of the new best solution. If after four hundred thousand generated solutions no better solution is found, the step range gets doubled, and the search continues in the current neighborhood with a larger neighborhood. The stopping condition is the same as before.

Table 1: RMS error (best values) after $4 \cdot 10^6$ calculating operations

Lens/Alg.	SD	IF	RAN	IR
C13353	9.7572	**4.9422**	**5.3896**	9.2435
CA11265	4.154	**2.5374**	**3.722**	4.9367
CA11268	2.6058	**2.4788**	**2.4984**	4.0278
CA11483	**3.2673**	3.3951	**3.1944**	3.5698
CA11525	3.5799	**1.0365**	**1.4805**	2.8385
CA11934	**2.1729**	**1.4969**	2.6169	3.5317
CA12392	**1.639**	**1.5905**	1.9988	3.3103
CA13013	1.7555	**0.9042**	**1.2872**	1.7656
CP12632	**4.576**	**4.3207**	4.9078	6.7152
CP12633	7.1202	**2.936**	**2.7363**	3.8963
CP12634	**5.7641**	**5.6363**	6.1473	6.4242
CP12636	**3.1178**	**3.0801**	3.9602	4.3642
Median	3.4236	2.7367	2.9654	3.9621

5.4 Comparison of Local Search Heuristics

To reduce the performance influence of the initial solution we fixed it on all of the local search heuristics which began from the same initial solution that had the parameters set to $a_1 = a_2 = a_3 = 0.5$, $b_1 = b_2 = b_3 = 0$, and $c_1 = c_2 = c_3 = 1$. As the number of steps the local search heuristics need to find a local optima can vary heavily, it is natural to run a multi start version. As the local search runs sometimes improve the solutions in later iterations and because some preliminary experiments with multi start versions of the local search algorithms did not show any obvious advantage, we do not consider the multistart version here. However, the trivial *random search algorithm* (RAN) is included in the comparison of the local search algorithms. RAN algorithm is essentially a random solution generator that has only one simple rule. The rule is the boundary definition of the search space, so that the solutions generated stay inside the search space limits, hence RAN resembles a pure guessing exercise and any meaningful algorithm has to outperform RAN.

Table 1 and Table 2 provide best found solutions for different local search algorithms. Best two solutions are written in bold.

We can observe that most of the algorithms find a good (RMS$<5\%$) solution on almost all instances on both the long and short runs. Obviously, the majority of IF results are among the best, but also SD and RAN perform very good on some instances. Therefore we further compare the algorithms using a statistical test. We test the null hypothesis H_0: *The median of differences between results of algorithms equals 0.* The test used is a non-parametric *related samples Wilcoxon signed rank test*, see [30], and the significance level is 0.05. This means that if asymptotic significance is less or equal to 0.05, the H_0 is rejected (there is a significant difference between algorithms). Note that

Table 2: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations

Lens/Alg.	SD	IF	RAN	IR
C13353	9.7572	**5.0976**	**5.3896**	10.137
CA11265	4.154	**2.5389**	**3.9455**	6.4726
CA11268	2.6058	**2.4797**	**2.4984**	4.0278
CA11483	**3.2673**	**3.4077**	3.6977	4.3763
CA11525	3.5799	**1.0381**	**1.4805**	4.4415
CA11934	**2.1729**	**1.5547**	2.6169	3.5317
CA12392	**1.639**	**1.5924**	1.9988	3.3103
CA13013	1.7555	**0.9043**	**1.2872**	2.7156
CP12632	**4.576**	**4.3292**	4.9078	6.7152
CP12633	7.1202	**2.9366**	**2.7363**	4.2827
CP12634	**5.7641**	**5.638**	6.1473	6.4713
CP12636	**3.1178**	**3.0801**	3.9602	5.2287
Median	3.4236	2.7377	3.217	4.4089

Table 3: Asymptotic significances of Wilcoxon signed rank test for results of local search heuristics at $4 \cdot 10^6$ calculating operations

	SD	IF	RAN	IR
SD		**0.007**	0.388	0.136
IF			**0.008**	**0.002**
RAN				**0.002**

the same test will be repeated on the other versions of algorithms further down the text.

Tables 3 and 4 confirm that IF significantly outperforms the other algorithms on the dataset. Also we can observe that the null hypothesis could not be rejected between RAN and SD. However we can see a significant deviation in the result of the IR algorithm which is the worst of the algorithms having the median value of $M_d = 3,9621$.

We conclude that in both the long and short runs algorithm IF prevails. Therefore IF will be our choice when infusing local search in the hybrid genetic algorithm.

6 Genetic Algorithms

The search for a more advanced heuristic method resulted in a very large pool of promising alternatives such as particle swarm optimization [19], firefly algorithm [28, 10], bat algorithm [9] and of course the well known genetic algorithms to name just a few of them. In this experimental study we use a *standard genetic algorithm* (SGA) [23] and a *hybrid genetic algorithm* (HGA) [15] that in fact mimics the evolutionary behavior [12, 20, 23], but is enhanced at every generation with the use of a local search algorithm. Encouraging preliminary results with HGA are reported in [15]. We wish to note that in the conference paper [13] local search based heuristics were compared to another version of genetic algorithm. As the algorithm in [13] used nonstandard genetic operators, it has been argued that the results are not very useful, and hence we decided to do another comparison including SGA. We wish to note that the experimental results given in this paper may slightly differ to preliminary reports [13, 15] because the preliminary results were performed on various computers, and the new experiment reported here is completely rerun on the same machine. Also, parts of the code were rewritten in order to be more computer and system independent.

Table 4: Asymptotic significances of Wilcoxon signed rank test for results of local search heuristics at $1.2 \cdot 10^6$ calculating operations

	SD	IF	RAN	IR
SD		**0.008**	0.695	**0.034**
IF			**0.004**	**0.002**
RAN				**0.002**

6.1 Standard Genetic Algorithm (SGA)

In our genetic algorithms we use three genetic operators: selection, cross-breading and mutation. The selection [12] operator works as a kind of a filter where more fitter individuals in a population get to have higher weights as the less fitter. This is then transmitted to the cross-breading operator in the way that the individuals with higher weights are more likely to be chosen as parents.

The cross-breading or crossover operator [12, 20, 23] is where a population is created by generating new solutions. These are created by randomly combining and crossing parameters from two randomly chosen parent solutions from the current population. The crossing is done via cross point so that every parent pair produces a pair of children. The cross point is chosen randomly and the children are generated in the following sequence $C_1 = [P_1^{bCP}, CP, P_2^{aCP}]$ and $C_2 = [P_2^{bCP}, CP, P_1^{aCP}]$, where C_n is the child being generated, CP is the cross point parameter, P_n^{aCP} are all of the parents parameters that are after the CP and P_n^{bCP} are all of the parents parameters that are before the CP.

The last operator in every generation is the self adapting mutation[23] operator which finalizes the individuals in the new population. The mutation operates in the following manner: in the randomly chosen individual, a random number of parameters are chosen to be changed (mutated) which is done by adding a randomly chosen value for $da_1 = da_2 = da_3 = \{-0.01, -0.009, -0.008, \ldots, 0.01\}$, for $db_1 = db_2 = db_3 = \{-0.25, -0.24, -0.23, \ldots, 0.25\}$ and $dc_1 = dc_2 = dc_3 = \{-2.5, -2.4, -2.3, \ldots, 2.5\}$ to the current parameter value.

The whole algorithm then begins with the generation and calculation of the initial population (the zero population). Next it sorts the population entities from the fittest to the least fit and applies weights to them. After the sorting process the algorithm generates with the crossover operator the next generation, which is then submitted to mutation with the adaptive mutation operator. When the new generation is fully formed the algorithm begins the process from

the point of selection. It continues to do so until the last generation is finalized. In order to assure comparable running times, the number of generations to be generated is calculated as the quotient of the maximal number of iterations minus the population size and the population size $N_G = (T_{max} - N_P)/N_P$. Where N_G stands for number of generations, T_{max} for the total number of iterations and N_P for the number of individuals in each generation (population size).

6.2 Hybrid Genetic Algorithm (HGA)

To test our theory of an advanced genetic algorithm we altered the standard genetic algorithm in a way that we infused a local optimization as an operator in every generation. We call the modified algorithm *hybrid genetic algorithm* (HGA). The hybrid genetic algorithm works in the same way as the standard one but with an extra operator before the crossover. It starts with generating the initial solution and sorts the entities in the current solution from the fittest to least fit. Then instead of directly cross breading the new generation it first runs the iterative improvement with fixed neighborhood algorithm on 10 best entities of the current generation which in turn get locally optimized (enhanced) for a number of iterations. After that the HGA follows the same path as the standard genetic algorithm does. For the number of generations to be executed on HGA algorithm, the formula is a bit more complicated, because it has to include the iterations of the local search. The formula can be written as $N_G = (T_{max} - N_P)/(N_P + 10 * N_{lo})$. Where the additional parameter N_{lo} stands for number of local search iterations. The result has to be rounded, because the algorithm cannot stop in the middle of a generation evaluation. For example if you would calculate the number of generations for the HGA13 with the above formula you would get 9.972 which gets rounded to 10. This is the reason for the minor deviation ($div_{max} = 2,5\%$) of the overall T_{max} on the HGA algorithms.

7 Parameters of the Genetic Algorithms

In order to enable fair comparison among various heuristics, the same runtime was given to all competitors. As the wallclock runtime can depend heavily on particular implementation, we measure runtime in so called basic time steps. One step of local search algorithm is naturally defined as a generation (and handling) of one feasible solution. For the genetic algorithms, time needed for the basic operations is estimated in terms of local search basic steps. This is explained in detail in the first subsection. Genetic algorithms are divided into four groups depending on the time allowed for local search improvement of the members of population.

We fix the length of the local search runs and then look for most suitable parameters of the particular HGA version.

Table 5: Parameter combinations for SGA*

Algorithm	# pop.	# gen.	# LS iter.
SGA 1	1000	3999	NA
SGA 2	5000	799	NA
SGA 3	10000	399	NA
SGA 4	50000	79	NA
SGA 5	100000	39	NA

Table 6: Parameter combinations for HGA*1

Algorithm	# pop.	# gen.	# LS iter.
HGA 1 1	1000	40	10000
HGA 2 1	5000	38	10000
HGA 3 1	10000	36	10000
HGA 4 1	50000	26	10000
HGA 5 1	100000	20	10000

This gives rise to four groups of algorithms: SGA, HGA*1, HGA*2, and HGA*3. Tables 5, 6, 7, and 8 give different parameter combinations for the genetic algorithms.

Tuning of other parameters of genetic algorithms is explained in detail below.

7.1 Runtime

To be able to compare the genetic algorithm performance to the local search algorithms we locked the total amount of computation iterations (one computation iteration in our case is the evaluation of the RMS error at the given coefficient values) on the genetic algorithms to four million on the long run and 1.2 million on the short runs, as it was in the local search algorithms. We then chose different population sizes and calculated the number of generation and local search iterations needed to achieve the desired four million calculation iterations as close as possible (minor deviations can occur due to the restriction that we are always evaluating a whole generation).

7.2 Parameter Tuning

In order to perform the final experiment we first have to choose the algorithms that would be competing in the experiment. As it would be unfeasible to compare all of the possible variations of the multi start genetic algorithms, we formed four groups, as presented in the previous section. We applied the statistical test on these groups and accord-

Table 7: Parameter combinations for HGA*2

Algorithm	# pop.	# gen.	# LS iter.
HGA 1 2	1000	20	20000
HGA 2 2	5000	19	20000
HGA 3 2	10000	19	20000
HGA 4 2	50000	16	20000
HGA 5 2	100000	13	20000

Table 8: Parameter combinations for HGA*3

Algorithm	# pop.	# gen.	# LS iter.
HGA 1 3	1000	10	40000
HGA 2 3	5000	10	40000
HGA 3 3	10000	10	40000
HGA 4 3	50000	9	40000
HGA 5 3	100000	8	40000

Table 9: RMS error (best values) after $4 \cdot 10^6$ calculating operations for SGA*

Lens/Alg.	SGA1	SGA2	SGA3	SGA4	SGA5
C13353	9.3526	8.0242	9.3895	**4.8163**	**4.848**
CA11265	5.2983	5.2999	3.8483	**3.0568**	**2.8673**
CA11268	4.5748	3.3628	2.7874	**2.7845**	**2.5248**
CA11483	3.8848	3.8262	3.8775	**3.6547**	**3.5613**
CA11525	4.0658	1.6667	2.2655	**2.0129**	**1.2501**
CA11934	2.5642	3.3076	3.3842	**1.8616**	**2.1933**
CA12392	3.493	2.5458	**2.376**	**2.3139**	2.4982
CA13013	2.9739	**1.1658**	1.4496	1.2332	**1.128**
CP12632	4.3657	4.5959	5.9514	4.4332	4.4827
CP12633	4.447	3.3542	2.855	**2.5418**	**2.5485**
CP12634	5.7747	5.7038	**5.6663**	5.7493	**5.6712**
CP12636	4.2818	4.1577	3.8485	**3.5941**	**3.491**
Median	4.3238	3.5945	3.6163	2.9207	2.7079

ing to the results chose the one that would advance into the final experiment.

7.3 SGA* Test

When observing the test results presented in Table 11 and Table 12 we see that SGA1 in both the long and short runs statistically differs from the other four algorithms. It also has the worst median where $M_d = 4.3238$. We could not reject the null hypothesis in the case of SGA2 and SGA3 on the long run and on the short run between SGA2, SGA3 and SGA5. The long run shows us shared leadership between SGA4 and SGA5, but in the long run the SGA5 prevailes with the overall best median of $M_d = 2.7079$. This leads us to ultimately choose the SGA5 as the representative of the standard genetic algorithms in the final experiment.

7.4 HGA*1 Test

The statistical results show that there in no significant statistical difference between the HGA*1 algorithms (the null hypothesis could not be rejected). Because of that we have to take a look at the median values. In both runs HGA41 had the best median value which was a bit lower on the long run $M_d = 2.5840$. Hence we chose the HGA41 algorithm from this group to advance in the final experiment.

Table 10: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations for SGA*

Lens/Alg.	SGA1	SGA2	SGA3	SGA4	SGA5
C13353	9.3526	8.0242	9.3895	**6.3401**	**5.7489**
CA11265	5.2983	5.2999	**3.8483**	**3.3367**	3.933
CA11268	4.5748	3.3628	**2.7874**	**2.7845**	2.8694
CA11483	3.8848	3.8262	3.8775	**3.6547**	**3.6344**
CA11525	4.0658	**1.6667**	2.2655	**2.0129**	2.0875
CA11934	**2.5642**	3.3076	3.3842	**2.4779**	2.8535
CA12392	3.493	2.5458	**2.376**	**2.3139**	2.5325
CA13013	2.9739	**1.1658**	1.4496	**1.2493**	1.4786
CP12632	**4.3657**	4.5959	5.9514	**4.4332**	4.807
CP12633	4.447	3.3542	2.855	**2.6167**	**2.5485**
CP12634	5.7747	**5.7038**	**5.6663**	5,7493	5.7481
CP12636	4.2818	4.1577	**3.8485**	**3,5941**	3.8838
Median	4.3238	3.5945	3.6163	3.0606	3.2519

Table 11: Asymptotic significances of Wilcoxon signed rank test for results of SGA* at $4 \cdot 10^6$ calculating operations

SGA*	1	2	3	4	5
1		**0.034**	0.071	**0.004**	**0.004**
2			0.937	**0.019**	**0.002**
3				**0.005**	**0.005**
4					0.272

7.5 HGA*2 Test

In the HGA*2 group the HGA12 significantly differs from the rest of the group in both runs, with the worst median of $M_d = 3.1785$. The rest of the algorithms perform pretty much the same, so there is no visible difference between them in the long run and a slight inconclusive difference in the short run. Therefore we once again chose the algorithm to be advanced to the final experiment based on the minimal median value. The HGA42 algorithm has the best overall median value of $M_d = 2.7805$ and consequently is the one which represents this group in the final experiment.

7.6 HGA*3 Test

The last group's results are similar to the previous two. On the long run the HGA13 algorithm shows no significant difference from the others. There is also no significant statis-

Table 12: Asymptotic significances of Wilcoxon signed rank test for results of SGA* at $1.2 \cdot 10^6$ calculating operations

SGA*	1	2	3	4	5
1		**0.034**	0.071	**0.004**	**0.015**
2			0.937	**0.019**	0.117
3				**0.005**	0.158
4					0.084

Table 13: RMS error (best values) after $4 \cdot 10^6$ calculating operations for HGA*1

Lens/Alg.	HGA11	HGA21	HGA31	HGA41	HGA51
C13353	5.9871	**3.8263**	3.8339	**3.5616**	5.4869
CA11265	2.9985	2.8531	3.025	**2.824**	**2.8131**
CA11268	2.5578	**2.3826**	2.5418	2.5907	**2.3417**
CA11483	**3.1709**	**3.1571**	3.2288	3.2077	3.6024
CA11525	**1.4021**	**1.3965**	1.5203	1.5096	1.4318
CA11934	3.0945	2.1895	2.5715	**1.8058**	**1.9126**
CA12392	2.3158	2.365	2.3345	**2.038**	**2.0077**
CA13013	1.3588	**1.0952**	1.2681	**0.9672**	1.1257
CP12632	**4.38**	4.3803	**4.3495**	4.3852	4.402
CP12633	3.1962	**2.5102**	2.8362	**2.5773**	2.6248
CP12634	5.7581	**5.6829**	**5.6919**	5.7342	5.7757
CP12636	**2.288**	3.0882	**2.3877**	2.5551	2.8276
Median	3.0465	2.6816	2.7038	2.5840	2.7189

Table 14: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations for HGA*1

Lens/Alg.	HGA11	HGA21	HGA31	HGA41	HGA51
C13353	6.1767	**4.3002**	4.3135	**3.8082**	5.4869
CA11265	3.2252	3.3075	3.4191	**3.1401**	**2.8131**
CA11268	2.5578	**2.3826**	2.5418	2.6582	**2.3417**
CA11483	**3.2573**	3.4174	3.3694	**3.2284**	3.6024
CA11525	**1.4021**	**1.4084**	1.5203	1.5822	1.4318
CA11934	3.0945	2.4831	3.0026	**1.8058**	**1.9126**
CA12392	2.4432	2.365	2.3345	**2.038**	**2.0077**
CA13013	1.5113	1.3467	1.4484	**1.1819**	**1.1989**
CP12632	4.5072	4.5397	**4.3849**	**4.3852**	4.5023
CP12633	3.2601	**2.6138**	3.0894	**2.5944**	2.8204
CP12634	5.8198	**5.7005**	**5.6919**	5.9876	5.7757
CP12636	**2.288**	3.0882	2.9039	**2.5551**	2.8276
Median	3.1599	2.8509	3.0459	2.6262	2.8167

Table 15: Asymptotic significances of Wilcoxon signed rank test for results of HGA*1 at $4 \cdot 10^6$ calculating operations

HGA*1	1	2	3	4	5
1		0.060	0.347	0.136	0.239
2			0.117	0.347	0.814
3				0.099	0.695
4					0.117

Table 16: Asymptotic significances of Wilcoxon signed rank test for results of HGA*1 at $1.2 \cdot 10^6$ calculating operations

HGA*1	1	2	3	4	5
1		0.239	0.583	0.158	0.099
2			0.308	0.136	0.48
3				**0.041**	0.239
4					0.583

tical difference between HGA23 and HAGA53. The best two on the long run are HGA33 and HGA43, comparing the medians. The short run results confirm the picture we get from the long run. The HGA13 and HGA23 do not significantly differ, and are the worst in the group. Also there is no significant difference between HGA33, HGA43 and HGA53. As above we choose the winner based on the median values. The best median value result was obtained by the HGA43 algorithm $M_d = 2.6589$.

8 Final Experiment

Based on statistical tests on four groups of genetic algorithms we acquired four winning algorithms, with seemingly best tuned parameters inside each group. The final experiment will compare those four algorithms with the best local search algorithm IF. Table 25 show the lowest

Table 17: RMS error (best values) after $4 \cdot 10^6$ calculating operations for HGA*2

Lens/Alg.	HGA12	HGA22	HGA32	HGA42	HGA52
C13353	4.7054	4.2986	4.1253	**3.3723**	**3.4518**
CA11265	3.796	**2.9932**	**2.9978**	3.0308	3.0097
CA11268	2.5909	2.5584	**2.4421**	2.6774	**2.5356**
CA11483	3.2911	3.2074	**3.1022**	3.3761	**3.1906**
CA11525	1.6185	**1.4497**	1.5384	**1.356**	1.5003
CA11934	3.0659	2.7858	2.9067	**2.5005**	**2.2335**
CA12392	2.3725	**2.0664**	2.3302	**2.2183**	2.4562
CA13013	1.1562	1.2387	**0.9072**	**1.0672**	1.1936
CP12632	**4.4391**	4.5954	5.1023	4.4996	**4.3441**
CP12633	2.7446	**2.5122**	2.6415	**2.5266**	2.576
CP12634	5.7346	5.7481	**5.7068**	**5.7009**	5.8207
CP12636	4.2152	**3.1483**	3.2091	**2.8835**	4.1649
Median	3.1784	2.8895	2.9522	2.7804	2.7929

Table 18: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations for HGA*2

Lens/Alg.	HGA12	HGA22	HGA32	HGA42	HGA52
C13353	5.3734	4.4111	4.1253	**3.3723**	**3.7201**
CA11265	4.061	**2.9932**	3.3168	**3.1738**	3.2775
CA11268	2.7644	2.5751	**2.4421**	2.7098	**2.5356**
CA11483	3.7451	**3.2304**	3.7034	3.3761	**3.2599**
CA11525	1.6185	1.5703	1.5384	**1.356**	**1.5177**
CA11934	3.3468	**2.7858**	3.0841	**3.0173**	3.1701
CA12392	2.4229	**2.1467**	2.4255	**2.2183**	2.4562
CA13013	1.1562	1.2387	**1.1321**	**1.1125**	1.1936
CP12632	**4.4624**	4.5954	5.3015	4.6303	**4.356**
CP12633	2.7446	**2.5122**	2.8733	**2.5266**	2.588
CP12634	**5.7346**	5.7481	**5.7068**	5.7586	5.8207
CP12636	4.2152	3.4343	**3.2091**	**2.8969**	4.2121
Median	3.5459	2.8895	3.1466	2.9571	3.2149

Table 19: Asymptotic significances of Wilcoxon signed rank test for results of HGA*2 at $4 \cdot 10^6$ calculating operations

HGA*2	1	2	3	4	5
1		**0.019**	**0.023**	**0.019**	**0.023**
2			0.638	0.239	0.937
3				0.182	0.754
4					0.754

RMS errors at four million calculating iterations for every competing algorithm. Table 26 show the lowest RMS errors at bout one million calculating iterations for every competing algorithm. The asymptotic significances of related samples Wilcoxon signed rank test for results are shown in Table 27 and Table 28.

We can see a significant deviation in the result of the SGA5 algorithm which is the worst of the algorithms having the median value of $M_d = 3.2519$ on the short run. In short runs SGA5 significantly differs to all other algorithms, hence is clearly the worst among the competitors. However, there is no significant difference between algorithms HGA41, HGA42, HGA43 and IF. Simply counting the number of emphasized results (best two on particular instance) favourizes IF ($8 + 8 = 16$), followed by HGA41 ($6 + 7 = 13$). The other two only have eight emphasized results: HGA42 ($3 + 5 = 8$) and HGA43 ($4 + 4 = 8$). So we conclude that on the dataset used here the best are

Table 20: Asymptotic significances of Wilcoxon signed rank test for results of HGA*2 at $1.2 \cdot 10^6$ calculating operations

HGA*2	1	2	3	4	5
1		**0.015**	0.084	**0.008**	**0.019**
2			0.182	0.875	0.347
3				**0.019**	0.814
4					0.209

Table 21: RMS error (best values) after $4 \cdot 10^6$ calculating operations for HGA*3

Lens/Alg.	HGA13	HGA23	HGA33	HGA43	HGA53
C13353	4.7929	7.4605	4.1179	**3.673**	**3.3926**
CA11265	3.5219	**2.8982**	**3.2986**	3.3426	3.4874
CA11268	2.4712	2.6368	**2.6012**	2.4733	**2.5077**
CA11483	**3.1692**	3.9305	3.3611	3.5551	**3.3291**
CA11525	1.924	1.7202	**1.3482**	**1.6992**	1.8313
CA11934	**2.7798**	3.1422	2.9557	**2.1928**	3.1123
CA12392	2.4091	2.379	2.3175	**1.9166**	**2.3143**
CA13013	1.4897	1.2844	**1.1117**	**1.031**	1.7124
CP12632	4.5923	4.8078	**4.4902**	**4.424**	4.5191
CP12633	2.7836	2.6338	**2.5233**	**2.4582**	2.7832
CP12634	5.806	**5.7097**	**5.8002**	5.8213	5.9054
CP12636	**2.617**	4.2187	3.3837	**2.8446**	3.2144
Median	2.7817	3.0202	3.1271	2.6589	3.1633

Table 22: RMS error (best values) after $1.2 \cdot 10^6$ calculating operations for HGA*3

Lens/Alg.	HGA13	HGA23	HGA33	HGA43	HGA53
C13353	5.6511	7.8509	5.0076	**4.6041**	**3.3926**
CA11265	3.5219	**3.4391**	**3.3527**	3.5108	3.4874
CA11268	2.7614	2.761	2.6012	**2.4733**	**2.5077**
CA11483	3.48	3.9305	**3.4179**	3.8551	**3.3291**
CA11525	2.1053	1.773	**1.6147**	**1.6992**	1.8313
CA11934	3.0145	3.4084	**2.9764**	**2.4433**	3.1123
CA12392	2.5199	2.4228	2.6216	**1.9166**	**2.3143**
CA13013	1.7247	1.5402	**1.1117**	**1.031**	1.7124
CP12632	4.7738	4.8078	**4.4902**	4.7755	**4.5191**
CP12633	3.5175	2.6338	**2.5233**	**2.4582**	2.7832
CP12634	5.9669	**5.7097**	**5.8002**	5.8213	5.9054
CP12636	**2.6872**	4.2187	3.4892	3.2777	**3.2144**
Median	3.2472	3.4237	3.1646	2.8755	3.1634

the algorithms HGA41 and IF which share the leadership. Finally, as the HGA41 has a slightly lower median value $M_d = 2,6263$ than the IF, the overall winner of the experiment is the HGA41 algorithm. (Note however that the last conclusion is not confirmed with statistical test.) Convergence of the best two algorithms on an instance is shown on Figure 2.

9 Conclusions

An experimental comparison of several heuristics on an engineering problem has been carried out. Among several local search heuristics, a version of iterative improvement on a suitably defined neighborhood was chosen, based on statistical test. A standard genetic algorithm, and three versions of hybrid genetic algorithms, in which members of population were improved by short runs of local search were considered. Parameters of the algorithms were tuned by running the algorithms on the dataset with several versions of parameters and the best combination of parame-

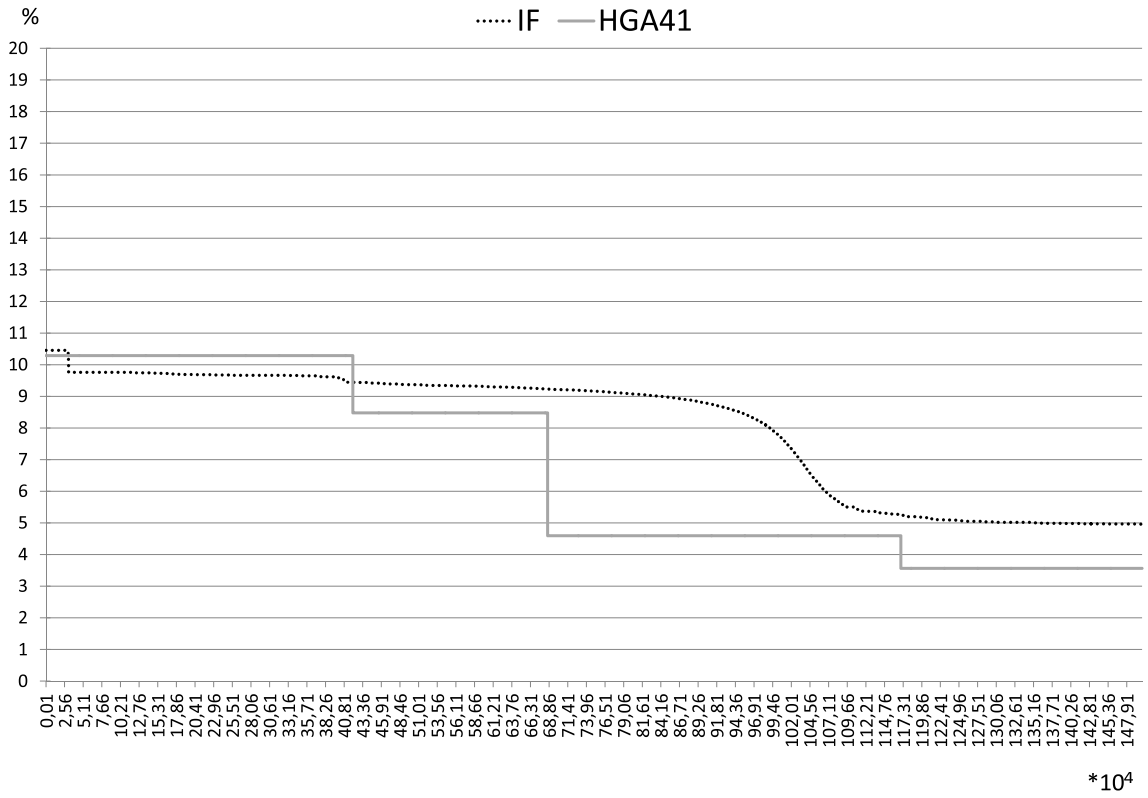


Figure 2: The convergence curves of the winning algorithms approximating the C13353 lens.

Table 23: Asymptotic significances of Wilcoxon signed rank test for results of HGA*3 at $4 \cdot 10^6$ calculating operations

HGA*3	1	2	3	4	5
1		0.308	0.347	0.084	0.48
2			**0.034**	**0.023**	0.53
3				0.099	0.347
4					**0.041**

Table 24: Asymptotic significances of Wilcoxon signed rank test for results of HGA*3 at $1.2 \cdot 10^6$ calculating operations

HGA*3	1	2	3	4	5
1		0.814	**0.05**	**0.05**	**0.05**
2			**0.015**	**0.012**	0.099
3				0.433	0.814
4					0.53

ters was selected. It may be interesting to observe that in hybrid genetic algorithms, versions with the shortest local searches were selected in all cases, meaning there is substantial number of generations possible within the runtime limit. Similarly, the standard genetic algorithm performed best when many generations were allowed and consequently, the population was smaller.

Table 25: RMS error (best values) of the final experiment after $4 \cdot 10^6$ calculating operations

Lens/Alg.	SGA5	HGA41	HGA42	HGA43	IF
C13353	4.848	**3.5616**	**3.3723**	3.673	4.9422
CA11265	2.8673	**2.824**	3.0308	3.3426	**2.5374**
CA11268	2.5248	2.5907	2.6774	**2.4733**	**2.4788**
CA11483	3.5613	**3.2077**	**3.3761**	3.5551	3.3951
CA11525	**1.2501**	1.5096	1.356	1.6992	**1.0365**
CA11934	2.1933	**1.8058**	2.5005	2.1928	**1.4969**
CA12392	2.4982	2.038	2.2183	**1.9166**	**1.5905**
CA13013	1.128	**0.9672**	1.0672	1.031	**0.9042**
CP12632	4.4827	**4.3852**	4.4996	4.424	**4.3207**
CP12633	2.5485	2.5773	**2.5266**	**2.4582**	2.936
CP12634	**5.6712**	5.7342	5.7009	5.8213	**5.6363**
CP12636	3.491	**2.5551**	2.8835	**2.8446**	3.08
Median	2.7078	2.5840	2.7804	2.6589	2.7367

In the final experiment, the best local search and the versions of genetic algorithms selected after tuning the main parameters were compared. Interesting enough, all three versions of the hybrid genetic algorithm performed better than the standard genetic algorithm, and that conclusion is supported by statistical tests. On the other hand, there is no statistically significant differences among the versions of the hybrid algorithm and the local search IF. Looking at the results closer, we conclude that a version of hybrid genetic

Table 26: RMS error (best values) of the final experiment after $1.2 \cdot 10^6$ calculating operations

Lens/Alg.	SGA5	HGA41	HGA42	HGA43	IF
C13353	5.7489	**3.8082**	**3.3723**	4.6041	5.0976
CA11265	3.933	**3.1401**	3.1738	3.5108	**2.5389**
CA11268	2.8694	2.6582	2.7098	**2.4733**	**2.4797**
CA11483	3.6344	**3.2284**	**3.3761**	3.8551	3.4077
CA11525	2.0875	1.5822	**1.356**	1.6992	**1.0381**
CA11934	2.8535	**1.8058**	3.0173	2.4433	**1.5547**
CA12392	2.5325	2.038	2.2183	**1.9166**	**1.5924**
CA13013	1.4786	1.1819	1.1125	**1.031**	**0.9043**
CP12632	4.807	**4.3852**	4.6303	4.7755	**4.3292**
CP12633	2.5485	2.5944	**2.5266**	**2.4582**	2.9366
CP12634	**5.7481**	5.9876	5.7586	5.8213	**5.638**
CP12636	3.8838	**2.5551**	**2.8969**	3.2777	3.0801
Median	3.2519	2.6263	2.9571	2.8755	2.7377

Table 27: Asymptotic significances of Wilcoxon signed rank test for results of the final experiment at $4 \cdot 10^6$ calculating operations

	SGA5	HGA41	HGA42	HGA43	IF
SGA5		0.060	0.583	0.239	**0.034**
HGA41			0.099	0.099	0.814
HGA42				0.814	0.308
HGA43					0.347

algorithm performs slightly better.

While the comparison here is based on extensive experiments and the conclusions are supported by statistical tests, there are obvious reasons that relativize the conclusions. Namely, the experiment was run on a realistic dataset that is relevant for the engineering application, and confirmed the hypothesis that these type of problems can successfully be solved by the heuristics used. On the other hand, the dataset used is relatively small, and hence the observations can be generalized only conditionally. Further work on other datasets and related optimization problems is planned.

Nevertheless, we believe that inclusion of a carefully chosen local search into a genetic algorithm is a good idea, and the present experimental study proves that.

Table 28: Asymptotic significances of Wilcoxon signed rank test for results of the final experiment at $1.2 \cdot 10^6$ calculating operations

	SGA5	HGA41	HGA42	HGA43	IF
SGA5		**0.006**	**0.008**	**0.01**	**0.005**
HGA41			0.638	0.136	0.48
HGA42				0.347	0.239
HGA43					0.099

References

[1] E. H. L. Aarts, J. K. Lenstra (1997) *Local Search Algorithms*, John Wiley & Sons.

[2] I. Ashdown (2001) Thinking Photometrically Part II., *Proceedings of the Pre-Conference Workshop (LIGHTFAIR)*.

[3] T. Bäck (1996) *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press.

[4] M. Coffin, M. J. Saltzman (2000) Statistical Analysis of Computational Tests of Algorithms and Heuristics, *INFORMS Journal of Computing*, vol. 12, pp. 24–44.

[5] Cree, Inc. *XLamp XP-E*, `www.cree.com/led-components-and-modules/products/xlamp/discrete-directional/xlamp-xpe`

[6] M. Dorigo, M. Birattari, T. Stützle (2006) Ant colony optimization, *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39.

[7] G. Dueck, T. Scheuer (1990) Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing, *Journal of Computational Physics*, vol. 90, pp. 161–175.

[8] P. Festa, M. G. C. Resende (2002) GRASP: An annotated bibliography, *Essays and Surveys on Metaheuristics*, C. C. Ribeiro, P. Hansen (eds.), Kluwer Academic Publishers, pp. 325—367.

[9] I. Fister Jr., S. Fong, J. Brest, I. Fister (2014) A Novel Hybrid Self-Adaptive Bat Algorithm, *The Scientific World Journal*, vol. 2014, Article ID 709738.

[10] I. Fister, M. Perc, S. M. Kamal, I. Fister (2015) A review of chaos-based firefly algorithms: Perspectives and research challenges, *Applied Mathematics and Computation*, vol. 252, pp. 155—165.

[11] F. Glover, M. Laguna (1997) Tabu Search In M. Panos, *Handbook of Combinatorial Optimization*, New York, Springer US, pp. 2093-2229.

[12] R. L. Haupt, S. E. Haupt (2004) *Practical Genetic Algorithms*, 2nd Ed., John Wiley & Sons.

[13] D. Kaljun, J. Žerovnik (2014) Local Search Optimization of a Spatial Light Distribution Model, *Proceedings of the Student Workshop on Bioinspired Optimization Methods and their Applications (BIOMA)*, pp. 81–91.

[14] D. Kaljun, J. Žerovnik (2014) Function fitting the symmetric radiation pattern of a LED with attached secondary optic, *Optics Express*, vol. 22, pp. 29587–29593.

[15] D. Kaljun, J. Žerovnik (2014) On local search based heuristics for optimization problems, *Croatian Operational Research Review*, vol. 5, no. 2, pp. 317–327.

[16] S. Kennedy (2005) Escaping the bulb culture: the future of leds in architectual illumination. *LEDs magazine*, vol. 1, pp. 13–15.

[17] P. J. Laarhoven, E. H. Aarts (1987) Simulated annealing: theory and applications, *Mathematics and Its Applications*, M. Hazewinkel (ed.), Springer, pp. 7–15.

[18] Ledil Oy., `www.ledil.com/`.

[19] L. Lobachinsky, A. Bahabad (2014) Using Particle Swarm Optimization to Design Broadband Optical Nano-antennas for Nonlinear Optics, *Frontiers in Optics*, Optical Society of America, paper FTh4E.3.

[20] M. Mitchell (1999) *An Introduction to Genetic Algorithms*, 5th Ed., The MIT Press.

[21] N. Mladenović, P. Hansen, J. Brimberg (2013) Sequential clustering with radius and split criteria, *Central European Journal of Operations Research*, vol. 21, suppl. 1, pp. 95–115.

[22] I. Moreno, C.-C. Sun (2008) Modeling the radiation pattern of leds, *Optics Express*, vol. 16, pp. 1808–1819.

[23] D. Simon (2013) *Evolutionary Optimization Algorithms*, John Wiley & Sons.

[24] C.-C. Sun, T.-X. Lee, S.-H. Ma, Y.-L. Lee, S.-M. Huang (2006) Precise optical modeling for led lighting verified by cross correlation in the midfield region, *Optics Letters*, vol. 31, pp. 2193–2195.

[25] E.-G. Talbi (2009) *Metaheuristics: From Design to Implementation*, John Wiley & Sons.

[26] The Millennium Prize Problems are seven problems in mathematics that were stated by the Clay Mathematics Institute in 2000, `www.claymath.org/millenium-problems/p-vs-np-problem`.

[27] The Subcommittee on Photometry of the IESNA Computer Committee (2002) Iesna standard file format for the electronic transfer of photometric data and related information, Technical Report ANSI IESNA LM-63-02, Illuminating Engineering Society of North America.

[28] M. Tuba, N. Bacanin (2014) Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems, *Neurocomputing*, vol. 143, pp. 197–207.

[29] A. Vesel, J. Žerovnik (2000) How well can ants colour graphs?, *CIT. Journal of Computing and Information Technology*, vol. 8, pp. 131–136.

[30] F. Wilcoxon (1945) Individual comparisons by ranking methods, *Biometrics*, vol. 1, pp. 80–83.

Implicit and Explicit Averaging Strategies for Simulation-Based Optimization of a Real-World Production Planning Problem

Juan Esteban Diaz and Julia Handl
Manchester Business School, The University of Manchester, Booth Street West
M15 6PB Manchester, United Kingdom
E-mail: juan.diaz@postgrad.mbs.ac.uk, j.handl@manchester.ac.uk

Keywords: discrete event simulation, failure-prone manufacturing, genetic algorithms, noise handling, production planning, simulation-based optimization, uncertainty

Received: December 1, 2014

In this study, we explore the impact of noise handling strategies on optimization performance in the context of a real-world production planning problem. Uncertainties intrinsic to the production system are captured using a discrete event simulation (DES) model, and the production plan is optimized using an evolutionary algorithm. The stochastic nature of the fitness values (as returned by the DES simulation) may impact on optimization performance, and we explore explicit and implicit averaging strategies to address this issue. Specifically, we evaluate the effectiveness of different strategies, when a limited budget of evaluations is available. Our results indicate a general advantage of implicit averaging in this setting, and a good degree of robustness with regard to population size. On the other hand, explicit averaging is found to be non-competitive, due to the cost of repeat-evaluations of the same solution. Finally, we explore a hybrid approach that uses explicit averaging to refine fitness estimates during final solution selection. Under increasing levels of fitness variability, this hybrid strategy starts to outperform pure implicit and explicit averaging strategies.

Povzetek: V študiji smo raziskali vpliv strategij za ravnanje s šumom na uspešnost optimizacije v okviru realnega problema načrtovanja proizvodnje. Negotovosti, ki se pojavljajo v proizvodnem sistemu so bile zajete z modelom simulacije diskretnih dogodkov (DES), proizvodni načrt pa je bil optimiran z uporabo evolucijskega algoritma. Ker stohastična narava vrednosti kriterijske funkcije (kot jo vrača DES) lahko vpliva na uspešnost optimizacije, smo raziskali eksplicitne in implicitne strategije povprečenja za reševanje tega problema. Natančneje, oceniti smo učinkovitost različnih strategij v primerih, ko je na voljo omejeno število ocenitev kriterijske funkcije. Rezultati v splošnem kažejo na prednost implicitnega povprečenja in dobro stopnjo robustnosti glede na velikost populacije. Po drugi strani pa smo za eksplicitno povprečenje ugotovili, da ni konkurenčno zaradi stroškov večkratnih ovrednotenj iste rešitve. Končno, raziskali smo hibridni pristop, ki uporablja eksplicitno povprečenje za izpopolnitev ocene kriterijske funkcije pri končni izbiri rešitev. S povečano stopnjo spremenljivosti kriterijske funkcije začne hibridna strategija prekašati čisti, eksplicitni in implicitni strategiji.

1 Introduction

Optimization problems that include uncertainty pose challenges that are difficult to address using standard optimization methodologies. While a portion of the optimization literature is concerned with the development of methodologies capable of identifying optimal solutions to problems with uncertainty, the application of these methods often requires stringent assumptions and / or simplifications that are necessary to satisfy relevant optimality conditions. Those methods are often insufficiently powerful to accurately incorporate the full complexity and uncertainty intrinsic to real-world problems into the problem formulation, even when their consideration is essential for the generation of reliable and feasible solutions. For this reason, solutions obtained from traditional approaches to optimization under uncertainty (such as fuzzy, stochastic and stochastic dynamic programming) may often be of limited value in producing realistic solutions for real-world problems.

Simulation-based optimization constitutes an interesting alternative in situations where the high level of complexity precludes a complete analytic formulation of a problem [7] and where uncertainty needs to be considered [8]. Simulation-based optimization involves the development of a detailed simulation model, which is then coupled with an optimizer in a black-box fashion. In other words, the optimizer operates on a (sub-)set of model parameters and the optimization process is based exclusively on the (usually stochastic) simulation responses. Evolutionary algorithms (EAs) are well-suited to black-box optimization settings, as highlighted by their wide application to real-world optimization problems that cannot be handled by analytical

approaches [14]. The feasibility and reliability of solutions become the primary consideration in such settings [8], and the EAs' flexibility in this respect typically offsets its disadvantages (specifically, the lack of guaranteed optimality of its identified solutions).

When EAs are employed as optimizers of simulation-based optimization models, fitness values become subject to the variability arising from the stochastic responses within the simulation model. The resulting noisy nature of the fitness values poses a challenge to the evolutionary optimizer, for it may mislead selection procedures [1] and lead to the propagation of inferior individuals or to the elimination of superior ones, thereby undermining algorithm performance [17]. Under these circumstances, noise handling strategies can play an important role in compensating for the impact of noise on the optimizer, and, specifically, in helping the optimizer to identify solutions that exhibit low fitness variability and give rise to high average fitness. Multiple studies have analysed situations in which noise causes perturbations during fitness evaluation, thus generating discrepancies between the observed and "true" fitness [14]. We refer the reader to [10] for a comprehensive survey of noise handling strategies proposed in the existing literature.

Implicit and explicit averaging are the two strategies most commonly employed to reduce the influence of noise in evolutionary optimization under noise. Implicit averaging relies on the EA mechanism itself to compensate for the impact of noise. Specifically, it assumes that the use of sufficiently large populations will ensure that individuals from promising regions of the search space are sampled repeatedly [10, 17], and the impact of noise can be reduced in this manner. On the other hand, explicit averaging strategies ensure that individuals are evaluated using average fitness values obtained across a specific number (n) of fitness evaluations (replicates). Statistically, this approach ensures that the expected error of fitness estimates (i.e. the difference between the observed and the "true" fitness mean) reduces with a factor of $\sqrt{n}$ [10].

Both implicit and explicit averaging strategies incur additional fitness evaluations due to (*i*) the increase in population size and due to (*ii*) the increase in the number of trials, respectively. Fitness evaluations present an important consideration in simulation-based optimization, as each replication of a simulation is time-consuming and the number of these replications may be limited by available computational time. Here, we investigate the efficiency and effectiveness of different noise-handling strategies in a realistic simulation-based optimization setting, in which the computational budget available for the optimization (and, therefore, the overall number of simulation replicates) is limited. Specifically, we compare explicit averaging against implicit averaging strategies for two different population sizes. Finally, we investigate a hybrid scheme that aims to combine the strengths of both approaches.

The remainder of this paper is organized as follows. Section 2 introduces the real-world optimization problem considered and the corresponding simulation-based optimization model developed in this study. Explicit averaging, implicit averaging and a hybrid strategy combining both approaches are described in Section 3. Section 4 presents details about the comparative analysis, and empirical findings are presented in Section 5. Overall conclusions, as well as the limitations of this study and future research directions, are discussed in Sections 6 and 7, respectively.

2 Simulation-Based Optimization Model

In this study, a simulation-based optimization approach based on the integration of discrete event simulation (DES) and a genetic algorithm (GA) is applied to address the production planning problem of a real manufacturing company presented by Diaz Leiva and Handel [6] with the difference that, here, the objective is to achieve profit maximization. Additional modifications made to the original DES and optimization models presented in [6] are stated in Sections 2.1 and 2.2, respectively.

This problem corresponds to a big bucket, multi-product, multi-level (sub-products), capacitated (constraints are considered) production planning problem of a failure-prone manufacturing system, consisting of multiple work centres with insufficient capacity to fully cover demand requirements.

The DES model was developed in SimEvents® (The MathWorks, Inc., 2014) and Matlab's GA (MI-LXPM) implementation [5] was used as the optimizer. This is the default MATLAB® R2014a's (The MathWorks, Inc., 2013) implementation for solving integer and mixed integer problems with GA. The GA employs Laplace crossover, power mutation and binary tournament selection as operators. The truncation procedure, which ensures compliance with integer constraints after crossover and mutation is described in [5]. The inbuilt constraint-handling method is the parameter free penalty function approach proposed by Deb [4].

All computations were executed in parallel on a 12 core Intel(R) Xeon(R) CPU L5640 @ 2.27GHz with 24 GB of RAM running Scientific Linux, release 6.2.

2.1 Simulation Model

The DES model employed in this study is a modified version of the model presented by Diaz Leiva and Handel [6].

The DES model represents the production of 31 products k within 7 work centres l. A work centre corresponds to the set of resources (e.g., machines, people, etc.) needed to manufacture certain products. Given that some products can be manufactured in several work centres a total of 41 processes j are considered in the DES model. A process j includes all series of events involved in the initialization of orders of a product k, its manufacturing in a specific work centre l and its storage in an specific sink s (with $s = 1, 2, \ldots, 41$).

This model intends to capture the delays (α_l) caused by work centre failures and provides the stock of products manufactured during a production period of one month (24 days composed of 3 shifts of 8 hours each). The total stock of a specific product (S_k) corresponds to the sum of lots manufactured across the different work centres as shown by the following equation:

$$S_k = \sum_{l=1}^{7} S_{k,l}, \tag{1}$$

where $S_{k,l}$ is the stock of product *k* manufactured in work centre *l*.

The first modification to the original model is that instead of using probability distribution functions (PDFs) to represent the time required to manufacture a specific production lot (T_j), constant values are assigned to those attributes according to specifications provided by the company.

Moreover, unlike the original DES model, the probability of occurrence of a work centre failure during the manufacturing of a product lot is here denoted as P_l and is modelled as attributes assigned to each work centre (rather than to each process). The probabilities used for P_l, as well as the PDFs employed to represent the delays caused by those failures (α_l), are summarized in Table 2.

Finally, an additional server was added to each process, so that the first lot of every process is processed by this server during the entire duration of each simulation replication (24 days). This modification was made to allow decision variables to take values equal to zero, a possibility not accounted for in the original model [6].

2.2 Optimization Model

The objective here is to generate production plans that try to maximize the expected sum of contributions to profit generated from processes undertaken by a failure-prone manufacturing system. The expected sum of contributions to profit is later referred to as "profit" for simplification purposes.

A total of 41 decision variables (x_j) are considered, which correspond to the number of lots to be produced in each process *j*, and are constrained to be non-negative integers. Those decision variables, specified by the GA, constitute the input to the DES model and the responses S_k obtained from the DES model are used for computing the value of the fitness function.

The fitness value *f* is calculated across *n* independent simulation replicates for each individual x as follows:

$$maximize \quad f(x) = \bar{c} = \frac{1}{n} \sum_{m=1}^{n} c_m, \tag{2}$$

where the value of n varies depending on the strategy applied (see Section 4 for details about n).

For each replication m, the responses (S_k) of the DES model are used to calculate c_m as follows:

$$c_m = \sum_{k=1}^{31} P_k, \tag{3}$$

where the total profit derived from product k is defined as:

$$P_k = S_k \times \rho_k, \tag{4}$$

where ρ_k denotes the contribution margin per lot of product k.

Additional constraints are imposed in the form of Equation 5 to avoid production levels greater than the maximum demand, to represent the requirement of sub-products and labour needed to undertake each process.

$$\sum_{j=1}^{41} a_{i,j} \times x_j \leq b_i \qquad (i = 1, 2, \ldots, 44), \tag{5}$$

where b_i denotes the magnitude of constraint *i* and $a_{i,j}$ corresponds to the amount deployed from b_i by manufacturing one lot in process *j*.

3 Noise Handling Strategies

In this study we focus exclusively on implicit and explicit averaging strategies, as these are straightforward to implement in any EA and present the approaches most commonly employed in practice. Other noise handling strategies, such as averaging by means of approximated models [3, 12, 16] and modifications of the selection scheme [2, 15] have been proposed in the literature, but are not considered here.

An explicit averaging strategy uses a fixed number n of simulation replicates to obtain an average fitness value for each individual, as described in Equation 2. These average fitness estimates are then used to inform selection probabilities in the evolutionary algorithm. Therefore, under the explicit averaging strategy (ES) here analysed, fitness of an individual x is computed across 10 independent fitness evaluations ($n = 10$) and the population size employed corresponds to 50 individuals. n is set at this relatively small level because a limited computational budget of 25,000 fitness evaluations is available, and assigning higher values to n or using a larger population size would reduce the number of generations that can be executed under ES.

In contrast, an implicit averaging strategy uses a single simulation replicate ($n = 1$) to evaluate the quality of an individual. Increased robustness towards noise is then achieved by increasing population size relative to standard settings of this parameter. Consequently, under the implicit averaging strategy (IS), a single fitness evaluation ($n = 1$) is used to compute fitness and a population size of 100 individuals is applied. Additionally, a baseline strategy (BS) is also analysed in order to evaluate the performance of implicit averaging when the population size is the same as in

ES, and therefore, twice as many generations are evolved compared to IS.

Moreover, we further describe a hybrid strategy (HS) that attempts to combine aspects of implicit and explicit averaging. This strategy applies implicit averaging ($n = 1$ and a population size of 100 individuals) throughout the evolution process, but switches behaviour towards the end of the optimization: instead of choosing the final solution based on a single fitness value, we propose to select from the final population the feasible individual with the best average fitness. Consequently, we implement a mechanism that computes the average fitness of every feasible individual of the final population across a number γ of fitness evaluations, generated from independent simulation replications. The value of γ depends on the number of feasible individuals (δ) in the final population and on the computational budget available for this last step (E), which in this case corresponds to 1000 fitness evaluations. γ is computed as follows:

$$\gamma = \left\lfloor \frac{E}{\delta} \right\rfloor . \tag{6}$$

Therefore, having a population size of 100, if every individual in the final population is feasible, 10 fitness evaluations are used to compute the average fitness of each individual. However, if infeasible individuals are present in the final population, those 1000 fitness evaluations are distributed amongst feasible individuals only.

Parameters specified for all four noise handling strategies introduced in this section are presented in Table 1.

4 Comparative Analysis

In order to test the effectiveness of the strategies under different levels of fitness variability, the following comparative analysis is undertaken for two different problem instances. Table 2 shows the different levels of uncertainty incorporated into each instance of the problem.

The performance of the EA under the proposed HS is compared with the performance observed for IS, ES and BS. In order to provide a fair comparison of the four strategies analysed, the stopping criteria selected to terminate the optimization procedure is the number of fitness evaluations. As mentioned in Section 3, a total budget of $25{,}000$ fitness evaluations is allocated for every strategy as shown in Table 1.

The simulation-based optimization model is run 60 different times for each strategy; in each run, the best solution (based on last fitness evaluations) is selected from the final population. Consequently, 240 production plans are generated per problem instance. The precise quality of each of these plans is evaluated using extensive simulation: average profit, measured in United States Dollar (USD), is computed for every production plan across 1000 profit values obtained via stochastic simulation.

Subsequently, the four sets of average profit values are depicted as cumulative distribution functions (CDFs) and stochastic dominance criterion [18] is applied to determine whether or not the optimization performance, as measured in average profit values, differs between strategies.

Furthermore, Mann-Whitney U test [11] is then conducted for paired comparisons to test whether the optimization performance achieved under the different strategies is statistically significant, expressed in the form of the following hypotheses:

- H_o : stochastic homogeneity of CDFs of average profit values obtained under both strategies
- H_a : average profit values obtained under one strategy are stochastically smaller than the ones obtained under the other strategy

Mann-Whitney U test is employed instead of t-test, since distributions of the samples analysed do not fulfil the normality assumption.

5 Results

Descriptive statistics (means, minimum values, maximum values and standard deviations) of the average profit values (as computed across 1000 independent replications) obtained under the four strategies as well as the corresponding average computation times are presented in Tables 3 and 4 for problem instance 1 and 2, respectively.

Since our intention is to test the hypothesis presented in section 4 among the four strategies, homogeneity of variances of the ranked values across the different samples is a necessary condition for Mann-Whitney U test to be a reliable test [9]. Therefore, non-parametric Levene tests [13] were performed on every combination of samples in both problem instances. In both problem instances, results from these tests indicate that variances did not differ significantly ($p > 0.05$) between the samples of ranks analysed, confirming the suitability of Mann-Whitney U test to evaluate the hypothesis above mentioned (Section 4).

Figures 1 and 2 illustrate as CDFs the 60 average profit values obtained with production plans generated under each strategy in problem instance 1 and 2, respectively. Both figures clearly show that the CDFs of average profit obtained under BS, IS and HS dominate the CDF of profit obtained under ES (first-order stochastic dominance). Furthermore, results from Mann-Whitney U test statistically show that average profit values obtained under ES are stochastically smaller ($p < 0.01$) than the ones obtained under BS, IS and HS in both problem instances, as shown in Tables 5 and 6.

These results demonstrate that ES is an inadequate noise handling strategy in our setting: this result is likely to be driven by the limited computational budget available, and a stronger performance of ES may potentially be achieved when considering performance upon convergence.

Table 1: Parameters used for baseline, implicit averaging, explicit averaging and hybrid strategies

	BS	IS	ES	HS
n	1	1	10	1
$PopulationSize$	50	100	50	100
Generations	500	250	50	240
Fitness evaluations	25,000	25,000	25,000	$\leq$ 25,000

Table 2: Probabilities for P_l and PDFs for α_l

	Instance 1	Instance 2	Instance 1 and 2	
Work centre (l)	P_l	P_l	α_l PDF	Mean (d)
1	0.00	0.00	—	—
2	0.10	0.30	Exponential	0.0847
3	0.25	0.45	Exponential	0.0935
4	0.15	0.35	Exponential	0.1338
5	0.00	0.00	—	—
6	0.00	0.00	—	—
7	0.00	0.00	—	—

Table 3: Descriptive statistics of average profits and average computation times per strategy in problem instance 1

	BS	IS	ES	HS
Mean (USD)	703,689	715,376	555,932	707,503
Minimum (USD)	550,417	484,229	460,416	550,014
Maximum (USD)	793,316	795,716	689,685	792,696
Std dev (USD)	55,539	60,416	47,322	60,754
Average computation time (s)	1256	1144	1235	1369

Table 4: Descriptive statistics of average profits and average computation times per strategy in problem instance 2

	BS	IS	ES	HS
Mean (USD)	707,600	703,420	543,140	723,460
Minimum (USD)	536,840	538,990	437,930	561,550
Maximum (USD)	785,810	783,040	656,800	790,460
Std dev (USD)	60,137	61,744	43,412	51,258
Average computation time (s)	1284	1108	1214	1265

No dominance (neither first nor second degree stochastic dominance) can be determined among the CDFs of average profit obtained under BS, IS and HS in problem instance 1, where all three strategies appear equally competitive. These results are in accordance with results from Mann-Whitney U test, which indicate stochastic homogeneity ($p > 0.05$) among samples obtained under BS, IS and HS in problem instance 1. It is interesting that population size (i.e. different setups of implicit averaging) has no significant effect in this setting as evidenced under BS and IS.

In problem instance 2, results from Mann-Whitney U tests also indicate stochastic homogeneity ($p > 0.05$) among samples obtained under BS, IS and HS. However, the CDFs of average profit obtained under HS dominates the CDFs of average profit obtained under IS and under BS, respectively (first and second-order stochastic dominance). These results indicate that, even in a setting with a limited evaluation budget, the accurate fitness estimates from explicit averaging can be beneficial to optimization. It is clear that the difference between IS and HS arises during the final stages of the optimization only, while IS continues optimization during additional generations, HS focuses resources on explicit averaging across its last population. When seen in combination with the poor performance of ES, our results suggest that the trade-off between improved exploration (from evaluating more individuals) and accurate fitness evaluations (through simulation replicates for the same individual) needs to be carefully balanced in this setting. This finding appears in line with previous research that has delivered contradictory results regarding the relative performance of explicit and implicit averaging.

Table 5: Values for Mann-Whitney U statistic obtained in problem instance 1

	BS	IS	HS	ES
BS	—	1525	1705	107**
IS	—	—	1662	117**
HS	—	—	—	113**
ES	—	—	—	—

** $p < 0.01$

Table 6: Values for Mann-Whitney U statistic obtained in problem instance 2

	BS	IS	HS	ES
BS	—	1719	1549	103**
IS	—	—	1485	96**
HS	—	—	—	36**
ES	—	—	—	—

** $p < 0.01$

6 Conclusion

Implicit averaging strategies reduce the impact of noise by having a sufficiently large population, which ensures that individuals from promising regions of the search space are sampled repeatedly [10, 17]. On the other hand, explicit averaging strategies use average fitness values, obtained across a specific number of fitness evaluations, to ensure that evaluation of individuals is based on fitness estimates that are more robust to noise [10].

One of the key findings of this study was that, in the context of our real-world problem, a noise-handling strategy based on explicit averaging did not provide a competitive performance. More generally, this points to the fact that the computational costs incurred by simulation replicates may be problematic in constrained time settings.

Furthermore, we found that implicit averaging performed robustly for both of the population sizes used. The performance of our hybrid strategy does indicate that some effort towards explicit averaging may become important with increasing variability. Under low levels of fitness variability, the hybrid strategy, implicit averaging and our baseline showed a comparable performance. This situation changed with increasing levels of fitness variability, when HS started to enhance overall performance.

Compared to a pure, implicit averaging strategy, the hybrid strategy misses out on the last few generations of optimization. Our results show, however, that this disadvantage is more than counter-balanced by the benefits from an accurate final selection step that reduces the likelihood of choosing an inferior individual (in terms of average fitness) as the final solution.

7 Limitations and Future Research

The relevance of obtaining more reliable fitness estimates increases with the level of variability, since there is a higher risk of choosing an inferior solution. It is, therefore, intuitive that the final selection mechanism implemented in HS would be more beneficial in such circumstances. But at the same time, the number of fitness evaluations needed to obtain reliable estimates is expected to raise with higher fitness variability, leaving to the evolutionary process a smaller share of the computational budget. Therefore, further research may focus on investigating the right trade-off between exploration (IS) and accurate fitness evaluation (ES). In this sense, the application of different sampling techniques (e.g., Latin Hypercube) during the final selection mechanism might be worthy of future investigation, as it may allow a reduction in the number of fitness evaluations required in this last step.

Our results underline issues around the computational cost of explicit averaging, but also highlight that sporadic use of this strategy may, nevertheless, be beneficial. In this study, the use of explicit averaging in the hybrid strategy was limited to the final selection step only. Future research

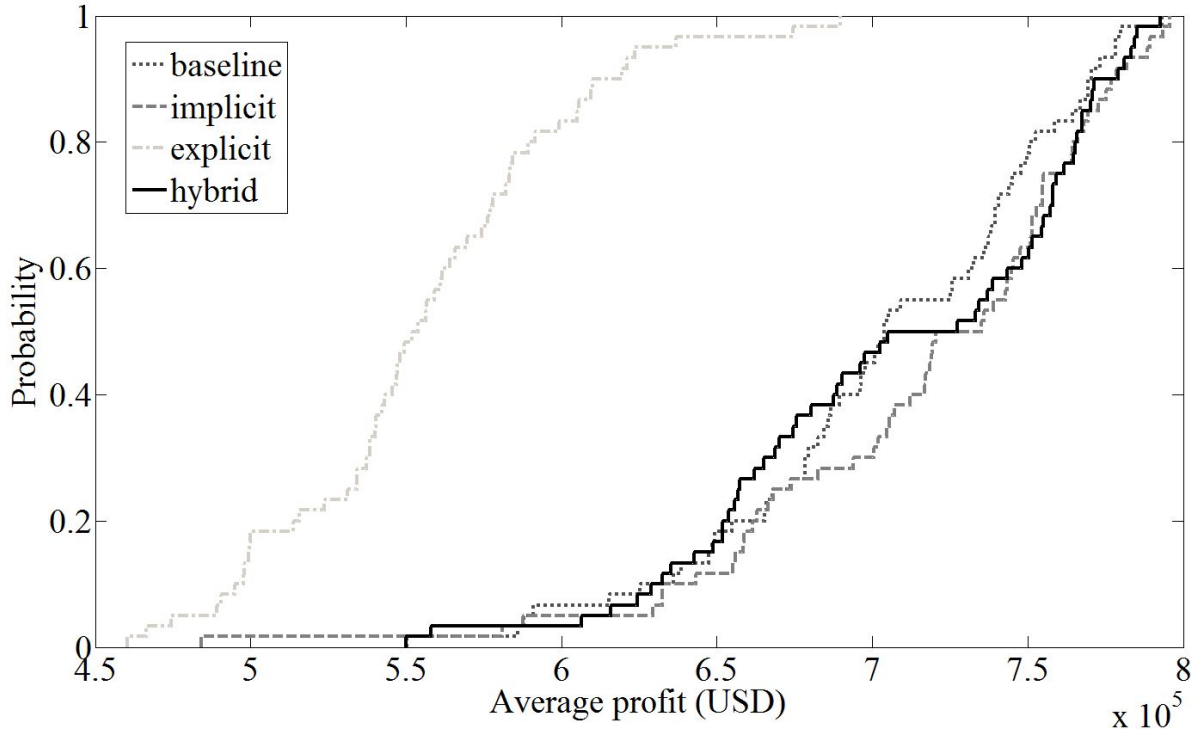


Figure 1: CDFs of average profit values obtained with production plans generated under the four different strategies in problem instance 1.

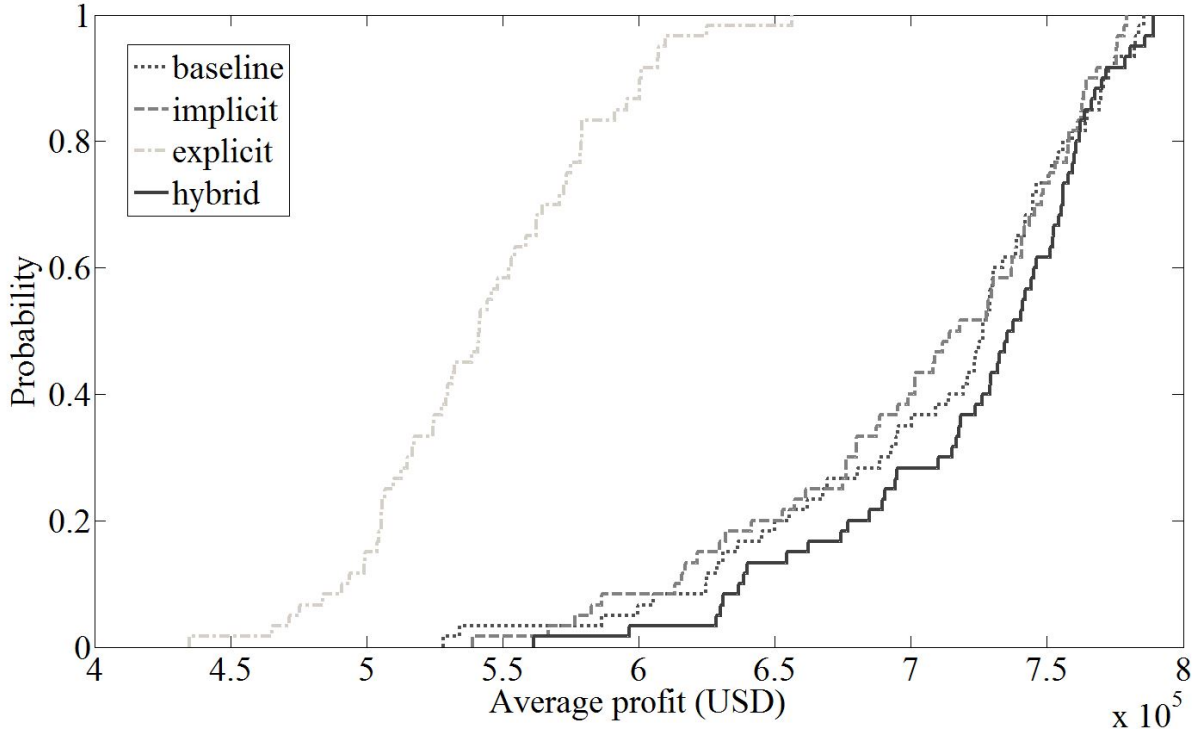


Figure 2: CDFs of average profit values obtained with production plans generated under the four different strategies in problem instance 2.

may consider the possibility of using explicit averaging at earlier points during the optimization process.

Acknowledgement

Juan Esteban Diaz expresses his gratitude to the Secretariat for Higher Education, Science and Technology of Ecuador, who has supported him with a doctoral scholarship.

References

[1] M. Bhattacharya, R. Islam, A. N. Mahmood (2014) Uncertainty and evolutionary optimization: A novel approach, *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 988–993.

[2] J. Branke, C. Schmidt (2003) Selection in the presence of noise, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 766–777.

[3] J. Branke, C. Schmidt, H. Schmec (2001) Efficient fitness estimation in noisy environments. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 243–250.

[4] K. Deb (2000) An efficient constraint handling method for genetic algorithms, *Computer methods in applied mechanics and engineering*, vol. 186, no. 2, pp. 311–338.

[5] K. Deep, K. P. Singh, M. Kansal, C.Mohan (2009) A real coded genetic algorithm for solving integer and mixed integer optimization problems, *Applied Mathematics and Computation*, vol. 212, no. 2, pp. 505–518.

[6] J. E. Diaz Leiva, J. Handl (2014) Simulation-based ga optimization for production planning, *Proceedings of the Student Workshop on Bioinspired Optimization Methods and their Applications (BIOMA)*, pp. 27–39.

[7] C. Ehrenberg, J. Zimmermann (2012) Simulation-based optimization in make-to-order production: scheduling for a special-purpose glass manufacturer, *Proceedings of the Winter Simulation Conference (WSC)*, pp. 1–12.

[8] G. Figueira, B. Almada-Lobo (2014) Hybrid simulation-optimization methods: A taxonomy and discussion, *Simulation Modelling Practice and Theory*, vol. 46, pp. 118–134.

[9] M.-R. Ghasemi, J. Ignatius, A. Emrouznejad (2014) A bi-objective weighted model for improving the discrimination power in MCDEA, *European Journal of Operational Research*, vol. 233. no. 3, pp. 640–650.

[10] Y. Jin, J. Branke (2005) Evolutionary optimization in uncertain environments - a survey, *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317.

[11] H. B. Mann, D. R. Whitney (1947) On a test of whether one of two random variables is stochastically larger than the other, *The annals of mathematical statistics*, vol. 18, no. 1, pp. 50–60.

[12] F. Neri, X. del Toro Garcia, G. L. Cascella, N. Salvatore (2008) Surrogate assisted local search in pmsm drive design, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 27, no. 3, pp. 573–592.

[13] D. W. Nordstokke, B. D. Zumbo (2010) A new nonparametric levene test for equal variances, *Psicológica*, vol. 31, no. 2, pp. 401–430.

[14] C. Qian, Y. Yu, Y. Jin, Z.-H. Zhou (2014) On the effectiveness of sampling for evolutionary optimization in noisy environments, *Lecture Notes in Computer Science*, vol. 8672, pp. 302–311.

[15] G. Rudolph (2001) A partial order approach to noisy fitness functions, *Proceedings of the Congress on Evolutionary Computation (CEC)*, vol. 1, pp. 318–325.

[16] Y. Sano, H. Kita, I. Kamihira, M. Yamaguchi (2000) Online optimization of an engine controller by means of a genetic algorithm using history of search, *Proceedings of the 26th IEEE Annual Conference of the Industrial Electronics Society (IECON)*, vol. 4, pp. 2929–2934.

[17] A. Syberfeldt, A. Ng, R. I. John, P. Moore (2010) Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling, *European Journal of Operational Research*, vol. 204, no. 3, pp. 533–544.

[18] S. Yitzhaki (1982) Stochastic dominance, mean variance, and gini's mean difference, *The American Economic Review*, vol. 72, no. 1, pp. 178–185.

Data Mining-Assisted Parameter Tuning of a Search Algorithm

Jurij Šilc
Computer Systems Department, Jožef Stefan Institute, Jamova cesta 39, SI-1000 Ljubljana, Slovenia
E-mail: jurij.silc@ijs.si

Katerina Taškova
Institute of Computer Science, Johannes Gutenberg University Mainz, Staudingerweg 9
55128 Mainz, Germany
E-mail: ktaskova@uni-mainz.de

Peter Korošec
Computer Systems Department, Jožef Stefan Institute, Jamova cesta 39, SI-1000 Ljubljana, Slovenia, and
Faculty of Mathematics, Science and Information Technologies, University of Primorska
Glagoljaška 8, SI-6000 Koper, Slovenia
E-mail: peter.korosec@ijs.si

Keywords: data mining, differential ant-stigmergy algorithm, low-discrepancy sequences, meta-heuristic optimization, parameter tuning

Received: December 1, 2014

The main purpose of this paper is to show a data mining-based approach to tackle the problem of tuning the performance of a meta-heuristic search algorithm with respect to its parameters. The operational behavior of typical meta-heuristic search algorithms is determined by a set of control parameters, which have to be fine-tuned in order to obtain a best performance for a given problem. The principle challenge here is how to provide meaningful settings for an algorithm, obtained as result of better insight in its behavior. In this context, we discuss the idea of learning a model of an algorithm behavior by data mining analysis of parameter tuning results. The study was conducted using the Differential Ant-Stigmergy Algorithm as an example meta-heuristic search algorithm.

Povzetek: Osnovni namen članka je pokazati, kako se lahko z uporabo tehnik podatkovnega rudarjenja lotevamo problema uglaševanja sposobnosti metahevristčnega iskalnega algoritma z vidika njegovih parametrov. Delovanje značilnega metahevrističnega iskalnega algoritma je določeno z naborom njegovih krmilnih parametrov, ki morajo biti za dosego najboljših sposobnosti pri danem problemu dobro uglašeni. Temeljni izziv je kako zagotoviti najboljšo nastavitev algoritma, ki bo rezultat vpogleda v njegovo vedenje. V zvezi s tem razpravljamo o ideji učenja modela za obnašanje algoritma na osnovi analize podatkovnega rudarjenja rezultatov uglaševanja njegovih parametrov. Študija je narejena z uporabo Diferencialnega algoritma s stigmergijo mravelj, kot primera metahevrističnega iskalnega algoritma.

1 Introduction

The research interest for meta-heuristic search algorithms has been significantly growing in the last 25 years as a result of their efficiency and effectiveness to solve large and complex problems across different domains [2]. The state-of-the-art nature-inspired meta-heuristic algorithms for high-dimensional continuous optimization include also algorithms inspired from the collective behavior of social organisms [14].

One such algorithm, which we will address in this paper, is the Differential Ant-Stigmergy Algorithm (DASA) initially proposed by Korošec [6], and further improved in [8]. DASA is inspired by the efficient self-organizing behavior of ant colonies emerging from a pheromone-mediated communication, known as stigmergy [3]. One of the first stigmergy-based algorithms designed for continuous function optimization was Multilevel Ant Stigmargy Algorithm [7].

Naturally, DASA can be classified within the Ant Colony Optimization (ACO) framework. However, the use of a continuous pheromone model in the form of Cauchy probability distribution with representation of the search space in the form of the so-called differential graph makes DASA dissimilar from the original ACO paradigm. The rationale behind DASA is in memorizing (via the pheromone model updates) the "move" in the search space that improves the current best solution and using it in further search. As it is the case with most of the meta-heuristic algorithms, the operational behavior of DASA is determined by a set of control parameters. In practice, these parameters have to be fine-tuned in order to obtain best performance of the algorithm. It can be quite inconvenient for the users as:

– they usually do not have insight into the behavior of the algorithm, and

– even if a default setting exists, it may not be adequate for a specific instance or type of a problem. Moreover, parameter tuning is computationally expensive task.

The principle challenge here is how to provide meaningful (default) settings for DASA, obtained as result of better insight into the algorithm's behavior. Furthermore, can we find optimal regions in DASA parameter space by analyzing the patterns in the algorithm's behavior with respect to the problem characteristics? Related to this, we discuss the preliminary findings based on data mining analysis of parameter tuning results. More precisely, the parameter tuning task is approached by two-step procedure that combines a kind of experimental design with data mining analysis.

We use Sobol' sequences [10] for even sampling of the algorithm parameter space to generate a large and diverse set of parameter settings. These are used as input to DASA to be tuned on a given function optimization problem. The performance of DASA on the given function optimization problem, in terms of function error, is captured at different time points for all sampled parameter settings. The data collected in the first step, DASA performance with corresponding parameter settings, is subject for intelligent data analysis, i.e., multi-target regression with Predictive Clustering Trees [1].

Parameter sampling combined with regression has been already used by Stoean et al. [11] for tuning metaheuristics: Latin hypercubes parameter sampling is combined with single-target regression with Support Vector Machines. Our approach modifies the former by replacing the Latin hypercube sampling by Sobol' sequences, as the former is best suited in the case when a single parameter dominates the algorithm's performance, while it should be used with care if there are interactions among the sampled parameters [9]. Moreover, we define the regression task as multi-target regression, taking into account more than one target (in this case the function error at few time points) with the goal to find parameter settings for the given algorithm that will not only solve the problem but will also solve the optimization problem fastest.

The reminder of this paper is structured as follows. Section 2 introduces the differential ant-stigmergy algorithm. Then, Section 3 addresses the parameter tuning task and Section 4 presents the experimental evaluation with the results. After that, Section 5 discusses the idea of post-hoc analysis of parameter tuning by data mining. Finally, Section 6 summarizes this study and outlines possible directions for further work.

2 The Differential Ant-Stigmergy Algorithm

The version of DASA used in our experimental evaluation is described in details by Korošec et al. [8] (see Figure 1).

DASA introduces the concept of variable offsets (referred as to parameter differences) for solving the continuous optimization problems. By utilizing discretized offsets of the real-valued problem parameters, the continuous optimization problem is transformed to a graph-search problem. More precisely, assuming a multidimensional parameter space with x_i being the current solution for the i-th parameter, we define new solutions x_i' as follow:

$$x_i' = x_i + \omega\delta_i, \tag{1}$$

where δ_i is called the parameter difference and is selected from the following set:

$$\Delta_i = \Delta_i^- \cup \{0\} \cup \Delta_i^+, \tag{2}$$

where

$$\Delta_i^- = \{\delta_{i,k}^-|\ \delta_{i,k}^- = -b^{k+L_i-1}, k = 1, 2, \ldots, d_i\} \tag{3}$$

and

$$\Delta_i^+ = \{\delta_{i,k}^+|\ \delta_{i,k}^+ = b^{k+L_i-1}, k = 1, 2, \ldots, d_i\}. \tag{4}$$

Here,

$$d_i = U_i - L_i + 1, \tag{5}$$

$$L_i = \lfloor \log_b(\epsilon_i) \rfloor, \tag{6}$$

$$U_i = \lfloor \log_b(\max(x_i) - \min(x_i)) \rfloor, \tag{7}$$

$i = 1, 2, \ldots, D$, D is dimension of the problem, b is the discretization base, ϵ is the maximal computer arithmetic's precision, and the weight $\omega = \texttt{Random\_Integer}(1, b - 1)$ is added to enable a more flexible movement over the search space.

In principle, DASA relies on two distinctive characteristics, differential graph and continuous pheromone model. Here, we will briefly discuss these two characteristics and outline the main loop of the DASA search process.

First, DASA transforms the D-dimensional optimization problem into a graph-search problem. The corresponding differential graph is a directed acyclic graph obtained by fine discretization of the continuous parameters' offsets. The graph has D layers with vertices, where each layer corresponds to a single parameter. Each vertex corresponds to a parameter offset value that defines a change from the current parameter value to the parameter value in the next search iteration. Furthermore, each vertex in a given layer is connected with all vertices in the next layer. The set of possible vertices for each parameter depends on the parameter's range, the discretization base and the maximal computer arithmetic's precision, which defines the minimal possible offset value. Ants use these parameters' offsets to navigate through the search space. At each search iteration, a single ant positioned at a certain layer moves to a specific vertex in the next layer, according to the amount of pheromone deposited in the graph vertices belonging to this layer.

1: $\vec{x}^{\text{tbest}} = \texttt{Rnd\_Solution}()$
2: $y^{\text{best}} = f(\vec{x}^{\text{tbest}})$
3: $y^{\text{tbest}} = \text{inf}$
4: $\mathcal{G} = \texttt{Graph\_Initialization}(b, \vec{L}, \vec{U}, \vec{\epsilon})$
5: $\texttt{Pheromone\_Initialization}(\mathcal{G})$
6: **while** terminating condition is not met **do**
7: $k = 0$
8: **for all** m ants **do**
9: **repeat**
10: $\vec{p}_i = \texttt{Find\_Path}(\mathcal{G})$ {path of the i-th ant}
11: $k = k + 1$
12: **if** $k > m^2$ **then**
13: $\vec{x}^{\text{tbest}} = \texttt{Rnd\_Solution}()$
14: $y^{\text{best}} = f(\vec{x}^{\text{tbest}})$
15: $\texttt{Pheromone\_Initialization}(\mathcal{G})$
16: **goto** line 7 {a local optimum was found, so the search process is restarted}
17: **end if**
18: **until** ($\vec{p}_i = \mathbf{0}$) {means of all parameters' offsets are 0}
19: $\omega = \texttt{Random\_Integer}(1, b - 1)$
20: $\vec{x}_i = \vec{x}^{\text{tbest}} + \omega\delta(\vec{p})$
21: **end for**{ants created solutions}
22: $y^{\text{cbest}} = \text{inf}$
23: **for all** m ants **do**
24: $y = f(\vec{x}_i)$ {function evaluation}
25: **if** $y < y^{\text{cbest}}$ **then**
26: $y^{\text{cbest}} = y$
27: $\vec{p}^{\text{cbest}} = \vec{p}_i$
28: $\vec{x}^{\text{cbest}} = \vec{x}_i$
29: **end if**
30: **end for** {created solutions were evaluated}
31: **if** $y^{\text{cbest}} < y^{\text{tbest}}$ **then**
32: $y^{\text{tbest}} = y^{\text{cbest}}$
33: $\vec{x}^{\text{tbest}} = \vec{x}^{\text{cbest}}$
34: $s = \texttt{Update\_Scales}(s_{\text{global}}, s_{\text{local}})$
35: $\texttt{Pheromone\_Redistribution}(\vec{p}^{\text{cbest}}, s)$
36: **if** $y^{\text{tbest}} < y^{\text{best}}$ **then**
37: $y^{\text{best}} = y^{\text{tbest}}$
38: $\vec{x}^{\text{best}} = \vec{x}^{\text{tbest}}$
39: **end if**
40: **else**
41: $\texttt{Update\_Scale}(s_{\text{global}})$
42: $\texttt{Pheromone\_Evaporation}(\mathcal{G}, \rho)$
43: **end if**
44: **end while**

Figure 1: The Differential Ant-Stigmergy Algorithm

Second, DASA performs pheromone-mediated search that involves best-solution-dependent pheromone distribution. The amount of pheromone is distributed over the vertices according to the Cauchy Probability Density Function (CPDF) [9]. DASA maintains a separate CPDF for each parameter. Initially, all CPDFs are identically defined by a location offset set to zero and a scaling factor set to one. As the search process progresses, the shape of the CPDFs changes: CPDFs shrink and stretch as the scaling factor decreases and increases, respectively, while the location offsets move towards the offsets associated with the better solutions. The search strategy is guided by three user-defined real positive factors: the global scale increase factor, s_+, the global scale decrease factor, s_-, and the pheromone evaporation factor, ρ. In general, these three factors define the balance between exploration and exploitation in the search space. They are used to calculate the values of the scaling factor and consequently influence the dispersion of the pheromone and the moves of the ants.

Finally, the main loop of DASA consists of an iterative improvement of a temporary-best solution, performed by searching appropriates paths in the differential graph. The search is carried out by m ants, all of which move simultaneously from a starting vertex to the ending vertex at the last level, resulting in m constructed paths. Based on the found paths, DASA generates and evaluates m new candidate solutions. The best among the m evaluated solutions is preserved and compared to the temporary-best solution. If it is better than the temporary-best solution, the latter is replaced, while the pheromone amount is redistributed along the path corresponding to the path of the preserved solution and the scale factor is accordingly modified to improve the convergence. If there is no improvement over the temporary-best solution, then the pheromone distributions stay centered along the path corresponding to the temporary-best solution, while their shape shrinks in order to enhance the exploitation of the search space. If for some fixed number of tries all the ants only find paths composed of zero-valued offsets, the search process is restarted by randomly selecting a new temporary-best solution and re-initializing the pheromone distributions.

3 Parameter Tuning

To obtain the best possible performance on a given problem, one should consider a task specific tuning of the parameter setting for the optimization algorithm used. Determining the optimal parameters is an optimization task in itself, which is extremely computationally expensive. There are two common approaches for choosing parameters values: parameter tuning and parameter control. The first approach selects the parameter settings before running the optimization algorithm (and they remain fixed while performing the optimization). The second approach optimizes the algorithm's parameters along with the problem's parameters. Here, we will focus on the first approach, parameter tuning.

A detailed discussion and survey of parameter tuning methods is given by Eiben and Smit [4]. According to this survey, one way to approach parameter tuning is by sampling methods. Sampling methods reduce the search effort by decreasing the number of investigated parameter settings as compared to the full factorial design: the basic full factorial design investigates 2^k parameter settings, subject to k parameters, each of which have 2 possible values; in the more general case, parameters can have arbitrary number of values; moreover, an increase in the number of investigated parameters means an exponential increase in the number of parameter settings to be tested. Two widely used sampling methods are Latin-squares [9] and Taguchi orthogonal arrays [12]. However, these are not the most robust sampling techniques, e.g., Latin-squares or Latin hypercube sampling is good in the case where one of the parameters dominates the algorithm's performance, while it should be used with care if there are interactions among the parameters.

Ultimately, we would like to find a sampling schema that will be able to detect the interactions among the parameters, will be independent from user-specified information regarding the particular parameter values to be considered (typical for factorial design), and will deliver small but representative sample of the parameter search space. The first two requirements are satisfied by the pure random sampling, but the last is not, as random sampling does not guarantee that the sampled values are evenly spread across the entire domain. The so-called low-discrepancy sequences were specially designed to fulfill all three requirements. Therefore, Sobol' sequences, a representative variation of low-discrepancy sequences introduced by Sobol' [10], was considered for sampling the parameter space of DASA in this study.

Sobol' sequences, sampled from a D-dimensional unit search space, are quasi-random sequences of D-tuples that are more uniformly distributed than uncorrelated random sequences of D-tuples. These sequences are neither random nor pseudo-random: they are cleverly generated not to be serially uncorrelated by taking into account which tuples in the search space have already been sampled. For a detailed explanation and overview of the schemas for generating Sobol' sequences, we refer to [9]. The particular implementation of Sobol' sampling used in our analysis is based on the Gray code order [5].

4 Experimental Evaluation

Since data mining methods can only discover patterns actually present in the data, the dataset subject to analysis must be large enough and informative enough to contain these patterns, i.e., to describe different types of algorithm's behavior. For this reason, we decided to use a simple test function, which matched this requirement and was used for building an example case model.

Table 1: Parameter settings for DASA* and DASA°

Algorithm	DASA°		DASA*	
Parameter	$D = 20$	$D = 40$	$D = 20$	$D = 40$
m	10	10	5	7
ρ	0.2	0.2	0.324	0.388
s_+	0.02	0.02	0.201	0.136
s_-	0.01	0.01	0.289	0.344
b	10	10	6	8

Table 2: Median values of the function errors for the Sphere function

Algorithm	DASA°		DASA*	
FEs	$D = 20$	$D = 40$	$D = 20$	$D = 40$
$25 \times D$	16.7	18.3	2.53	9.31
$250 \times D$	0.0003	0.0021	0	0
$2500 \times D$	0	0	0	0
$25000 \times D$	0	0	0	0

Therefore, the performance of DASA was evaluated on the Sphere function:

$$f(x) = |z|^2 + f(x_{\text{opt}}), \tag{8}$$

where $z = x - x_{\text{opt}}$ and x_{opt} is optimal solution vector, such that $f(x_{\text{opt}})$ is minimal. Function $f(x)$ is defined over D-dimensional real-valued search space x and is scalable with the dimension D. It has no specific value of its optimal solution (it is randomly shifted in x-space) and has an artificially chosen optimal function value (it is randomly shifted in f-space). In this study, we considered the Sphere function with respect to two dimensions, $D = 20$ and $D = 40$.

The performance of DASA is dependent on the values of five parameters: three real-valued parameters that directly influence the search heuristic (s_+, $s-$, and ρ) and two integer-valued parameters (m and b). Therefore, we considered all of them for tuning DASA performance on the Sphere function for both search space dimensions, $D = 20$ and $D = 40$. Using the Gray-code-based Sobol' generator we generated 5000 parameter settings (5-tuples). Note that the Sobol' sampling generates numbers on the unit interval: in order to obtain the true parameter settings, we mapped these values on the predefined search range of parameter values. The latter for each of the five tuned parameters was defined as follows: $4 \leq m \leq 200$, $0 \leq \rho \leq 1$, $0 \leq s_+ \leq 1$, $0 \leq s_- \leq \rho$, and $2 \leq b \leq 100$. Moreover, the mapped values for the integer-valued parameters m and b were rounded to the closest integer value. Finally, due to implementation reasons, the upper bound of the global scale decrease factor s_- was actually limited by the value of the evaporation factor ρ.

In the next step, the performance of the Sobol' sampled parameter settings were tested on the Sphere benchmark function. Due to the stochastic nature of DASA, every parameter setting was used in a multiple-run experimental evaluation. Each run included $25000 \times D$ function evaluations (FEs). The number of runs was set to 15. The results gathered by the parameter tuning process are most often subjected to ordinal data analysis, which includes ranking of the different sampled parameter sets according to some calculated statistics, e.g., best or mean performance of the algorithm in some predefined number of runs [13]. In this case, performance of the algorithm is expressed in terms of the function error, i.e., the difference between the obtained and optimal function value. In order to find a setting that will be satisfactory in terms of convergence speed, we captured the error values at four different time points, corresponding to $25 \times D$, $250 \times D$, $2500 \times D$, and $25000 \times D$ FEs.

The optimal performing parameter setting was chosen based on the median best performance over all runs aggregated over all time points for a given dimension ($D = 20$ and $D = 40$). A common approach is to use the mean performance, but we took the median in order to avoid the problems that the mean has when observing large variance in the function values across the runs. More precisely, given a function, an individual rank is assigned to every setting (out of 5000) for the four time points. A single final rank is calculated by ranking the sum of the four individual rankings assigned to the parameter settings. The best-ranked parameter setting for a given dimension defines instance of DASA referred to as DASA*.

The results of DASA tuning subject to ordinal data analysis are presented in Tables 1 and 2. Table 1 reports the tuned parameter settings for both DASA* instances.

In addition, the default parameter setting for DASA from [8] is given as a reference for comparison. The corresponding instance is referred to as DASA°.

Results in Table 2 represent the median values of the function errors, at the four time points, obtained by DASA* and DASA° instances for both dimensions. Note, that error value below 10^{-8} was treated as zero. The table clearly shows that DASA* is better than DASA°.

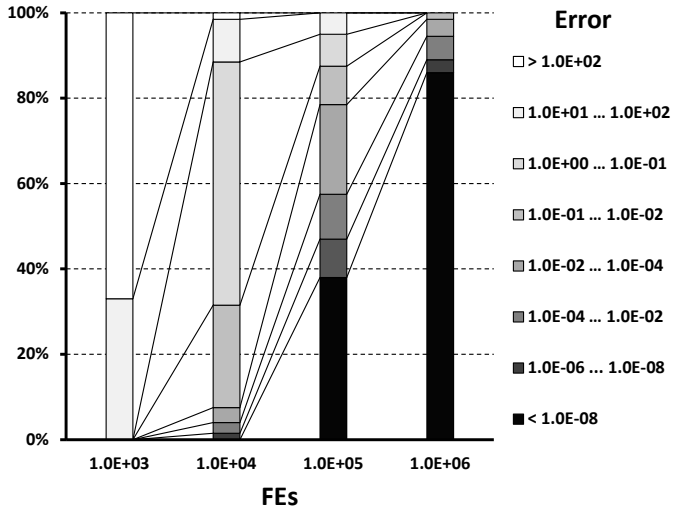


Figure 2: Median error distributions for the Sphere function in the case of $D = 40$.

5 Data Mining Analysis

Parameter tuning of an algorithm leads to a better performance, however it is a computationally expensive and problem-dependent task. Considering this, the idea is to extend the simple tuning that delivers a single parameter set and analyze the gathered data in an intelligent way. The intelligent analysis can extract patterns (regularities) in the explored parameter space that define a specific behavior of DASA. To this end, data mining methods for automated discovery of patterns in data can be used. As data mining methods can only discover patterns that are present in the data, the dataset subject to analysis must be large enough and informative enough to contain these patterns, i.e., to describe different types of algorithm's behavior. Related to this, we considered a data mining approach on a representative example, i.e., error model of the Sphere function.

To begin with, consider the graph in Figure 2 that visualizes the Sphere function error distribution. The graph depicts the distribution of the median error values obtained by 5000 parameter settings at four different points for $D = 40$. As we are more concerned with the practical significance between a large and a small error value than the statistical significant difference between two actual error values, the error values are discretized into nine intervals, each of which is represented with a color chosen according to the error magnitude between black (error below 10^{-8}) and white (error above 10^2). The graph clearly shows that the sampled settings determine different DASA performance. As evident, there is a big cluster of parameter settings that solve this function to the aimed accuracy (error below 10^{-8}) in the given time budget (10^6 FEs). Moreover, subset of this cluster solves the function for an order of magnitude less FEs. Our aim, therefore, is to find a (common) description of this cluster, in terms of DASA parameter relations, that represents a good behavior of DASA (as well as what parameter relations lead to a bad DASA performance).

For this purpose, we formulated the problem as a predictive modeling task using decision trees to model the function error values in terms of the values of DASA parameters. Since the function error variables are continuous, the task at hand is a regression task. Furthermore, as our goal is to model the behavior of DASA at all time points simultaneously, the problem at hand is then a multi-target regression. To this end, we used Predictive Clustering Trees (PCTs) which are implemented in Clus system [1]. In this case, the median error at the four time points define the four target attributes considered for modeling (with the PCT) in dependence from the descriptive attributes, i.e., the function dimension and the five DASA parameters. The resulting dataset is composed of 10002 rows described with the 5001 parameter settings (including the default setting) of DASA applied to the two different dimensions, and 10 columns corresponding to the 10 attributes.

Figure 3 presents a PCT model for the Sphere function error. Each internal node of the tree contains a test on a descriptive attribute, while the leaves store the model predictions for the function error, i.e., a tuple of four values. The predictions are calculated as the mean values of the corresponding error values for the data instances belonging to the particular leaf (represented by a box). In fact, each leaf identifies one cluster of data instances (the size of the cluster is the value in the small box). The predictive performance of the model was assessed with 10-cross-validation. Note that this particular model was learned on the complete dataset subject to constraints on the maximal tree size of 25 nodes. We did this because the original model contained 1643 nodes (of which 822 leaves) and despite its better predictive performance, both training and testing, it was not comprehensible; aiming for an explanatory model, small and comprehensible, we considered the smaller tree obtained with the limitation of the size. The predictive performance of both models in terms of Root Relative Mean Squared Error (RRMSE) and Pearson Correlation Coefficient (PCC) are given in Table 3. Note that RRMSE represents the relative error with respect to the mean predictor performance, while PCC represents the linear correlation between the data and the model predictions. Good models have RRMSE values closer to 0 and PCC closer to 1.

The model in Figure 3 outlines 13 clusters of data instances, of which two (depicted with light-gray boxes) represent a good DASA performance. According to this model, the number of ants, m, is the most important DASA parameter for its performance on the Sphere function. More precisely, if $m > 83$, independent of the values of the other parameters, DASA solves the 20-dimensional functional problem for the given time budget. Moreover, if $m \leq 83$ another DASA parameters become important as well. For example, if $43 < m \leq 83$ and $s_+ > 0.009$ and $D = 20$ then DASA solves the function with error 3×10^{-6}, while the pattern $m \leq 43$ and $s_{\leq}0.040$ and $s_- > 0.656$ describes a poor DASA performance regard-

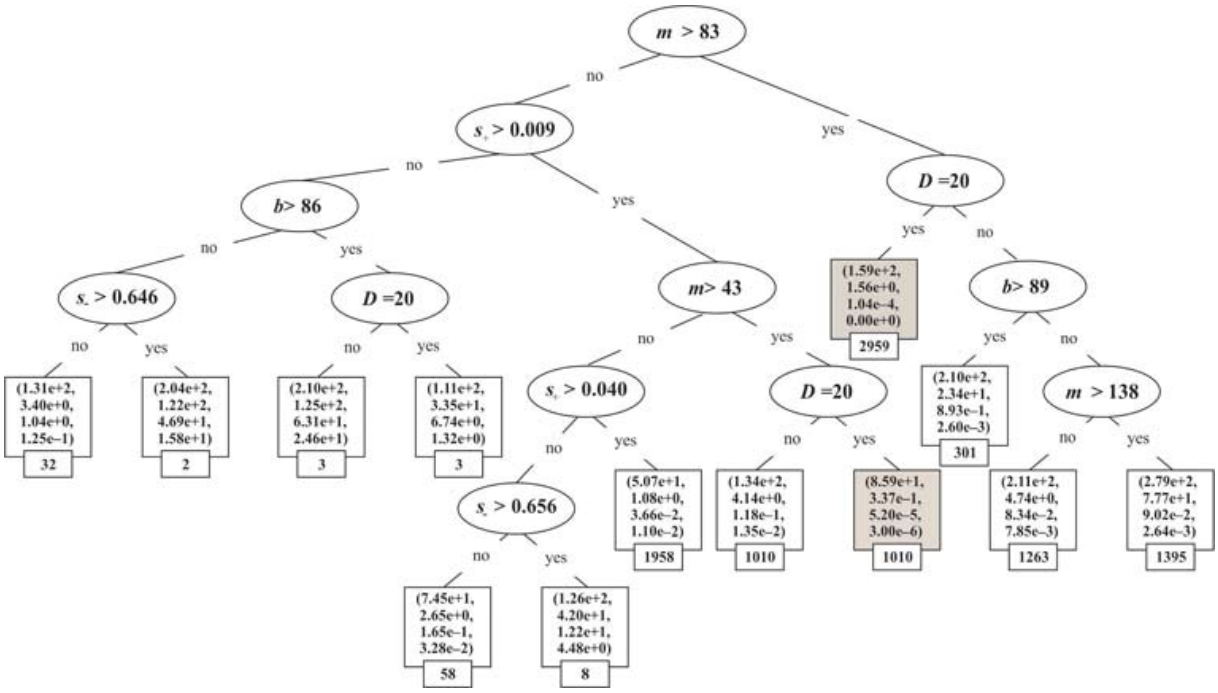


Figure 3: Predictive clustering tree representation of the error model for the Sphere function.

less of the function dimension. An interesting fact is that, the evaporation factor is not essential for DASA performance on the Sphere function. Moreover, the model also shows that is more difficult to describe the behavior of DASA for the 40-dimensional function problem than the 20-dimensional one.

Finally, note that the training performance (learned on the complete dataset) of the model in terms of the error and the correlation coefficient is best for the first target, while it gets worse with respect to the other three targets (see Table 3). This is especially significant if we take into account the testing performance of the model estimated with 10-cross-validation. However, the training performance is acceptable in our case, as we are interested in understanding the behavior of DASA and not aiming to obtain a model for prediction.

6 Conclusion

The principle challenge of meta-heuristic design is providing a default algorithm configuration, in terms of parameter setting, that will perform reasonably well in general (problem) case. However, while it is a good initial choice, the default algorithm configuration may result in low quality solutions on a specific optimization problem. In practice, the algorithms parameters have to be fine-tuned in order to obtain best algorithm's performance for the problem at hand, leading to the computational expensive task of parameter tuning. So, if the tuning task is unavoidable, the question is: can we use the results from the parameter tuning to extract some knowledge about the algorithm's behavior?

Related to this, the study focused on the problem of tuning the performance of the Differential Ant-Stigmergy Algorithm (DASA) with respect to its parameters. As it is the case with most of the meta-heuristic algorithms, the operational behavior of DASA is determined by a set (five) of control parameters. The existing default setting of DASA parameters [8] is obtained by experimentation with both real and benchmark optimization problems, but not as a result of some systematic evaluation. Furthermore, there is no deeper understanding of the impact of a particular parameter or parameters relations on the performance of DASA. In this context, we performed a systematic evaluation of DASA performance obtained by solving the Sphere function optimization problem with 5000 Sobol' sampled DASA parameter settings regarding two dimensions, 20 and 40.

Furthermore, we discussed the idea of learning a model of DASA behavior by data mining analysis of the parameter tuning results. In this context, we formulated the problem as multi-target regression and applied predictive clustering trees for learning a model of DASA behavior with respect to the function error performance. The obtained model revealed that the parameter denoting number of ants is the most important parameter for DASA performance on the 20-dimensional function problem. On the other hand, the evaporation factor is not essential for DASA performance on the Sphere function.

Further work will focus on additional experimental evaluation and data mining analysis of data with respect to more complex functions problems. This idea can be fur-

Table 3: Model performance with respect to RRMSE and PCC

Measure	RRMSE		PCC	
Training	1643 nodes	25 nodes	1643 nodes	25 nodes
$25 \times D$	0.166	0.408	0.986	0.913
$250 \times D$	0.373	0.679	0.928	0.734
$2500 \times D$	0.487	0.562	0.873	0.827
$25000 \times D$	0.472	0.546	0.882	0.837
Mean	0.396	0.557	0.843	0.689
Testing	1643 nodes	25 nodes	1643 nodes	25 nodes
$25 \times D$	0.219	0.450	0.976	0.893
$250 \times D$	0.638	0.817	0.777	0.584
$2500 \times D$	1.019	1.039	0.304	0.246
$25000 \times D$	1.053	1.055	0.234	0.218
Mean	0.806	0.875	0.426	0.312

ther extended to building models of DASA behavior that will include the optimization problem characteristics (such as multimodality, separability, and ill-conditioning) as descriptive attributes as well. The latter can provide insights on how to configure DASA performance with respect to the type of the optimization problem. Moreover, these insights can serve as a valuable information for improvement of DASA design.

References

[1] H. Blockeel, J. Struyf (2002) Efficient Algorithms for Decision Tree Cross-validation, *Journal of Machine Learning Research*, vol. 3, pp. 621–650.

[2] C. Blum, A. Roli (2003) Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308.

[3] E. Bonabeau, M. Dorigo, G. Theraulaz (1999) *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press.

[4] A. E. Eiben, S. K. Smit (2011) Parameter Tuning for Configuring and Analyzing Evolutionary Algorithms, *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31.

[5] S. Joe, F. Y. Kuo (2008) Constructing Sobol Sequences with Better Two-dimensional Projections, *SIAM Journal on Scientific Computing*, vol. 30, no. 5, pp. 2635–2654.

[6] P. Korošec (2006) Stigmergy as an Approach to Metaheuristic Optimization, Ph.D. dissertation, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

[7] P. Korošec, J. Šilc (2008) Using Stigmergy to Solve Numerical Optimization Problems, *Computing and Informatics*, vol. 27, no. 3, pp. 341–402.

[8] P. Korošec, J. Šilc, B. Filipič (2012) The Differential Ant-stigmergy Algorithm, *Information Sciences*, vol. 192, no. 1, pp. 82–97.

[9] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery (1992) *Numerical Recipes*, Cambridge University Press.

[10] I. M. Sobol' (1967) Distribution of Points in a Cube and Approximate Evaluation of Integrals, *USSR Compututational Mathematics and Mathematical Physics*, vol. 7, no. 4, pp. 86–112.

[11] R. Stoean, T. Bartz-Beielstein, M. Preuss, C. Stoean (2009) A Support Vector Machine-Inspired Evolutionary Approach for Parameter Setting in Metaheuristics, CIOP Technical report 01/09, Faculty of Computer Science and Engineering Science, Cologne University of Applied Science, Germany.

[12] G. Taguchi, T. Yokoyama (1993) *Taguchi Methods: Design of Experiments*, ASI Press.

[13] E.-G. Talbi (2009) *Metaheuristics: From Design to Implementation*, John Wiley & Sons.

[14] X.-S. Yang (2008) *Nature-Inspired Metaheuristic Algorithms*, Luniver Press.

A High Resolution Clique-based Overlapping Community Detection Algorithm for Small-world Networks

András Bóta
University of Szeged, Institute of Informatics Address,
P. O. Box 652., 6701 Szeged, Hungary
E-mail: bandras@inf.u-szeged.hu

Miklós Krész
University of Szeged, Juhász Gyula Faculty of Education,
Boldogasszony bvd. 6, 6720 Szeged, Hungary
E-mail: kresz@jgypk.u-szeged.hu

Keywords: network science, community detection, overlapping communities

Received: June 24, 2013

In this paper we propose a clique-based high-resolution overlapping community detection algorithm. The hub percolation method is able to find a large number of highly overlapping communities. Using different hub-selection strategies and parametrization we are able to fine tune the resolution of the algorithm. We also propose a weighted hub-selection strategy, allowing the algorithm to handle weighted networks in a natural way, without additional filtering. We will evaluate our method on various benchmarks, and we will also demonstrate the usefulness of our algorithm on a real-life economic case-study.

Povzetek: Predstavljena je nova hevristika za reševanje evklidskega BDMST problema. Primerjalni testi pokažejo prednosti pred obstoječimi metodami.

1 Introduction

One of the landmarks in graph theory was the introduction of small-world networks by Watts and Strogatz [31]. They have observed, that in real-life networks, the typical distance between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes in the network. Since then, several other properties of real-life networks was discovered. The degree distribution of these networks follows a power-law [2], and the edge distribution is not only globally, but also locally inhomogeneous. This latter feature is called community structure [11]. The goal of community detection is the discovery of this structure. While the phenomenon of communities is well observed, an exact definition is difficult to find.

In recent years, a large number of community detection algorithms have been proposed. Most of these consider communities to be disjoint vertex sets, and adopt the following intuition: They are looking for a partitioning of the nodes, which maximizes the number of edges between the nodes inside the sets, and minimizes them between the sets. It is also a goal to find meaningful communities, i.e. they discard trivial solutions of the problem (like a single community containing all of the vertices). Newman proposed modularity [27] as an efficient way to measure the goodness of disjoint communities. A comprehensive review of community detection can be found in [10].

The traditional definition of community allows disjoint vertex sets only. Based on the observation that in real-life networks, nodes can belong to multiple communities, Palla et al. introduced the concept of overlapping community detection and proposed the clique percolation method [29] as a solution. The idea of finding maximal cliques and joining them according to some criteria is the basis of several overlapping community detection algorithms [17, 21]. Other approaches are based on block models [12, 7], edge clustering [9, 1], label propagation [13] or optimization according to some fitness function [25, 20].

Measuring the goodness of overlapping community detection algorithms is complicated, since there is no agreement on the definition of an overlapping community. The specifications of different applications depend mainly on the ratio of overlaps between communities: several approaches require only a loose relaxation of the original "non-overlapping" definition in such way that occurence of nodes belonging to multiple communities is strongly restricted [13]; other concepts prefer highly overlapping community structure [20]. The resolution of the methods are closely tied to the ratio of overlaps. A highly overlapping community structure is often associated with a large number of relatively small communities, however the opposite is not always true. Hierarchical or multiresolutional methods combine these approaches.

Corresponding to this, the output of the above mentioned algorithms can be fundamentally different. There are basically two types of evaluation in the literature: one can use some kind of benchmark network like in [18, 22, 26, 32], and compare the results to the already known community

structure of the network [14, 28]. Another option is using a real-life application as an example, similar to a case-study.

In this paper, the authors propose the hub percolation overlapping community detection method. A node, that is a member of many adjacent cliques is considered more important. We refer to these nodes as hubs. We expand and join cliques if they contain the same hubs. One of the advantages of this method is, that both the hub selection and the joining criteria is adjustable. This allows us to discover different kinds of community structures from large, loosely overlapping groups to ones with a dense, highly overlapping structure. We also propose a hub-selection strategy able to handle weighted networks in a natural way without the need for filtering or pruning edges. Finally, we will rely on the framework proposed by Pluhár et al. in [3], and show how several popular algorithms can be represented in it.

We will use well-known benchmark networks [29, 26, 11] to demonstrate the difference between hub selection strategies in terms of community sizes, the size of overlaps and the number of singletons: nodes without communities. Then we will evaluate the performance of our method in two different ways. We will use the community based graph generator of Lancichinetti and Fortunato [18] to compare the results of our method to the OSLOM algorithm of the same authors [20], the COPRA method of Gregory [13], and the clique percolation method of Palla et al. [29]. We will also present a case study: we will examine the communities of an economic network constructed from the Hungarian company register. We will focus our attention on three aspects of the companies: the geographical location of them, the industrial sector they belong to and the age of the companies.

2 General framework

The authors of [3] described a general framework for overlapping community detection. In this section we summarize their approach. Here, and throughout the paper, by a graph G we mean an undirected simple graph with vertex (or node) set $V(G)$ and edge set $E(G)$. Edges might have arbitrary weight.

According to the framework introduced in [3], most community detection algorithms consist of two phases. Taking an input graph G, the first phase constructs a hypergraph $\mathcal{F} = (V, \mathcal{H})$, where $V(\mathcal{F}) = V(G)$, and $\mathcal{H} \subset 2^V$. The elements of $\mathcal{H}$ are considered the building blocks of communities. The second phase adds a distance function d to set $\mathcal{H}$, creating a metric space $\mathcal{M} = (\mathcal{H}, d)$. Using function d, a clustering algorithm creates a set of clusters $\mathcal{C}$. Finally, the arising clusters are associated to the subsets of V such that $K_i = \cup_{H \in C_i \in \mathcal{C}} H$, where K_i, the ith community corresponds to C_i, the ith cluster and K_i is just the union of the vertex set of those hyperedges that belong to C_i.

It is easy to show, that this framework applies to most community detection algorithms. In the case of the clique percolation method [29], $\mathcal{H}$ contains the k-cliques[1] of the original graph, and function d is:

$$d(K_i, K_j) = \begin{cases} 1, & \text{if } |K_i \cap K_j| = k - 1, \\ \infty, & \text{otherwise} \end{cases}$$

In the same paper [3], the authors have proposed the N^{++} community detection algorithm with a general distance function, where $\mathcal{H}$ is the same as above and $d(K_i, K_j) = 1$ only if $|K_i \cap K_j| \geq c$, where c is a parameter of the algorithm. In other cases $d(K_i, K_j) = \infty$. This method has proven its usefulness in applications [6, 16].

It is also possible to describe non-clique based methods using this formulation. In the case of COPRA [13], each element of $\mathcal{H}$ initially only contains one unique vertex $v \in V(G)$. In the second phase, these are joined according to a belonging coefficient. A threshold is introduced to provide a lower bound for community membership.

3 The hub percolation method

The motivation for creating an advanced community detection algorithm came from our previous work with the general framework of community detection [3]. Our aim was to create a flexible clique-based method taking into consideration our experiences with the clique percolation method [29] and the N^{++} method [3]. Much of the details of the algorithm described in this section comes from experiences gained during test runs on well-known benchmark networks like [32, 29, 26, 22].

The hub percolation method has two simple ideas at its core. A natural property of most approaches for overlapping community detection[2] is that cliques (fully connected subgraphs) are considered to be the purest communities. Therefore our method uses cliques at the beginning of the building process. An important observation on real-life networks is, that inside a community some members are more important than others with respect to the role of the nodes in connecting different communities. We will denote these nodes as hubs. In the building process the cliques of the graph are extended according to a limited percolation rule: two k-cliques are joined if they share $k - 1$ vertices. As a result of this process, the set of extended cliques consists of the building blocks of community detection. The joining phase of our method merges these extended cliques if they share the same hubs. Considering these ideas, an outline of the hub percolation algorithm is as follows:

1. Find the set C of all maximal cliques of size greater or equal than 3 on graph G.
2. Select the set of hubs H.
3. Create the set of extended cliques C'.

[1] Fully connected subgraphs containing exactly k nodes.

[2] In the following chapters, we will refer to overlapping community detection simply as community detection.

4. Compute the set of communities K: Take the union of extended cliques if one of them contains all the hubs in the other one.

Finding the set of all maximal cliques in a graph is a well-studied NP-hard problem of graph theory. Unfortunately an n-vertex arbitrary graph may contain i$3^{n/3}$ maximal cliques in the worst case [24]. Because of their unique structure, this number is significantly lower in small world networks allowing algorithms like in [30, 4, 8] to list the set of maximal cliques in reasonable time even for large networks. In this work we used the modified Bron-Kerbosch algorithm described in [8].

The hub selection strategy is an important part of the algorithm. Hubs represent the locally important nodes in the network. As a consequence, whether a node is a hub should depend on the t-neighborhood[3] of the given node, where t is a small number. In our interpretation hubs connect communities, therefore the deciding factor in hub selection should be the number of cliques the vertex belongs to. Each node v is assigned a hub value h_v according to the above rule, then some of them are selected if their value is higher than the average or median hub values in their t-neighborhood. It is also possible to extend the selection strategy to weighted networks. We will discuss hub selection in the next subsection.

In our method, cliques of the network are extended with a a one-step percolation rule, then merged if they share the same hubs. Introducing the filtering parameter $k \geq 2$, let us consider all cliques of size equal to k on the subgraph induced by the set of hubs H. We will denote the set of k-cliques on $G[H]$ as C_H. Then, we expand the elements of C_H according to a one-step percolation rule. Let C_e denote the set of merged cliques $c_e = c_H \cup c_0 \cup \cdots \cup c_\ell$ with $c_H \in C_H$, $c_0, \ldots, c_\ell \in C$ and $|c_0 \cap c_H| \geq 2, \ldots, |c_\ell \cap c_H| \geq 2$ [4].

The last step of our method corresponds to the joining phase of the community detection framework. We merge elementary communities if they contain the same hubs, more precisely, we take the union of two elementary communities c_{e_0} and c_{e_1} if $c_{e_0} \cap H \subseteq c_{e_1} \cap H$. We iterate this process by adding the new merged clique to C_e and removing the original ones. At the end of the process C_e contains the communities of graph G. Note, that depending on the hub selection strategy C_e may contain duplicate members, the merging process eliminates this problem as well. Each element of C_e is a union of the cliques of G and contains at least k hubs. We will refer to the members of C_e as elementary communities.

The hub percolation method
Input : Graph G, parameter k

1. Find all maximal cliques of graph G using any exact algorithm or heuristic. Let C denote the set of cliques.
2. For all $v \in V(G)$, let $h_v = |H_v|$, $H_v = \{h | v \in h, h \in C\}$.
3. Select the set of hubs H according to the hub selection strategy.
4. Let C_H denote the set of k-cliques on the subgraph induced by the hubs $G[H]$.
5. Create the set of extended cliques C_e according to the following rule: for all $c_H \in C_H$ find all cliques $c \in C$ where $|c \cap c_H| \geq 2$. Let $c_0, \ldots, c_\ell$ denote the cliques satisfying this criterion. Create the union of cliques $c_e = c_H \cup c_0 \cup \cdots \cup c_\ell$, and add c_e to C_e.
6. For all $c_{e_0}, c_{e_1} \in C_e$ add the union of them to C_e if $c_{e_0} \cap H \subseteq c_{e_1} \cap H$, and remove c_{e_0} and c_{e_1} from C_e. Iterate until there are no more merges.
7. The set C_e contains the communities of graph G.

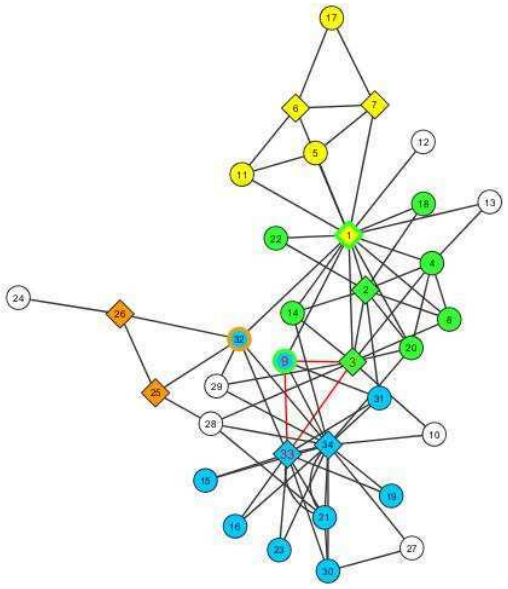


Figure 1: The communities of Zachary's karate club network [32]. Hubs are marked as diamond shapes. Nodes with multiple colors indicate overlapping nodes. The median hub selection strategy was used with $k = 2$. Nodes $9, 3, 33$ form an additional community and node 9 belongs to three communities.

It is easy to see, that the general framework proposed in applies to the hub percolation method. The edges of the hypergraph correspond to the extended communities in C_e, while the distance function is

$$d(K_i, K_j) = \begin{cases} 1, & \text{if } K_i \cap H \subseteq K_j \cap H \text{ or} \\ & K_j \cap H \subseteq K_i \cap H, \\ \infty & \text{otherwise} \end{cases}$$

[3] A t-neighbothood of a vertex v is the set of vertices N_v^t, where $u \in N_v^t$ only if the length of the shortest path from u to v is less or equal then t.

[4] Our experiances on various benchmark networks indicate that this value gives the best performance.

The community structure of Zachary's karate club network [32] can be seen on Figure 1. This network is a well-known social network, that represents friendships between the members of the club. Our method identifies five communities[5], the most interesting ones being the green and blue ones, as well as the one represented by the red triangle. During Zachary's observation the club split into two parts, because some friendships were broken. Most community detection algorithms are able to identify these subgroups even before the actual split. In our case the borders of the green and blue communities represent the borders between the two subgroups, and the red group identifies the edge that was broken when the split occurred.

3.1 Hub selection strategies

Hub selection is a crucial part of the algorithm. As we have mentioned before, each node in the graph is assigned a hub value based on the number of cliques it belongs to. Based on this value the selection strategy chooses the set of hubs H. Hubs represent "locally important" nodes so the criterion of the hub property of nodes or "hubness" should depend only on the tight neighborhood of the node. In our interpretation, this criterion depends on some simple statistical property of the first or second neighborhood of the given node, namely the average or median of neighboring hub values.

At the beginning of our work on several famous benchmark networks [26, 32] we have quickly found out, that the 2-neighborhood strategies are often not robust enough to select the appropriate hubs: hubs were relatively rare, which resulted in small overlaps and a larger than acceptable number of nodes without community memberships. A general experience was, that hubs should be "common enough", so that most of the nodes have one or more in their direct neighborhood.

Considering this, the 1-neighborhood median selection strategy provided the best community structure on these benchmark networks. This may not be the case, however, with other real-life networks. In order to extend our algorithm to handle different kinds of networks, we can generalize suggest another hub selection rule. Still considering only the direct neighborhood of nodes, we calculate the average hub value and multiply it with a parameter $q > 0$. If the hub value of the node is higher than the mean, we select it as a hub. This approach makes hub selection more flexible, allowing the algorithm to adapt to different requirements. A small value of q selects higher number of hubs resulting in larger communities with greater overlaps, while increasing q has the opposite effect. This also allows the algorithm to discover several layers of community structure on the network.

Finally, hub selection can be extended to weighted graphs in a natural way. As before, the hub value of a node is the number of cliques it belongs to. Then the values are multiplied with the strength[6] of the node. After this, the process is the same as in the previous strategies.

In summary we propose the following hub selection strategies:

- 1-neighborhood median: A node is selected as a hub if its hub value is greater than the median of hub values in its one-neighborhood.
- 1-neighborhood mean with multiplier: A node is selected as a hub if its hub value is greater than the mean of hub values in its one-neighborhood multiplied with a parameter $q > 0$.
- 1-neighborhood weighted mean with multiplier: The hub values are multiplied with the strength of the nodes. Beside this, the strategy is the same as above.

As a recommendation, the median strategy should be tried first, and if it does not give satisfactory results the average strategy should be used with $q = 1$ initially, decreasing or increasing its value in small steps depending on the requirements. In practice $0 < q < 2$ seems to hold.

3.2 Implementation

The bottleneck of the algorithm is finding all maximal cliques in graph G. A general graph with n vertices may contain up to $3^{n/3}$ maximal cliques. In correspondence, the original algorithm of Bron and Kerbosch has a worst-case running time of $O(3^{n/3})$. In small-world networks however, the number of maximal cliques is smaller by magnitudes, decreasing the running time of the algorithm. Furthermore, refinements of the Bron-Kerbosch algorithm have been published in recent years, enabling the use of this method on large sparse networks [30, 8]. In cases when even faster computation is required, there are existing heuristics for clique search [5].

The hub value of each node can be calculated in a single pass on the set C of cliques. All of the hub selection strategies suggested in the previous section have a local fashion: they can be computed in a single pass on the vertices and their one-neighborhoods.

The computation of C_H does not require a repeated run of the Bron-Kerbosch algorithm on $G[H]$, since the cliques of G contain the cliques of $G[H]$ as subsets. Therefore it is enough, that for each $c \in C$, if $|c \cap H| \geq k$, simply add all k-combinations of c to C_H. Depending on the size of the network and the hub selection strategy, H may be quite large, but the use of flags on the nodes of the graph G to signal the hub property can reduce the computation of this step to a single pass on C. The percolation step can be executed by computing the 1-neighborhood of each $c_H \in C_H$. Let c_H^+ contain all of the direct neighbors of vertex set c_H, and initially let $c_e \leftarrow c_H$. For all nodes $v \in c_H^+$, if $|\{v\}^+ \cap c_H| \geq 2$ add v to c_e. Again this step can be computed by making a single pass on C_H.

[5]The median hub selection strategy was used with $k = 2$, see subsection 3.1.

[6]The sum of the weights on all adjacent edges.

In order to make the joining step, the computation of the hubs of each elementary community is required: for each $c_e \in C_e$ let $c_{H_e} = c_e \cap H$. Let C_{H_e} denote the sets of hubs of the elementary communities. An important remark is, that $C_{H_e} \neq C_H$ since in the previous step additional hubs may have been added to the elements of C_H. Removing the "sub-hubs" (hubs being contained in other elements of C_{H_e}) can be executed in quadratic time in worst case. In general, performance can be improved by sorting C_{H_e} in descending order according to the sizes of $c_{H_e} \in C_{H_e}$. After this, starting from the first element, remove all the sets of vertices from C_{H_e} which are subsets of the first one, then repeat for the second, third, ... until no more vertex sets can be removed from C_{H_e}. Finally, the elements of C_e and C_{H_e} must be compared: for all $c_{H_e} \in C_{H_e}$ find all $c_e \in C_e$ where $c_e \cap H \subseteq c_{H_e}$ and take the union of these vertex sets.

We can conclude, that the two most time-consuming steps of the method is the computation of C and C_H, all other operations take at most quadratic time[7]. The algorithm of Eppstein and Strash [8] is able to list all maximal cliques in large sparse networks in reasonable time. For faster computation heuristics [5] or the use of quasi-cliques [23] can be applied. The size of C_H depends on two factors: the size of H and k. The former is governed by the hub selection strategy, the latter is a parameter of the algorithm. Choosing a different hub selection strategy, that produces a smaller number of hubs, or decreasing k may speed up computations.

4 Sensitivity to parameters

We have created the hub percolation method with the intent to provide a versatile tool for community detection. Therefore, an important question arises: how does the hub selection strategy and the filtering parameter influence the community structure found by the algorithm? For the purpose of examining their effect, we will use several well-known benchmark networks including the word association graph of Palla et al.[29], a scientific collaboration network [26] and a graph of American football games [11].

The first network we will examine was created by Newman [26] on the condensed matter archive at www.arxiv.org based on preprints posted to the archive between January 1, 1995 and March 31, 2005. The graph is undirected, unweighted and contains 39540 nodes and 175683 edges. We will evaluate the median and average hub selection strategies and we will also experiment with different values for k. We will measure the number of communities, the average overlap[8], the number of singletons[9] and hubs in the network. We will also present the community size distribution for each selection strategy.

[7]We have also conducted experiments, and found that clique detection may take from 60% up to 95% of the running time.

[8]The sum of the cardinalities of all community divided by the number of nodes.

[9]Nodes without communities.

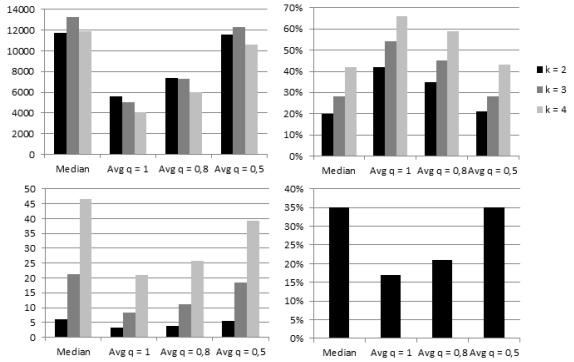


Figure 2: Upper left: Number of communities for different hub selection strategies and values for k; Upper right: The percentage of nodes without communities; Lower left: The average overlap; Lower right: The percentage of hub nodes in the network.

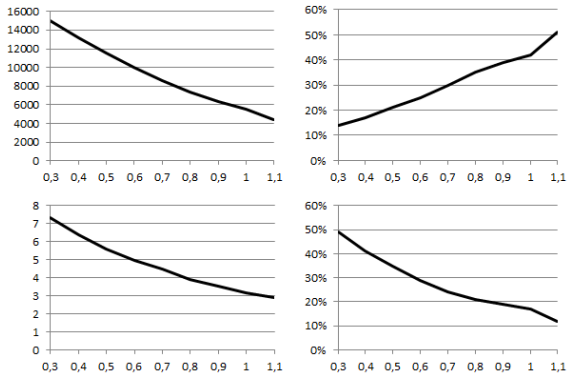


Figure 3: Upper left: Number of communities for the average hub selection strategy with $q = 0.3, \ldots, 1.1$ and $k = 2$; Upper right: The percentage of nodes without communities; Lower left: The average overlap; Lower right: The percentage of hubs in the network.

On Figure 2 we have compared four different hub selection properties: the median strategy and the average strategy with values $q = 1, 0.8, 0.5$. The number of communities is the greatest and the number of singletons is the lowest with the median strategy and the average strategy with $q = 0.5$; these strategies provide the greatest cover on the network. The number of hubs is also the greatest with these strategies: roughly one in three nodes, this confirms our expectations, that hubs should be "common". We can see, that the average overlap and the number of singletons increases with k, while the number of communities does not change. The reason for the above fact is that by increasing k, the nodes are concentrated in highly overlapping communities keeping the number of communities constant, while many nodes are left out of the community building process.

We will further examine the average hub selection strategy with $k = 2$ on Figure 3. The main observations remain the same with higher values for k. As before, the number of hubs and communities grows inversely proportional

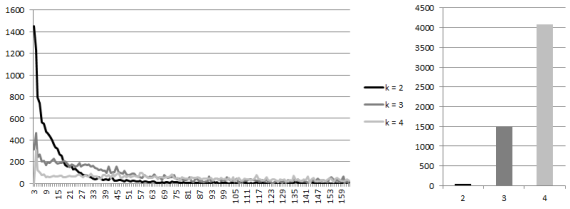


Figure 4: The community size distribution of the median hub selection strategy with different values of k. Left: The number of communities with size below 150. Right: The number of communities with size greater than 150.

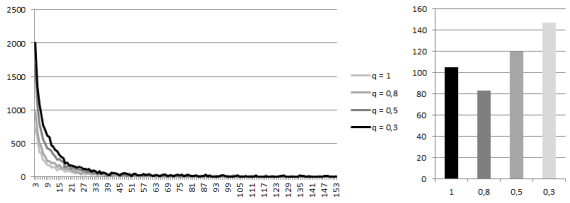


Figure 5: The community size distribution of the average hub selection strategy with different values of q, $k = 2$. Left: The number of communities with size below 150. Right: The number of communities with size above 150.

with q, while the number of singletons grows proportionally with it. The average overlap slowly decreases when q is increased, indicating that decreasing the number of hubs causes communities to become smaller and scarcer.

We can see, that the community size distributions follow a power-law on Figures 4 and 5. The median hub selection strategy is depicted with different values for k. Increasing k results in much larger communities: With $k = 2$, The largest community had 255 members, with $k = 4$ the maximum was 869. This confirms our previous observation, that increasing k creates a highly overlapping community structure. Similar observations can be made with the average hub selection strategy. Increasing q decreases the number of communities evenly among the community sizes, even the size of the largest communities does not change much.

A strict requirement for all community detection algorithms should be, that the number of nodes left without community memberships should be minimized. Therefore we can conclude, that the filtering parameter should be kept as low as possible, and the ratio of hubs should be above 30%.

We have measured the running time of our method as well[10]. The results for the average hub selection strategy with different values of q and k can be seen on Figure 6. We have seen before, that decreasing q increases the number of hubs – the size of H and C_H. This directly increases the computational time of the joining phase. The filtering parameter k also has an impact on the running time of the method, since it influences the size of C_H. As a conclusion we can say, that the filtering parameter should be kept as low as possible, and the average hub selection strategy

[10]We have implemented our method in JAVA, and we have used a computer with an Intel i7-2630QM processor, and 8 gigabytes of memory.

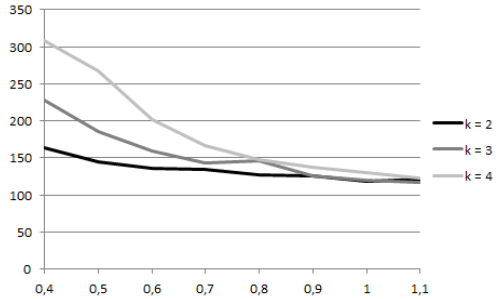


Figure 6: The running time measured in seconds with the average hub selection strategy and different values of q and k.

should be used to further refine the results of the algorithm.

We can draw similar conclusions on the other two networks, with a few exceptions. The relationship between the ratio of hubs, the community size and the average overlap is the same in all networks. The ratio of singletons shows a similar behavior as it grows inversely proportional to the ratio of hubs. There is a difference however; the graph of football games contains no singletons for the majority of the parameter configurations, while the ratio of singletons never goes below 30% in the word association network. This can be explained by the difference in the structure of the networks. The graph of football games is an union of cliques by definition, while word associations do not have this property. Since our method is clique-based, it is able to cover all nodes of the former test set, while in the latter case nodes not part of any triangles are left out of the building process.

The relationship between the ration of hubs and the hub selection strategies is also similar, that is for the average selection strategy increasing q decreases the number of hubs. However, the exact pairs of these values change together with the networks. For example setting $q = 0.5$ results in 35% of nodes being selected as hubs on the collaboration network, 21% on the word association network and 90% on the graph of football games. Therefore in any application, it is important to find the hub selection strategy that produces the ratio of hubs so that the number of communities, the size of the overlaps and the number of singletons move according to the specifications of the application.

We have previously concluded that the filtering parameter should be kept as low as possible to reduce both the ratio of singletons and the computation speed. As we will see below, there are some situations where a higher value is desirable. In the next chapter we are going to examine networks with a large number of highly overlapping communities.

5 Performance on benchmark networks

For the purpose of evaluation, we have used benchmark networks created with the graph generator of Lancichinetti

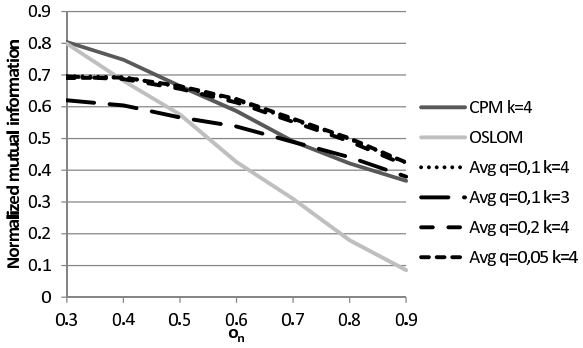


Figure 7: The performance of hub percolation compared to CPM and OSLOM with $\mu_t = 0.1$.

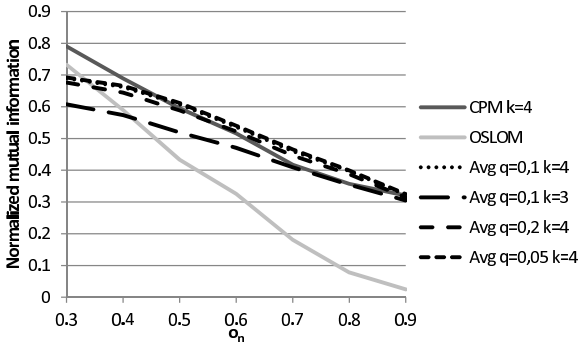


Figure 8: The performance of hub percolation compared to CPM and OSLOM with $\mu_t = 0.2$.

and Fortunato [18]. We have generated both weighted and unweighted networks, with the following parameters:

- We have created undirected graphs with $|V(G)| = 1000$
- The average degree was 15
- The maximum degree was 50
- The exponent of the degree distribution was -2
- The minimum community size was 3
- The maximum community size was 25
- The exponent of the community size distribution was -1
- The mixing parameter μ_t was between 0.1 and 0.2
- The fraction of overlapping nodes o_n was between 0.3 and 0.9

A detailed description of the used model and its parameters can be found in [18]. We have selected the parameters above, because they are close to the recommendations of Lancichinetti and Fortunato, yet they provide a challenge to our method. Again following the recommendations of the above authors, we have used mutual information [19] to measure the similarity between the communities given by our method and those of the benchmark. Because of the probabilistic nature of the benchmark we have generated 10 different networks for each parameter configuration and averaged the similarity measurements.

We have compared the performance of our method to that of the clique percolation method and OSLOM. We have tried several values for k-clique percolation, and have found that $k = 4$ clearly provides the best results, therefore we have used this parameter setting for comparison. We have also made comparisons with COPRA but found, that the above methods are clearly superior on these benchmark networks, so we have omitted these results from the figures.

On Figures 7 and 8 we can see that the best results were provided by the 1-neighborhood average hub selection strategy with low q values and $k = 4$. We can also see,

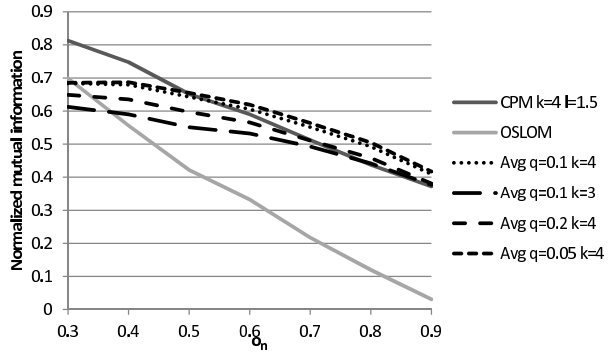


Figure 9: The performance of hub percolation compared to CPM and OSLOM with $\mu_t = \mu_w = 0.1$.

that our method reaches peak performance at $q = 0.1$, but the selection of q has little influence on the results. The median selection strategy performs poorly on these networks, and increasing or decreasing k worsens performance. If we compare our method to CPM and OSLOM, we can conclude, that hub percolation gives better results on networks with a high number of overlapping nodes.

Our observations remain the same when using the weighted benchmarks of the same authors with the recommended parameters $\mu_t = \mu_w$ and $\beta = 1.5$. Low values of q and $k = 4$ gives the best results for hub percolation, and 4-clique percolation with a weight threshold $l = 1.5$ is the best for CPM. As before, hub percolation gives better results on networks with a high number of overlapping nodes.

6 Case-study: an economic network

In this section, we will examine the community structure of a specific economic network constructed from the Hungarian company register. We will consider a network of companies: each vertex is a special type of company (Ltd.), and the companies are connected if they share a common owner (or member in the case of Ltd.'s). We will call this network as an intersection network, because two vertices

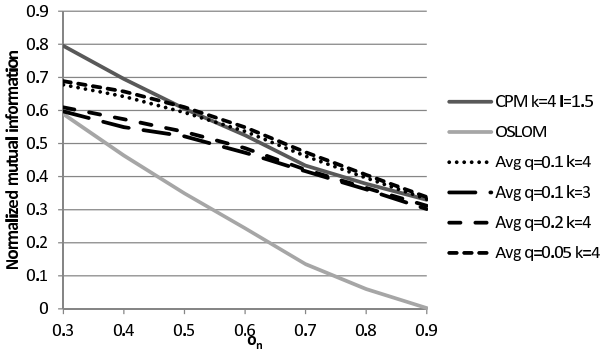


Figure 10: The performance of hub percolation compared to CPM and OSLOM with $\mu_t = \mu_w = 0.2$.

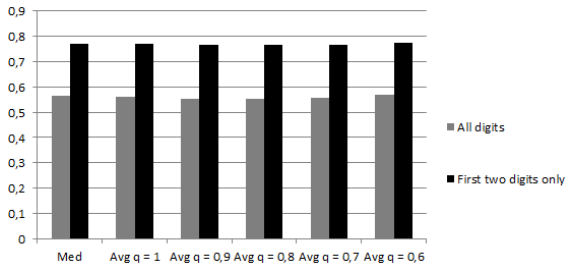


Figure 11: The locality of the communities of the intersection network with different hub-selection strategies, using all digits of the zip-code (left) or the first two digits (right).

are connected, if the sets of owners associated with them have a non-empty intersection. Due to the changes in the regulations governing the company register's construction, there are large amounts of missing and erroneous data. The register's sometimes has unordered structure so the identification of the companies, owners and the construction of the graph itself required the application of several data mining methods, data cleaning and filtering.

The resulting graph is not connected, there is a high number of small disconnected components in it, but fortunately it contains a giant component as well. The small components often cannot be divided into two or more communities, thus they do not provide useful information about the structure of the graph. Therefore, in our analysis we will consider only the giant component. This graph is a small-world network with the previously mentioned properties. It has 239685 vertices and 1423080 edges. Depending on the hub-selection strategy, our method was able to discover the community structure of this network in 5-7 hours[11]. We have considered comparing our method with CPM on this dataset, but the publicly available[12] implementation was unable to produce results.

There are several points of interests regarding the community structure of the network. In this paper, we are going to focus on three of them. The first one is the geographical location of these groups and companies inside them. Our main question is, are the communities of the graph local in a geographical sense? Using the register, we can assign zip-codes to the companies, and by counting the number of different zip-codes inside the community – the frequency of individual zip-codes, we can easily address the above question. We can further divide the frequency of the most frequent location by the size of the community, and by averaging this fraction over all of the communities we can represent the locality of these communities as a simple number. The structure of the zip code also allows us to fine-tune the resolution of the analysis. The Hungarian zip-code contains four digits: the first one divides the country into nine large regions, the first two identifies 80 sub-regions. On Figure 11 we can see the computed average locality of the communities. Both the accurate locations – all digits of the zip-code – and the sub-region classification system is used. We can conclude, that the communities are local indeed; in average 77% of companies inside communities belong to the same sub-region, and even in the case of the accurate locations, this percentage does not go below 55%. This implies, than companies owned by the same people tend to stay in the same geographical area. It is important to emphasize, that we are observing a special form of companies: the Ltd.'s. Our results makes sense, because this company form is popular for small companies, that do not have the resources to cover a large area. On Figure 11 we can also see a comparison between the different hub-selection strategies[13]. Even though the number of communities and the size of overlap changes according to the observations in the previous section, all strategies gave a similar stable performance.

We can perform the same analysis considering the industrial sectors the companies belong to. Do the communities of the graph belong to similar industrial sectors? The sector classification numbers for the individual companies are available, but due to changes in regulation it is impossible perform a high-resolution scan. On the other hand we can make use of a rough classification system containing 118 different industrial sectors. The method is the same as before: we compute the most frequent sector for each community, and we average the relative frequencies over all communities. As a result we can say, that in average 84% of the companies inside the communities belong to the same industrial sector. The communities are even more "local" to the industrial sector most of their members belong to, than to their geographical location. The reason for this is similar to the previous one: small companies tend to specialize, and it is rare for an owner to have an interest in multiple sectors. Again, we have compared different hub-selection strategies and found, that they have similar performance.

We can see a small example of this behavior on Figure 12. The whole economic graph is too large to visualize, so we are going to take a look at a small subgraph of three communities. The red community contains companies fo-

[11] On the same hardware as above.

[12] We have used CFinder [29] downloaded from http://cfinder.org/

[13] $k = 2$ was used in all experiments.

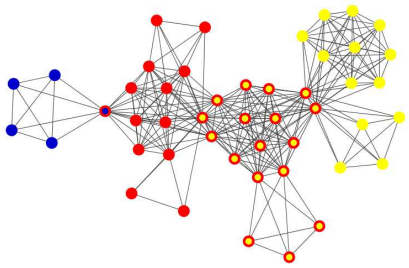


Figure 12: Three communities of the economic intersection graph.

cusing on printing services, the blue one on distributing in general, while the companies in the yellow one are centered around public utilities in real estate and engineering. A huge overlap can be seen between the red and yellow companies: vertices in the non-overlapping part of the red groups are focused on distributing, while the companies in the overlap are either copy shops, smaller publishing companies or hardware and electronics retail shops.

Our last point of interest is the age of the companies. Since the date of establishment is available for all companies, we can ask the question: Were the companies inside the communities of the graph established in a short time period? We can answer this question by computing the standard deviation of the dates of establishments for all companies. The expected value of the standard deviation inside the communities is 5 years. This relatively large value indicates, that the establishment of these companies is spread in time over a considerable interval.

As a conclusion we can say, that our method is capable of identifying communities that share a common geographical location and industrial sector.

7 Conclusions

In this paper, we have introduced the hub-percolation method: a clique-based high-resolution overlapping community detection algorithm. This method is based on two observations: cliques are the most natural representations of communities, and some vertices are crucial in the birth of communities: they connect different sub-communities together by forming bridges between them. There are multiple ways to select these vertices; that authors have suggested several hub-selection strategies, some of them have tunable parameters. The method also has a filtering parameter k which influences the size and structure of the overlaps between the communities. Adjusting k and the hub-selection strategy allows the user to apply this method to a variety of small-world networks with different densities. It also allows the user to discover several layers of community structure on the same network. We have examined the effect of different parameter choices on several well-known benchmark networks. We have concluded, that the selection strategy should be chosen so that 30-50% of the vertices are selected as hubs and k should be kept low to minimize the number of singletons.

We have shown two ways to measure the goodness of the hub-percolation algorithm. One of them was an economical case-study, a network where the vertices represent companies, or more precisely Ltd.'s, and the companies are connected if they share one or more members. Our method is able to identify communities, that are geographically local and belong to the same industrial sectors in reasonable time considering the size of the network. We have also used benchmark graphs created with the graph generator of Lancichinetti and Fortunato [18]. Using these networks, we have compared our method with the well-known clique percolation algorithm of Palla et al. We have concluded, that in average the two methods have similar performance, but hub-percolation gives better performance on networks with a high number of overlapping nodes.

Finally, a slight adjustment in the hub-selection strategy allows us to handle weighted networks without the need to filter the graph edges according to some possibly non-trivial weight limit. Using the previously mentioned graph generator, we have created weighted benchmark graph, and compared the goodness of our method with those of the weighted clique percolation algorithm. Our conclusions were the same as before: similar performance in average, better results with high overlaps.

Acknowledgement

The first author was supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TÁMOP-4.2.2.C-11/1/KONV-2012-0013)

The second author was supported by the European Union and co-funded by the European Social Fund. Project title: "Telemedicine-focused research activities on the field of Mathematics, Informatics and Medical sciences." Project number: TÁMOP-4.2.2.A-11/1/KONV-2012-0073 .

References

[1] Y-Y. Ahn, J. P. Bagrow, S. Lehman, Link communities reveal multiscale complexity in networks. *Nature*, **466**(7307):761–764, 2010.

[2] A-L. Barabási, R. Albert, Emergence of Scaling in Random Networks. *Science*, **286**(5439):509–512,1999.

[3] A. Bóta, L. Csizmadia, A. Pluhár, Community detection and its use in Real Graphs. *Proceedings of the 2010 Mini-Conference on Applied Theoretical Computer Science*, 95–99, 2010.

[4] F. Cazals, C. Karande, A note on the problem of reporting maximal cliques, *Theoretical Computer Science*, **407**(1):564–568, 2008.

[5] James Cheng, Linhong Zhu, Yiping Ke, and Shumo Chu, Fast algorithms for maximal clique enumeration with limited memory. *Proceedings of the 18th ACM SIGKDD*. ACM, New York, 1240–1248, 2012.

[6] A. Csernenszky, Gy. Kovács, M. Krész, A. Pluhár, T. Tóth, The use of infection models in accounting and crediting. *Challenges for Analysis of the Economy, the Businesses, and Social Progress* Szeged (2009) pp. 617–623..

[7] A. Decelle, F. Krzakla, C. Moore, L, Zdeborova, Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Phys. Rev. E*, **84**(6):066106, 2011.

[8] D. Eppstein, D. Strash, Listing all maximal cliques in large sparse real-world graphs. *Experimental Algorithms*, Springer Berlin Heidelberg, 364–375, 2011.

[9] T. Evans, R. Lambiotte, Line Graphs, Link Partitions and Overlapping Communities, *Phys. Rev. E*, **80**(2):016105, 2009.

[10] S. Fortunato, Community detection in graphs. *Physics Report*, **486**(3):75–174, 2010.

[11] M. Girvan, M. E. J. Newman, Community structure in social and biological networks. *Proc. Natl. Acad. Sci.*, **99**(12):7821–7826, 2002.

[12] P. K. Gopalan, D. M. Blei, Efficient discovery of overlapping communities in massive networks. *PNAS*, **110**(36):14534-14539, 2013.

[13] S. Gregory, Finding overlapping communities in networks by label propagation. *New J. Phys.*, **12**(10):103018, 2010.

[14] E. Griechisch, A. Pluhár, Community Detection by using the Extended Modularity. *Acta Cybernetica*, **10**:69–85, 2011.

[15] D. Kempe, J. Kleinberg, E. Tardos, Influential Nodes in a Diffusion Model for Social Networks. *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, Springer-Verlag (2005) 1127–1138.

[16] M. Krész and A. Pluhár, Economic Network Analysis based on Infection Models. To appear in *Encyclopedia of Social Network Analysis and Mining*, Springer (2014).

[17] J. M. Kumpula, M. Kivela, K. Kaski, J. Saramaki, Sequential algorithm for fast clique percolation. *Phys. Rev. E*, **78**(2):026109, 2008.

[18] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E*, **80**(1):016118, 2009.

[19] A. Lancichinetti, S. Fortunato, J. Kertész Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, **11**(3):033015, 2009.

[20] A. Lancichietti, F. Radicchi, J. J. Ramasco, S. Fortunato, Finding statistically significant communities in networks. *PLoS One*, **6**(4):e18961, 2011.

[21] C. Lee, F. Reed, A. McDaid, N. Hurley, Detecting highly overlapping community structure by greedy clique expansion. *Preprint*, 2010.

[22] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, S. M. Dawson, The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, **54**(4):396-405, 2003.

[23] V. Maniezzo, R. Battiti, J-P Watson, On Effectively Finding Maximal Quasi-cliques in Graphs. In *Learning and Intelligent Optimization*, Lecture Notes in Computer Science (5313) 41–55, Springer Berlin Heidelberg, 2008.

[24] J. Moon, L. Moser, On cliques in graphs. *Israel Journal of Mathematics*, **3**(1):23-28, 1965.

[25] T. Népusz, A. Petróczi, L. Négyessy, F. Bazsó, Fuzzy communities and the concept of bridgeness in complex networks. *Phys. Rev. E*, **77**(1):016107, 2008.

[26] M. E. J. Newman, The structure of scientific collaboration networks. *PNAS*, **98**(2):404–409, 2001.

[27] M. E. J. Newman, M. Girvan, Finding and evaluating community structure in networks. *Phys. Rev. E*, **69**(2):026113, 2004.

[28] V. Nicosia, G. Mangioni, C. Carchiolo, M. Malgeri, Extending the definition of modularity to directed graphs with overlapping communities. *Journal of Statistical Mechanics: Theory and Experiment*, **3**:P03024, 2009.

[29] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, **435**(7043):814–818, 2005.

[30] E. Tomita, A. Tanaka, H. Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, **363**(1):28–42, 2006.

[31] D. J. Watts, S. H. Strogatz, Collective dynamics of small-world networks *Nature*, **393**(6684):440–442, 1998.

[32] W. W. Zachary, An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 452–473, 1977.

History-based Approach for Detecting Modularity Defects in Aspect Oriented Software

Hanene Cherait and Nora Bounour
LISCO Laboratory, BadjiMokhtar – Annaba University P.O. Box 12, 23000 Annaba, Algeria
E-mail: {hanene_cherait, nora_bounour}@yahoo.fr

Keywords: modularity defects, aspect oriented programming, crosscutting concerns, frequent itemset mining, logical coupling, refactoring.

Received: May 4, 2014

The evolution of Aspect oriented (AO) software would degrade and modify its structure and its modularity. In this scenario, one of the main problems is to evaluate the modularity of the system, is the evolved AO software still has a good modularity or not? Unfortunately, this research area is not explored yet. This paper presents a history-based approach that detects modularity defects in evolved AO software. It is a two-step automated approach: 1) in the first step, it applies data mining over an AO software repository in order to detect logical couplings among its entities. It analyses fine-grained logical couplings between AO software entities as indicated by common changes. 2) These last are then analysed to detect modularity defects in the AO software system. The approach focuses on the evaluation of an AO system's modularity and points out potential enhancements to get a more stable one. We provide a prototype implementation to evaluate our approach in a case study, where modularity defects are detected in 22 releases of three well-known AspectJ systems: Contract4J, Health-Watcher and Mobile-Media. The results show that the approach is able to detect logical couplings among aspect entities, as well as modularity defects that are not easily (or not) detectable using static source code analysis.

Povzetek: Članek se ukvarja z zaznavanjem defektnosti modulov med evolucijo programov.

1 Introduction

Aspect-oriented programming (AOP) [11] allows a developer to modularize a crosscutting concern's implementation by introducing a new kind of module called "Aspect". This last encapsulates crosscutting concerns and thus improves modularity, understandability, and evolvability of the code. As any software system, AO systems are continuously modified and increase in size and complexity. After many enhancements and other evolution activities, the AO software modularity can be violated, and modifications become hard to do. The insufficient modularity of crosscutting concerns complicates AO software evolution and reduces crosscutting concern reusability. Therefore, methods and techniques are needed to detect modularity defects in AO software, in order to improve its decomposition and enhance its modularity.

To detect modularity defects in AO software, we need to understand the relationships among entities that belong to software aspects, more specifically, to the crosscutting concerns of the system. However, many works [2, 5, 20] have proved with empirical evidence the ripple effects in AO software i.e. changes are propagated to unrelated entities in the program. So, it is difficult to detect modularity defects in AO software through a static analysis of the source code (e.g. [26]). In reality, two crosscutting concerns that are supposed to be independent statically may frequently change together.

This paper presents a history-based approach to detect modularity defects in AO software. It consists of two main steps: first, an AO software repository is mined to detect logical couplings between the aspect's entities of the system—how entities actually change together. In our approach, we don't detect coupled aspects only. But, we can extract the aspect entities related to this coupling. Second, the resulted logical couplings are analysed to detect modularity defects. We identify modularity defects by external logical couplings i.e. if two entities always change together to accommodate modification requests, but they belong to two independent aspects; we consider this as a modularity defect. These last can be used to improve the AO software modularity in order to prevent it from decay. For example, the detected defects could be removed or minimized by using appropriated refactorings to change the AO software decomposition.

The rest of the paper is organized as follows: in the next section we give the background used in this paper. We describe our approach in section 3; where we present the relationship between logical couplings and modularity defects, and how this relationship can be used to detect modularity defects in AO software. The tool chain is presented in section 4. Our approach is applied on a case study in section 5. Section 6 summarizes related work. Finally, section 7 closes with conclusions and future work.

2 Background

In this section, we first introduce definitions of important concepts related to our proposal. Then, we give a brief description of the AspectJ language.

2.1 Modularity defects

The IEEE Standard Glossary of Software Engineering Terminology (IEEE, 1990) defines modularity as "the degree to which a software system is composed of discrete components such that a change to one component has minimal impact on other components". So, it allows each part to be modified, substituted or deleted with minimal impact on the rest of the system. Modularity has been playing a pervasive role in the context of software development and evolution. It can be considered a fundamental engineering principle as it allows:

- to develop different parts of the same system by distinct people;
- to test systems in a simultaneous fashion;
- to substitute or repair defective parts of a system without affecting with other parts;
- to reuse existing parts in different contexts; and
- to restrict change propagation.

In reality, however, during software evolution two modules that are supposed to be independent may always change together, due to unwanted side effects caused by quick and dirty implementation [24]. When such couplings exist, the software can deviate from its designed modular structure, which is called a *modularity defect (violation).* Such modularity defects could cause modularity decay over time and may require expensive system-wide refactorings. Detecting and fixing modularity defects make programs easier to understand and to evolve.

2.2 Logical coupling

Semantically coupled software entities may not structurally depend on each other i.e. different entities of a software system may be related to each other although this relationship is not easily detectable in the software source code. When different entities of a software system change together (as the system evolves) their common behavior is referred to as logical coupling [7]. Recently, researchers have used revision histories to more effectively identify semantically coupled components by checking how components historically change together [9, 10].

Logical couplings detection extract interesting dependencies between software entities that is not possible with the analysis of a single version. So, based on the historical data we can detect logical couplings between the entities of a software system. In this last, two entities are coupled whenever a change in an entity A implies a change in another entity B—one says that B depends on A.

The logical couplings have been used for different purposes: to identify hidden architectural dependencies, to point developers to possible places that need change, or to use them as change predictors. In our context, we use such dependencies to evaluate the modularity of an AO software system.

2.3 AspectJ

AOP is a new paradigm introduced by Kiczales et al. [11] that provides separation of crosscutting concerns. It modularizes the crosscutting concerns in a clear-cut fashion, yielding a system architecture that is easier to implement, and to evolve. With AOP, a program is composed with a set of aspects, and a base code describing the core modules. An *aspect weaver*, which is a compiler-like entity, composes the final system by combining the core and crosscutting modules through a process called *weaving* [12].

AO languages offer abstractions for the implementation of crosscutting concerns whose modularization cannot be achieved by using traditional programming languages. During the last decade, a considerable number of AO languages have been introduced. AspectJ [12] has been the pioneer of the AO languages, and it is still one of the most relevant frameworks supporting the AOP methodology. For the remaining of this paper, we will use AspectJ as our target language, although the observations made are also valid for other currently available AspectJ-like languages. AspectJ defines two types of crosscutting: *dynamic* crosscutting and *static* crosscutting.

Dynamic crosscutting: is the weaving of new behaviour into the execution of a program using: *join point*, *pointcut* and *advice*. We briefly introduce each of these constructs as follows:

- **Join Point:** denotes points at which crosscutting code can be executed. The join point is a well-defined "point" in the dynamic execution flow of an application. For instance, in object oriented languages, join points may refer to passing messages and writing on instance variables.
- **Pointcut:** is a program element that picks out join points and exposes data from the execution context of those join points. The pointcut language of AspectJ offers a set of *primitive pointcut designators*, like call specifying method call or get/set specifying field access. These primitive pointcut designators can be combined using logical operators (and "&&", or "||", not "!").
- **Advice:** represents a program module which is to be executed at the designated join points. There are three types of advices *before*, *after* and *around*, which correspond to the program modules to be executed prior, after or instead of the designated events, respectively. It is defined in terms of pointcuts. The code of a piece of advice runs at every join point picked out by its pointcut.

Static crosscutting: is the weaving of modifications into the static structure—the classes, interfaces, and aspects—of the system. By itself, it does not modify the system behavior, but it operates over the static structure of type hierarchies. AspectJ provides inter-type member declarations (introductions) and other declare forms. It makes static changes to the modules of the system, for example, we can add a method or field to a class.

Finally, an Aspect is a modular unit designed to implement a crosscutting concern. It contains the code that

expresses the weaving rules for both dynamic and static crosscutting. An aspect may also incorporate member variables, methods, etc., just like a normal class Java.

3 Our approach

3.1 Basic idea

To understand better our contribution, it is important to define clearly the relationship between logical coupling detection and modularity defects. In this section, we present the utility of logical couplings in the detection of software modularity defects. And, we explain how this idea can be used in the context of AO software.

There is a strong correlation between modularity defects and logical couplings. Some modularity defects are not easily detectable by static or dynamic software analysis. Fluri et al.'s [7] study shows that a large number of change coupling relationships are not entailed by structural dependencies.

Extracting logical couplings and analysing them can help in detecting modularity defects in a software system. The basic idea is that we can distinguish two types of logical couplings, as depicted in Figure 1: internal and external logical couplings. A logical coupling is an internal logical coupling, if it relates two entities that belong to the same module in the software decomposition. On the other hand, an external logical coupling relates two entities that belong to two different (independent) software modules.

The last type is the most important in our context; because the existence of external logical couplings in a software system presents possible modularity defects in that system. Two modules that are supposed to be changed independently are changed together i.e. a change in an entity that belong to a specific module will necessitate changes in other(s) entity(s) that belong to other module(s). So, we call such logical couplings "negative logical couplings" or "modularity defects".

In AOP, the crosscutting concerns are modularized by identifying a clear role for each one in the system, implementing each role in its own module, and loosely coupling each module to only a limited number of other modules [12]. Unfortunately, these systems need to evolve continually in order to cope with ever-changing software requirements. Empirical results show that AO software is not immune from the negative side effects of software evolution [2, 5, 20]. This fact harms the modularity of the AO program, hinders the concerns encapsulation and reduces the aspect reusability. To overcome this problem is a hot topic.

Our research question is: how we can detect efficiently the modularity defects in AO software? To this end, we use the idea described above to achieve our goal. In this context, software modules (Figure 1) are the crosscutting concerns (Aspects) of the system, and the modularity defects are considered as external logical

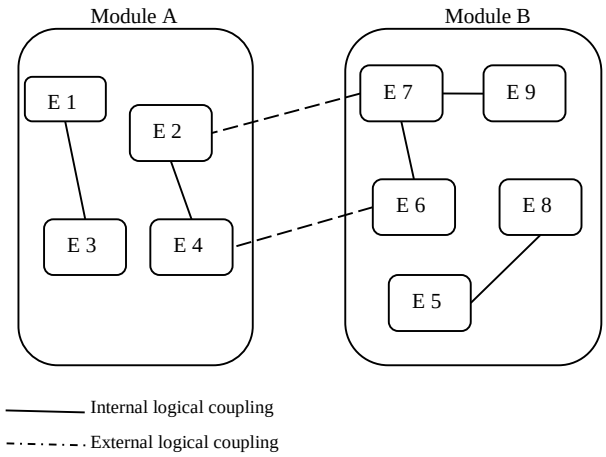


Figure 1: Types of logical couplings.

couplings among these aspects. So a modularity defect in AO software can be defined as follows:

Definition of a modularity defect: let A and B two independent aspects. A modularity defect (x, y) is a logical coupling between the two entities x and y, where $x \in A$ and $y \in B$.

To resume up, just how well does the AO software system evolution justify its best modularity? The existence of modularity defects (external logical couplings) in an AO system shows that the separation of crosscutting concerns (modularity) into that system is violated. The coupled crosscutting concerns are candidates for restructuring or refactoring. Here, the detected modularity defects are used to guide improvement efforts; in order to get a more stable decomposition with very little dependencies i.e. an ideal situation would allow changing each crosscutting concern independently of the others. This is very useful to reconstruct a best modularization for the AO software system and a good reusability of their crosscutting concerns.

3.2 Approach overview

The purpose of this paper is to present an approach to uncover modularity defects in AO software by analysing its evolution history. As depicted in Figure 2, our approach consists of two complementary steps, which form an integrated approach for detecting modularity defects: 1) An AO software repository is mined to detect logical couplings between the software entities that belong to the different aspects of the system; 2) The resulted logical couplings are then analysed according to the AO software decomposition to detect and locate modularity defects. If two entities x and y are frequently changed together, and they belong to two independent aspects A and B respectively, so the logical dependency (x,y) represents a modularity defect. The main purpose of such modularity defects is to evaluate how modular an AO application is, and to guide improvement efforts i.e. these couplings can be used to guide the software developer during restructuring and refactoring tasks.

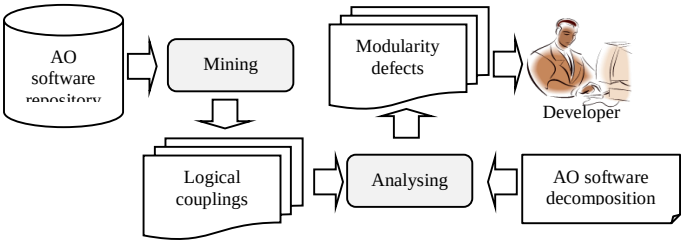


Figure 2: Modularity defect detection.

So, it is to the developer to examine the corresponding code (the entities that are related to a modularity defect) to improve it and enhance the AO software modularisation.

3.3 Logical coupling detection

In this step, a syntactic analysis of the Aspects source code is performed, such that additions and modifications of aspect's entities can be recorded. So, the source data for the mining will constitutes of the different building blocks of the software aspects: fields, methods, pointcuts, advices, and introductions. With our Mining approach, we address the following questions: 1) what are the coupled aspect entities in the AO system? and 2) what are the strengths of these couplings?

3.3.1 Coupled aspect entities

As we are mining for entities that are frequently changed together, it seems natural to use the technique called frequent itemset mining, which is able to discover interesting relations in a database. Our mining approach follows these steps: it acquires aspects data from a repository and transforms them into change-sets, which consist of the names of the entities added or modified in each transaction. Filtering may help in avoiding irrelevant data at this stage. We aim to track and mine software entities belonging to aspects, so we do not take into account base code entities. We focus on the logical couplings among crosscutting concerns only. So, we keep in every transaction only the aspect entities of an AO system (not base code entities). These are then processed using the Apriori frequent itemset mining algorithm [1].

Let $E = \{e_1, e_2, \dots, e_n\}$ be a set of aspect's entities i.e. entities that belong to the different aspects of the AO software, and $X \subseteq E$ an entity-set. We define repository R as a set of transactions: $R = \{t_1, t_2, \dots, t_m\}$, where $t_i = \{e_{i1}, e_{i2}, \dots, e_{ik}\}$ and $e_{ij} \in E$. Also, let $s(X)$ be the set of transactions that contain entity-set X, formally $s(X) = \{Y \in R | Y \supseteq X\}$. Finally, the support of an entity-set X is the fraction of transactions in the repository that contain X: $support(X) = \frac{|s(X)|}{|R|}$. Then X is called a logical coupling when its support is higher than a given minimum support: $support(X) \geq minsupport$.

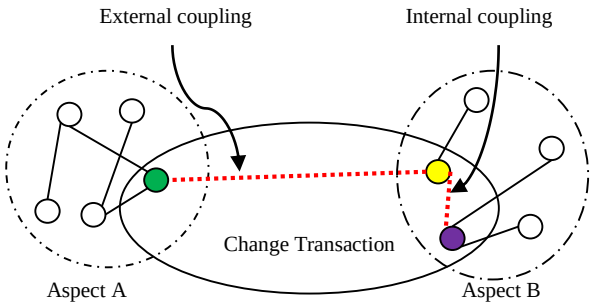


Figure 3: Analysing logical couplings.

3.3.2 Strength of a logical coupling

The strength of a logical coupling is considered as the support of this logical coupling. So, the strength of a logical coupling $\{e_1, \dots, e_n\}$ where each e_i is an aspect entity, is measured by support which is the number (or percentage) of transactions containing the entities $e_1, \dots, e_n$.

3.4 Modularity defect detection

The logical couplings extracted in the above step, are then analysed to detect modularity defects in the AO software system. This analysis is based on the structural decomposition of the AO software as crosscutting concerns (Aspects). In this step, the detected logical couplings are classed into two categories: internal and external logical couplings. As depicted in Figure 3, we define internal coupling as a dependency between two entities that belong to the same Aspect. The couplings between entities of an Aspect and any other entities that belong to other aspects are considered as external couplings.

These external logical couplings are considered as possible defects in the AO software modularity. Formally, a set of external logical couplings ELC is defined as ELC = $\{(e_i, e_j) | e_i \in A, e_j \in B\}$, where A and B are two independent aspects. So, the set of modularity defects MD in an AO program P is defined as:

$$MD(P) = \sum_{i=1}^{|ELC|} ELC_i$$

Since modularity defects are logical couplings, each modularity defect has a strength/support value (the number of transactions that contain the external logical coupling). So, we can say that (x, y) is a modularity defect that occurred once, (y, z) is a modularity defect that occurred twice, and so on.

3.5 Discussion

Using the detected logical couplings between aspect entities, we can deduce the coupled crosscutting concerns (aspects) in the AO system. As depicted in Figure 4, if two aspect entities e_1 and e_2 that belong to the independent aspects A_1 and A_2 respectively (modularity defect). Then, we deduce automatically that A_1 and A_2 are coupled aspects.

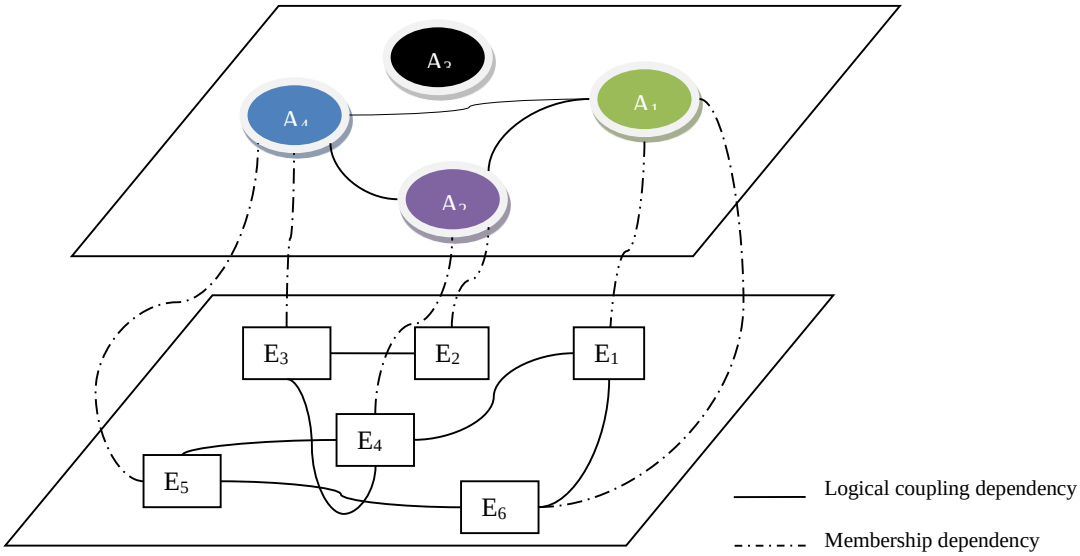


Figure 4: Coupled aspects.

The detected modularity defects can be used to assist restructuring and refactoring tasks. If some aspects change at the same time very often over several releases, they can be used to point to candidates for refactoring.

On the other hand, our results can be used to evaluate the AO software modularity. We can for example define a modularity measure using the detected logical couplings (it can be equal to the number of no coupled aspects, devised by the total number of aspects). Based on this measure we can evaluate the AO system modularity. This measure can be used later to compare many implementations of AO software systems. So, we can answer interesting questions as: Is the AO program P more modularized then the AO program P'? Is the implementation of the crosscutting concern C in program P is much more encapsulated than in program P'? If we detect crosscutting concerns (aspects) that have no coupling to any other crosscutting concerns, these can be a perfect reusable crosscutting concerns.

4 Tool chain

This section describes the tool-chain with which we identify modularity defects in AO programs written in AspectJ [12]. This last is a well-established AOP language. As depicted in Figure 5, the overall process is performed using three main tools:

The AspectJML Tool: an existing open source proposed by Melo Junior and Mendonça [13]. It is an XML-based markup language for representing source code written in AspectJ. The AspectJ source code is converted in XML (eXtended Markup Language) format [21] through the power of AspectJML. This XML-based representation is then used by the other tools in the tool-chain.

The Mining Tool: We have implemented this tool to extract logical couplings from the AspectJ repository. First, this tool takes change transactions from the repository and filters them to keep just the changed entities belonging to the aspects of the system (not base

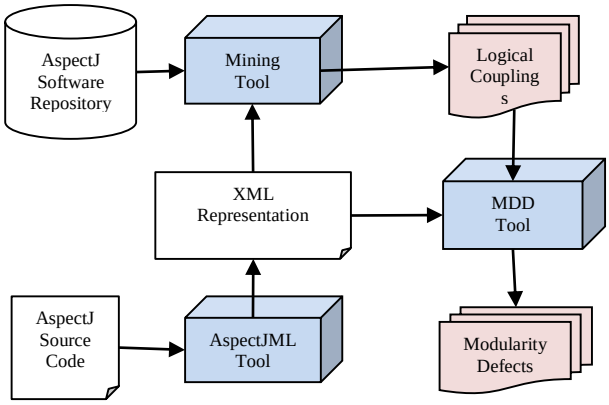


Figure 5: Tool chain.

code entities) in every transaction. Then, every entity in the transactions is replaced by its identifier. This last is extracted from the XML-based representation of the AspectJ source code. So, the tool gets change transactions of entity identifiers and organizing them in a single XML document. Finally, the transactions are mined using the Apriori algorithm.

Here, we used the XQuery implementation of the Apriori algorithm proposed by Wan and Dobbie [22]. The output of this tool is the logical couplings in the AspectJ source code that have a support higher than a specific threshold (min support). Every logical coupling is a set of entity identifiers.

The MDD Tool: a Modularity Defect Detection tool is implemented to filter the logical couplings obtained by the Mining tool. Here, the tool extracts modularity defects by eliminating internal logical couplings. It uses the XML-based representation of the source code to test if the entities that belong to a specific logical coupling are existing in the same aspect, or in different aspects using their identifiers. The results present possible modularity defects.

5 Case study

In order to assess the feasibility and correctness of our approach, this case study uses 22 releases of three well-known AspectJ programs available as open source. These systems were selected because they are rich in kinds of crosscutting concerns. Also they are used as case study in different research works [4, 5, 8, 14].

Table 1 describes these systems. It gives the number of versions and aspects of each software system. The first one, called Contract4J, it supports "Design by Contract" programming in Java. We considered the 5 releases of Contract4J in our study. The second is a product line for deriving applications that manipulate photos, videos and music on mobile devices called Mobile Media [6]. We selected its 7 releases in this experimentation. The last system called Health Watcher [19]; is a real Web-based information system that allows citizens to register complaints about health issues in public and heath care institutions to investigate and take the required actions. We selected the 10 releases of Health Watcher in our study.

After the application of our approach on these systems, we find many internal logical couplings as: “frequently changing a pointcut involves changing its related advices”, “changing a field, involves changing the methods that use this field”, etc. Many modularity defects (external logical couplings) are detected also. Table 2 presents the detected coupled aspects in each system. For each coupled aspects, it gives the number of detected modularity defects (external logical couplings). Besides, it gives the support of each coupling. Here the support is the average of the supports of the related modularity defects i.e. the support of a logical dependency between two Aspects A and B is the sum of supports of their related modularity defects, devised by the number of such modularity defects.

In the program Contract4J we have detected that the aspects *ConstructorBoundaryConditions* and *MethodBoundaryConditions* are tightly coupled with 7 modularity defects. Here the coupled low-level entities with the higher support for these aspects are: the method *doTest* in the aspect *ConstructorBoundaryConditions* and the method *doBeforeTest* in the aspect *MethodBoundaryConditions*.

Software	#versions	#Aspects
Contract4J	5	8—14
Mobile Media	7	4—42
Health Watcher	10	11—23

Table 1: Subject programs.

The aspect *ConstructorBoundaryConditions* is also coupled with the aspect *UsageEnforcement* through 3 modularity defects. The modularity defect with the higher support here is between: the advice applied after the pointcut *postCtor* in the aspect *ConstructorBoundaryConditions* and the pointcuts *preNotInContract*, *postNotInContract*, and *invarNotInContract* that belong to the aspect *UsageEnforcement*.

In the Mobile Media program, we have detected much more coupled aspects than those detected in the Contract4J program. The aspects *DataModelAspectEH* and *UtilAspectEH* are coupled via 2 modularity defects: the pointcuts *loadMediaDataFromRMS* and *readMediaAsByteArray* belonging to *DataModelAspectEH* and *UtilAspectEH* respectively are frequently changed together. Also, the pointcuts *getMedias* and *getBytesFromMediaInfo* are tightly coupled.

We have also detected that the aspect *SortingAspect* is coupled with 3 other aspects, which restricts its evolvability and reusability. It is coupled with the aspect *FavouritesAspect* via 4 modularity defects. Besides, it is coupled with the aspects *ControllerAspectE* and *CopyPhotoAspect* through one modularity defect. The most frequent detected modularity defects here are of the type pointcut duplications. For instance, the pointcuts *handleCommandAction* and *appendMedias* are duplicated in the aspects *FavouritesAspect* and *SortingAspect*. Besides, the pointcut *showImage* is defined in the aspects *ControllerAspectEH* and *SortingAspect*. So any modification in such pointcuts implies changes in many aspects.

Application	Coupled aspects	#Modularity defects	Support
Contract4J	ConstructorBoundaryConditions MethodBoundaryConditions	7	0,6
	ConstructorBoundaryConditions UsageEnforcement	3	0,6
Mobile Media	DataModelAspectEH UtilAspectEH	2	0,4
	FavouritesAspect SortingAspect	4	0,4
	ControllerAspectEH SortingAspect	1	0,2
	CopyPhotoAspect SortingAspect	1	0,2
Health Watcher	—	—	—

Table 2: Detected modularity defects.

Finally, in the Health Watcher application we have detected many internal logical couplings, but we do not detect serious modularity defects in that system (except of a few external logical couplings which are detected once). So, in contrast to the above applications (Contract4J and Mobile Media), we can say that Health Watcher has a good modularization, and their crosscutting concerns (Aspects) are good reusable modules.

6 Related work

This section of the paper presents related works discussing the benefits of our proposal in contrast to the other ones. Our work involves the following research areas:

AO software analysis: Existing approaches for detecting dependencies among AO software generally use static analysis [15, 17, 25, 26]. Such approaches are mainly based on an instruction-level to analyse the evolution of an AO software system: the source code is analysed and source code slicing is used to perform change impact analysis. We may say that such code-based approaches reveal syntactic dependencies and what we are really interested in is logical dependencies among AO software concerns. On the other hand, the information is derived using analysis of textual software artefacts that are found in a single version of the software. In contrast, our approach is based on an empirical observation of AO system structural modifications. We treat the whole evolution history to detect the modularity defects.

Mining AO software repositories: There are many approaches and techniques for detecting logical couplings in OO software [9, 10]. These works prove that such historical analysis is often able to capture couplings among software entities that cannot be captured by static and dynamic analysis. But this research area still not enough explored for AO software. Few works are dedicated to mine AO software repositories. For instance, Qian et al. [16] treat the detection of change patterns in AspectJ programs. They analyse the successive versions of an AspectJ program, and then decompose their differences into a set of atomic changes. Finally, they employ the Apriori data mining algorithm to generate the most frequent item-sets. In [3], we have also detected change patterns in AspectJ software by Mining a rewriting rule-based repository. In this paper, our goal is different, as we aim at identifying logical couplings between the aspect entities instead of change patterns.

Detecting software modularity defects: Many works prove the benefits of analysing the OO software evolution history for assessing its modularity. In [18] the authors state that to improve current modularity views, it is important to investigate the impact of design decisions concerning modularity in other dimensions, as the evolutionary view. They propose the ModularityCheck tool to assess package modularity using co-change clusters, which are sets of classes that usually changed together in the past.

Wong et al. [23, 24] presented CLIO, a tool that detects and locates modularity violations. CLIO compares how components should co-change according to the modular structure and how components usually co-change retrieving information from version history. A modularity violation is detected when two components usually change together but they belong to different modules, which are supposed to evolve independently. We use the same idea to detect modularity defects in AO software. However, these works extract couplings at a file level; in contrast, we detect logical couplings at entity level. Our detected fine-grained logical couplings can be very useful for restructuring and refactoring tasks.

7 Conclusion

Unintended modularity defects of AO software may not be easily detectable by static or dynamic analysis techniques, but could cause modularity decay and bad separation of crosscutting concerns. To detect such modularity defects, we suggested a history-based approach based on the logical couplings in the AO software system.

Our approach applies frequent itemset mining over an AO software repository in order to detect logical couplings among its entities. The extracted logical couplings are then analysed to detect modularity defects in the AO software system. Many case studies are experimented to demonstrate the feasibility of our approach. The results show that the approach is able to detect logical couplings among aspect entities, as well as modularity defects. The approach leads naturally to an evaluation of AO system's modularity. The results of our approach can be used for reducing the dependencies between AO software Aspects and consequently promoting its modularity.

The same idea can be used for detecting other types of AO software defects. For example, we can analyse the AO software evolution history for detecting bad smells, anti-patterns, etc.

References

[1] R. Agrawal, and R. Srikant (1994). Fast algorithms for mining association rules in large databases. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, Proceedings of 20th International Conference on Very Large Data Bases, Santiago, Chile, pp. 487–499.

[2] R.T. Alexander, J. M. Bieman, and A. A. Andrews (2004). Towards the Systematic Testing of Aspect-Oriented Programs. Report CS-04-105, Colorado State University, Fort Collins-USA.

[3] H. Cherait, and N. Bounour (2014). Detecting Change Patterns in Aspect Oriented Software Evolution: Rule-based Repository Analysis. International Journal of Software Engineering and Its Applications (IJSEIA), Vol. 8, No. 1, pp. 247-266.

[4] R. Dyer, H. Rajan, and Y. Cai (2012). An Exploratory Study of the Design Impact of Language Features for Aspect-oriented Interfaces. In Proceedings of AOSD'12, Potsdam, Germany.

[5] F. Ferrari, R. Burrows, O. Lemos, A. Garcia, E. Figueiredo, N. Cacho, F. Lopes, N. Temudo, L.

Silva, S. Soares, A. Rashid, P. Masiero, T. Batista, and J. Maldonado (2010). An Exploratory Study of Fault-Proneness in Evolving Aspect-Oriented Programs. In Proceedings of ICSE '10, Cape Town, South Africa, ACM press, pp. 65 – 74.

[6] E. Figueiredo, N. Cacho, C. Sant'Anna, M. Monteiro, U. Kulesza, A. Garcia, S. Soares, F. Ferrari, S. Khan, F. Castor Filho, and F. Dantas (2008). Evolving software product lines with aspects: an empirical study on design stability. In Proceedings of ICSE'08.

[7] B. Fluri, H. C. Gall, and M. Pinzger (2005). Fine-grained analysis of change couplings. In Proceedings of 5th WICSA'05, pp. 66–74.

[8] P. Greenwood, T. T. Bartolomei, E. Figueiredo, M. Dósea, A. F. Garcia, N. Cacho, C. Sant'Anna, S. Soares, P. Borba, U. Kulesza, and A. Rashid (2007). On the impact of aspectual decompositions on design stability: An empirical study. In Proceedings of ECOOP, pp. 176–200.

[9] A. E. Hassan (2008). The road ahead for mining software repositories. In Frontiers of Software Maintenance, pp. 48–57.

[10] H. Kagdi, M. L Collard, and J. I. Maletic (2007). A Survey and Taxonomy of Approaches for Mining Software Repositories in the Context of Software Evolution. Journal of Software Maintenance and Evolution: Research and Practice, Vol. 19, No. 2, pp. 77-131.

[11] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J. M. Loingtier, and J. Irwin (1997). Aspect-oriented programming. In Proceedings of 11th European Conference on Object-Oriented Programming, Springer-Verlag, LNCS Vol. 1241, pp. 220–242.

[12] R. Laddad (2003). AspectJ in Action: Pratical Aspect-Oriented Programming. Manning Pubblications Company.

[13] L. S. Melo Junior, and N. C. Mendonça (2005). AspectJML: A Markup Language for AspectJ. In Proceedings of the 2nd Brazilian Workshop on Aspect Oriented Software Development, Uberlândia, MG, Brazil.

[14] A. C. Neto, M. Ribeiro, M. Dosea, R. Bonifacio, P. Borba, and S. Soares (2007). Semantic Dependencies and Modularity of Aspect-Oriented Software. In Proceeding of the First International Workshop on Assessment of Contemporary Modularization Techniques (ACoM'07).

[15] E. K. Piveta , M. Hecht , M. S. Pimenta , and R. T. Price (2006). Detecting bad smells in AspectJ. Journal of Universal Computer Science.

[16] Y. Qian, S. Zhang and Z. Qi (2008). Mining Change Patterns in AspectJ Software Evolution. In Proceedings of the International Conference on Computer Science and Software Engineering, pp. 108-111.

[17] M. Rinard, A. Salcianu, and S. Bugrara (2004). A classification system and analysis for aspect-oriented programs. In Proceedings of FSE'04, pp. 147–158.

[18] L. L. Silva, D. Félix, M. T. Valente, M. de A. Maia (2014). ModularityCheck: A Tool for Assessing Modularity using Co-Change Clusters. In Proceedings of the Brazilian Conference on Software: Theory and Practice (CBSoft'14) - Tool Session.

[19] S. Soares, E. Laureano, and P. Borba (2002). Implementing distribution and persistence aspects with AspectJ. In Proceedings of the 17th OOPSLA.

[20] F. Steimann (2006). The Paradoxical Success of Aspect-Oriented Programming. In Proceedings of OOPSLA'06, pp. 481-497.

[21] J. Suzukiand, and Y. Yamamoto (1998). Managing the software design documents with xml. In Proceedings of the 16th annual international conference on Computer documentation, ACM Press: New York, pp. 127-136.

[22] J. W. W. Wan, and G. Dobbie (2003). Extracting Association Rules from XML Documents using XQuery. In Proceedings of WIDM'03, New Orleans, Louisiana, USA, pp. 94-97.

[23] S. Wong, Y. Cai, and M. Dalton (2009). Detecting Design Defects Caused by Design Rule Violations. Report DU-CS-09-04, Drexel University.

[24] S. Wong, Y. Cai, M. Kim, and M. Dalton (2011). Detecting software modularity violations. In Proceedings of 33rd International Conference on Software Engineering (ICSE'11), pp. 411–420.

[25] G. Xu, and A. Rountev (2008). AJANA: A General Framework for Source-Code-Level Interprocedural Dataflow Analysis of AspectJ Software. In Proceedings of AOSD'08, Brussels, Belgium.

[26] J. Zhao (2002). Change Impact Analysis for Aspect-Oriented Software Evolution. In Proceedings of the 5th International Workshop on Principles of Software Evolution, Orlando, Florida, pp. 108-112.

Towards Crafting an Improved Functional Link Artificial Neural Network Based on Differential Evolution and Feature Selection

Ch. Sanjeev Kumar Dash and Ajit Kumar Behera
Silicon Institute of Technology, Silicon Hills
Patia, Bhubaneswar-751024, Odisha, India
E-mail: sanjeev_dash@yahoo.com, ajit_behera@hotmail.com

Satchidananda Dehuri
Department of Systems Engineering
Ajou University, San 5, Woncheon-dong
Yeongtong-gu, Suwon-443-749, South Korea
E-mail: satchi@ajou.ac.kr

Sung-Bae Cho
Soft Computing Laboratory
Department of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-gu
Seoul 120-749, South Korea
E-mail: sbcho@yonsei.ac.kr

Gi-Nam Wang
Department of Industrial Engineering
Ajou University, San 5, Woncheon-dong
Yeongtong-gu, Suwon-443-749, South Korea
E-mail: gnwang@ajou.ac.kr

Keywords: differential evolution, functional link artificial neural networks, classification, feature selection, genetic algorithms

Received: October 11, 2014

The proposed work describes an improved functional link artificial neural network (FLANN) for classification. The improvement in terms of classification accuracy of the network is realized through differential evolution (DE) and filter based feature selection approach. Information gain theory is used to filter out irrelevant features and provide relevant features to the functional expansion unit of FLANN as an input, which in turn maps low to high dimensional feature space for constructing an improved classifier. To fine tune the weight vector of the given network, differential evolution is used. The work is validated using skewed and balanced dataset retrieved from the University of California Irvine (UCI) repository. Our systematic experimental study divulges that the performance of the differential-evolution trained FLANN is promising than genetic algorithm trained FLANN, ISO-FLANN, and PSO-BP.

Povzetek: Predstavljena je gradnja klasifikacijske nevronske mreže, ki doseže boljše performanse z več novimi pristopi.

1 Introduction

Recently it is noticed that classification of big data [4] has demanded a great deal of attention. In this task, it is required to predict the value (the class label) of a user specified attribute based on the values of other predicting attributes. Although the task has been studied for many decades by the machine learning, statistics, and data mining communities but the complexity and shear size of the dataset creates lots of avenues in pursuit of perfection. Hence, an effort towards developing a smooth, accurate, and scalable classifier is always encouraging; it can face the challenge posed by the big data analysis. In this context, we urged that this work is a step towards handling big data, which has been plagued with many local optimal solutions and highly non-linear. Although, we have carried out our experimentation with the dataset obtained from the University of California, Irvine (UCI) repository [18] for validation, but it can be extended to handle big data.

Over the decades, neural networks [60] have been used as an alternative tool for approximating non-linearly separable boundary of classes in a classification problem. Pao et al. [41], have shown that FLANN may be conveniently used for function approximation and can be extended for classification with faster convergence rate

and lesser computational load vis-à-vis multi-layer perceptron (MLP) structure. With this motivation, several classifiers such as adaptive Particle Swarm Optimization-Back-propagation (PSO-BP) learning [13], improved swarm optimized FLANN (ISO-FLANN) Dehuri, et al. [15] have already been developed with certain efficacy. The FLANN is basically a flat network and the need of the hidden layer is removed and hence the learning algorithm used in this network becomes very simple. The functional expansion effectively increases the dimensionality of the input vector and hence the hyper planes generated by the FLANN provide greater discrimination capability in the input pattern space.

Feature selection can be broadly classified into two categories: i) filter approach (it depends on generic statistical measurement); and ii) wrapper approach (based on the accuracy of a specific classifier) [2]. In the proposed work, the feature selection is performed based on information gain theory (entropy) measure with a goal to select a subset of features that preserves as much as possible the relevant information found in the entire set of features. We know that the architectural complexity of FLANN [12] is directly proportional to the number of features and the functions considered for expansion of the given feature value. Therefore, for reducing the architectural complexity, we first select a subset of features (i.e., feature selection) [28, 29] using gain ratio and then apply the usual procedure of function expansion and training by differential evolution [56]. In this work, the remarkable performance of DE as a global optimizer on continuous error function minimization problems has been studied [8] in the classification by effectively learning the FLANN. DE has also become a powerful tool for solving optimization problems that arise in other application areas like finance, medical, image processing [62], automatic clustering of big unlabeled datasets, et cetera.

This paper is set out as follows. Section 2 gives an overview of FLANN network, feature selection, and differential evolution. In Section 3, the proposed method is discussed. Experimental setup, results, and analysis are presented in Section 4. Section 5 concludes the paper with a future line of research

2 Background

The background of the research work is presented in this Section. In Subsections 2.1 and 2.2, literatures study of FLANN as a classifier and predictor is discussed. Feature selection and its importance are the focus of Subsection 2.3. Differential evolution, a meta-heuristic computing paradigm is discussed in Subsection 2.4.

2.1 Review of Literature

FLANNs are higher-order neural networks without hidden units introduced by Klasser and Pao [30]. Despite of their linear nature, FLANNs can capture non-linear input–output relationships, provided that these are fed with an adequate set of polynomial inputs, or the functions might be a subset of a complete set of orthonormal basis functions spanning through n-dimensional representation space, are constructed out of the original input attributes [41]. FLANNs can be used for non-linear prediction and classification. Related to this context, Subsections 2.1.1 and 2.1.2 are briefing out some of the works on FLANNs for classification and non-linear prediction.

2.1.1 FLANNs for Classification

In [53] a genetic algorithm used to select an appropriate number of polynomials as a functional input to the FLANN has been applied to the classification problem. However, their main concern was the selection of the optimal set of functional links to construct the classifier. In contrast, the proposed method gives much emphasis on how to develop the learning skill of the classifier by using filtered feature vectors. Misra and Dehuri [36] have used a FLANN for classification problem in data mining with a hope to get a compact classifier with less computational complexity and faster learning. Hu and Tseng [63] have used the functional link net known as BpFLANN for classification of bankruptcy prediction. With a motivation to restrict certain limitations, Dehuri, et al. [12] have coupled genetic algorithm based feature selection with FLANN (GFLANN). In the sequel, Dehuri and Cho [13] have given a road map on FLANN and designed a new PSO-BP adaptive learning mechanism for FLANN. In [14], Dehuri and Cho have contributed another stimulating work on FLANN [14] in succession with an improved swarm optimized FLANN for classification [15].

2.2 FLANNs for Prediction

Pao et al., have presented a functional link neural network (CoFLANN) in [40] to learn the control systems. They have shown several beneficial properties of generalized delta rule network with hidden layer and back-propagation (BP) learning. Haring and Kok [20], have proposed an algorithm (ClFLANN) using evolutionary computation (specifically genetic algorithm and genetic programming) for the determination of functional links (one based on polynomials and another based on expression tree) in neural network. Patra et al. [44] have proposed a CeFLANN based on BP learning and applied to the problem of channel equalization in a digital communication channel. Haring et al. [21], have proposed a ClaFLANN to select and transform features using evolutionary computation and showed that this kind of selection of features is a special case of so-called functional links. Hussain et al. [25] have described a new approach for decision feedback equalizer (DFE) based on the functional-link neural network (DfFLANN). The structure is applied to the problem of adaptive equalization in the presence of inter-symbol interference (ISI), additive white Gaussian noise, and co-channel interference (CCI). The experimental results provide significantly superior bit-error rate (BER) performance characteristics as compared to the conventional methods. Chen et al. [6] have presented an adaptive implementation of the functional-link neural network

(AFLNN) architecture together with a supervised learning algorithm named Rank-Expansion with Instant Learning (REIL) that rapidly determines the weights of the network. The beauty of their proposed algorithm is one-shot training as opposed to iterative training algorithms in the literature. Dash et al. [9], have proposed an ElfFLANN with trigonometric basis functions to forecast the short-term electric load. Panagiotopoulos et al. [39] have reported better results by applying FLANN for planning in an interactive environment between two systems: the challenger and the responder. Patra et al. [44] have proposed a FLANN with BP learning (SiFLANN) for identification of non-linear dynamic systems. Moreover, Patra et al. [44] have used FLANN to adaptive channel equalization in a digital communication system with 4-QAM signal constellation named as QsFLANN. They have compared the performance of the FLANN with a multilayer perceptron (MLP) and a polynomial perceptron network (PPN) along with a conventional linear LMS-based equalizer for different linear and nonlinear channel models. Out of the three ANN equalizer structures, the performance of the FLANN is found to be the best in terms of MSE level, convergence rate, BER and computational complexity for linear as well as nonlinear channel models over a wide range of SNR and EVR variations. With the encouraging performance of FLANN [47, 48, 49], Patra et al. [45] have further motivated and came up with another FLANN known as IpFLANN with three sets of basis functions such as Chebyshev, Legendre, and power series to develop an intelligent model of the CPS involving less computational complexity. In the sequel, its implementation can be economical and robust. Park and Pao [43] have reported the performance of a holistic-styled word-based approach to off-line recognition of English language script. The authors have combined the practices of radial basis function neural net (RBNN) and the random vector functional-link net approach (RVFLANN) and obtained a method called the density-based random-vector functional-link net (DBRVFLANN). The combination is helpful in improving the performance of word recognition. A Chebyshev functional link artificial neural networks (CFLANN) is proposed by Patra et al. [49] for non-linear dynamic system identification. Sing et al. [54] has estimated the degree of insecurity in a power system by the proposed IeFLANN with a set of orthonormal trigonometric basis functions. An evolutionary search of genetic type and multi-objective optimization [34] such as accuracy and complexity of the FLANN in the Pareto sense is used to design a generalized FLANN (SyFLANN) with internal dynamics and applied to system identification. A reduced-decision feedback functional link artificial neural network (RDF-FLANN) structure for the design of a nonlinear channel equalizer in digital communication systems is proposed by Weng et al. [58]. Authors have reported that the use of direct decision feedback can greatly improve the performance of FLANN structures. Weng et al. [57], have proposed a reduced decision feed-back Chebyshev functional link artificial neural networks (RDF-CFLANN) for channel equalization. In [46], FLANNs with trigonometric polynomial functions (IsFLNN) are used in intelligent sensors for harsh environment that effectively linearizes the response characteristics, compensates for non-idealises and calibrates automatically.

Interval regression analysis has been a useful tool for dealing with uncertain and imprecise data. Since the available data often contain outliers, robust methods for interval regression analysis are necessary. Hu [24] has proposed a genetic-algorithm-based method (IraFLANN) for determining two functional-link nets for the robust nonlinear interval regression model: one for identifying the upper bound of data interval, and the other for identifying the lower bound of data interval.

2.3 FLANNs Classifier

The FLANN architecture [10, 47, 48, 49] uses a single layer feed forward neural network by removing the concept of hidden layers. The learning of a FLANN may be considered as approximating or interpolating a continuous, multivariate function *f*(X) by an approximating function $f_{W(X)}$. In FLANN a set of basis functions Φ and a fixed number of weight parameters W are used to represent $f_{W(X)}$. With a specific choice of a set of basis functions, the problem is to find the weight parameters W that provides the best possible approximation of 'f' on the set of input-output examples. So, the most important thing is that how to choose the basis functions to obtain better approximation.

Let us consider a set of basis function $\gamma = \{\Phi_i \in L(A)\}_{i\in I}$ with the following properties:

(i) Φ_1=1, (ii) the subset $\gamma_j = \{\Phi_i \in \gamma\}_{i=1}^{j}$ is a linearly independent set, i.e., if $\sum_{i=1}^{j}(w_i\Phi_i) = 0$, then $w_i = 0$ for all i=1,2,…j, and

(iii) $\sup_j \left[\sum_{i=1}^{j}\|\Phi_i\|_A^2\right]^{1/2} < \infty$.

Let $\gamma_N = \{\Phi\}_{i=1}^{N}$ be a set of basis functions to be considered for FLANN. Thus, the FLANN consists of *N* basis functions $\{\Phi_1, \Phi_2, ..., \Phi_N\} \in \gamma_N$ with the following input-output relationship for the j[th] output:

$$\hat{y} = \rho(s_j);\ s_j = \sum_{i=1}^{N}(w_{ji}.\Phi_i(X)), \quad (1)$$

where $X \in A \subset R^n$, i.e., $X = [x_1, x_2, ..., x_n]^T$ is the input pattern vector, $\hat{y} \in R^m$ i.e., $\hat{y} = [y_1, y_2, ..., y_m]^T$ is the output vector and $w_j = [w_{j1}, w_{j2}, ..., w_{jN}]$ is the weight vector associated with the j[th] output of the FLANN. The non-linear function $\rho(.) = \tanh(.)$ is used to transfer the weighted sum into desired output format of an input pattern.

Considering the *m*-dimension output vector, equation (1) can be written in matrix notation as follows:

$$S = W\Phi, \tag{2}$$

where W is an $(m \times N)$ weight matrix of the FLANN given by $W = [w_1, w_2, ..., w_m]^T$, $\Phi = [\Phi_1(X), \Phi_2(X), ..., \Phi_N(X)]^T$ is the basis function vector, and $S = [S_1, S_2, ..., S_N]^T$ is a matrix of linear outputs of the FLANN. The *m*-dimensional output vector $\hat{y}$ may be given by

$$\hat{y} = \rho(s) = f_w(X), \tag{3}$$

The training of the network is done in following way:

Let 'k' patterns be applied to the network in a sequence repeatedly. Let the training sequence be denoted by (X_k, y_k) and the weight of the network be $W(k)$, where the 'k' is also the iteration. Referring to equation (1) the j^{th} output of the FLANN at iteration *k* is given by:

$$\hat{y}(k) = \rho(\sum_{i=1}^{N}\left(w_{ji}(k)\Phi_i(X_k)\right)) = \rho(w_j(k)\Phi^T(X_k)), \tag{4}$$

for all $X \in A$ and *j*=1,2…,*m*, where $\Phi(X_k) = [\Phi_1(X_1), \Phi_2(X_2), ..., \Phi_N(X_k)]^T$.

Let the corresponding error be denoted by:

$$e_j(k) = y_j(k) - \hat{y}_j(k).$$

In words, the weighted sum of the functionally expanded features is fed to the single neuron of the output layer of the FLANN. The weights are optimized by the DE method during the process of training. The set of functions considered for function expansion may not be always suitable for mapping the non-linearity of the complex task. In such cases few more functions may be incorporated into the set of functions considered for expansion of the input data set.

However, dimensionality of many problems itself is very high and further increasing the dimensionality to a very large extent may not be an appropriate choice. So, this is one of the reasons, why we are carrying out this work.

2.4 Feature Selection

Feature selection (FS) [29] is an essential task to remove irrelevant and/or redundant features. In other words, feature selection techniques provide a way to select a subset of potential attributes or variables from a dataset. For a given classification problem, the network may become unbelievably complex if the number of the features used to classify the pattern increases very much. So the reason behind using FS techniques include reducing dimensionality by removing irrelevant and redundant features, reducing the amount of attributes needed for learning, improving algorithms' predictive accuracy, and increasing the constructed model's comprehensibility. After feature selection a subset of the original features is obtained which retains sufficient information to discriminate well among classes. The selection of features can be achieved in two ways:

Filter Method: The filter approach is independent of the learning algorithm, computationally simple, fast, and scalable. Using filter method, feature selection is done once and then can be provided as input to different classifiers. In this method features are ranked according to some criterion and the top k features are selected.

Wrapper model: This approach uses the method of classification itself to measure the importance of feature sets; hence the selected features depend on the classifier model used [26]. In this method a minimum subset of features is selected without learning performance deterioration.

Wrapper methods generally result in better performance than filter methods because the feature selection process is optimized for the classification algorithm to be used. However, wrapper methods are too expensive for large dimensional database in terms of computational complexity and time since each feature set considered must be evaluated with the classifier algorithm used. Filter based feature selection methods are in general faster than wrapper based methods.

2.5 Differential Evolution

Differential evolution (DE) [50, 55, 57] is a population based stochastic search algorithm. As a stochastic optimizer, it has the capability to handle non-linearity, non-convexity, multi-modality, and even dynamic characteristics of the problem. Unlike canonical GA, it typically operates on real valued individual encodings. Like GAs [35], DE maintains a pool of potential solutions which are then perturbed in an effort to explore yet better solutions to a problem in hand. In GAs, the individuals are perturbed based on *crossover* and *mutation*. However in DE, individuals are perturbed based on the difference of different individuals, borrowing ideas from the Nelder-Mead simplex method [50]. One of the advantages of this approach is that the resulting 'step' size and orientation during the perturbation process automatically adapts to the landscape of fitness function.

There are many variants of DE algorithms developed [8, 50] in past few years, the most classical variants is based on the *DE/rand/1/bin* scheme [55]. The different variants of the DE algorithm are described using the notation *DE/x/y/z*, where *x* specifies how the base vector is chosen (e.g., *rand*-if it is randomly selected, or *best*-if the best individual is selected), *y* is the number of difference vectors used, and *z* denotes the crossover scheme (*bin* for crossover based on independent binomial experiments, and *exp* for exponential crossover).

A pool of *n*, *d*-dimensional solution vectors $x_i = (x_{i1}, x_{i2}, ..., x_{id}), i = 1, 2, ..., n$ is randomly initialized and evaluated using a fitness function *f*(.). During the process of search, each individual (*i*) is iteratively refined. The following three steps are iterated one after another till desired optimum is reached.

i) **Mutation**: Create a donor vector which encodes a solution, using randomly selected members of the population.

ii) Crossover: Create a trial vector by combining the donor vector with *i*.

iii) Selection: By the process of selection, determine whether the newly-created trial vector replaces *i* in the population or not.

The pseudo-code of DE is illustrated as follows:

DE (*Scaled Factor* f_m, *Crossover Rate* C_r, *Pool Size n*)
{
INITIALIZATION: *Generate a population of n, d-dimensional solution vectors in the search space*.
DO
FOR *each i of the Population of individuals*
MUTATION: *Generate a donor vector* v_i *using equation (5).*
CROSSOVER: *Generate a trial vector* u_i *using equation (6).*
SELECTION: *Evaluate the trial vector* (u_i). *if* $f(u_i) > f(x_i)$ *(for maximization problem) then replace* x_i *by* u_i *else* x_i *will survive to next generation*.
END FOR
WHILE (*Termination Criterion Met*)
}

For the mutation of i^{th} individual of the population, three different individuals' x_{r1}, x_{r2}, and x_{r3} with $r1 \neq r2 \neq r3 \neq i$ will be randomly chosen from the pool to generate a new vector known as *donor* vector. The donor vector is described as follows.

$$v_i = \underbrace{x_{r1}}_{\substack{base \\ Vector}} + f_m.\underbrace{(x_{r2} - x_{r3})}_{\substack{Individial \\ Difference}}, \quad (5)$$

where, the scaling parameter f_m called mutation factor and a general setting for the parameter is $f_m \in [0,2]$. However, Storn and Price suggest $f_m \in [0.5,1]$ as such a setting may result in good optimization effectiveness.

Selecting three indices randomly imply that all individuals of the current pool have the same chance of being selected, and therefore influencing the creation of the difference vector. The mutation factor controls the amplification of the difference vector and in turn used to avoid stagnation of the search process. There are several alternative versions of the above process for creating a donor vector [for details see [8, 50]. In [62] a self-adaptive DE is used that can tune this scaling factor dynamically.

After the creation of the donor vector (v_i), a binomial crossover (*bin*) operates on the vector v_i and the target vector x_i to generate a trial vector in the following way.

$$u_{ij} = \begin{cases} v_{ij} & if\ (rand \le c_r)\ or\ \ (j = rand(1,2,...,d)) \\ x_{ij} & if\ (rand > c_r)\ and\ (j \neq rand(1,2,....,d)) \end{cases}, \quad (6)$$

where x_{ij}, v_{ij}, and u_{ij} are the *j*-dimensional components of the vectors x_i, v_i, and u_i, respectively; *rand* is a random number generated in the range (0, 1); is the user-specified crossover constant from the range (0, 1). The resulting trial (child) vector replaces its parent if it has higher fitness (a form of selection); otherwise the parent survives unchanged into the next iteration of the algorithm (shown in equation (7)).

$$x_i(t+1) = \begin{cases} u_i(t) & if\ (f(u_i(t)) > f(x_i(t))) \\ x_i(t) & otherwise \end{cases} \quad (7)$$

It is provided a comprehensive comparison of the performance of DE with a range of other optimizers, including GA, and report that the results obtained by DE are consistently as good as the best obtained by other optimizers across a wide range of problem instances in [50]. There are a number of reasons to integrate FLANN with DE. The first reason is to reduce local optimality during the training of FLANN. Although genetic algorithm coupled with FLANN has already been established to reduce local optimality while designing a classifier but DE has some merits over GA. Therefore, we are ignited to carry out this work. Secondly, the real encoding of DE solves the problem of encoding-decoding mapping. Thirdly, it can achieve faster convergence speed. Lastly DE does not undergo any complex process of parameter tuning and works very reliably with excellent overall results.

3 Proposed Method

With an objective to design a smooth and accurate classifier, the proposed approach is combining the idea of filter based feature selection and simple FLANN classifier [15]. It is a two phase method. In phase one, we are selecting a set of relevant features by using the entropy while in the second phase the weights of FLANN are trained using differential evolution. Figure 1 depicts the overall architecture of the approach.

In the first phase, we rank the features or attributes according to information gain ratio and then delete an

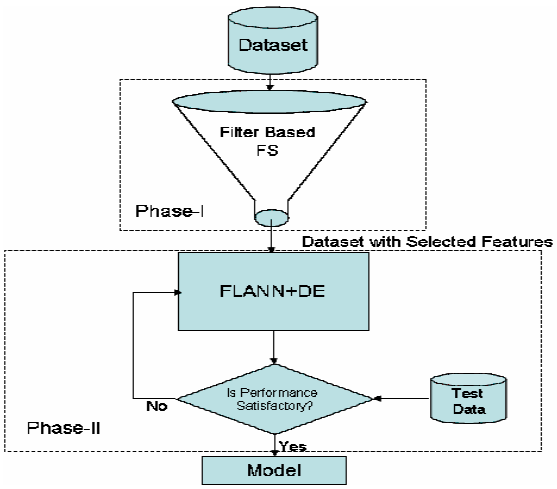


Figure 1: Architecture of Proposed Method.

appropriate number of features which have the least gain ratio [2]. The exact number of features deleted varies from dataset to dataset. The expected information needed to classify a tuple in D is given by equation (8),

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i), \quad (8)$$

where p_i is the non-zero probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_{i,D}|/|D|$. A log function to the base 2 is used, because the information is encoded in bits. *Info*(*D*) is the average amount of information needed to identify the class level of a tuple in D. *Info*(*D*) is also known as entropy of *D* and is based upon only the properties of classes.

For an attribute 'A' entropy "$Info_A(D)$" is the information still required to classify the tuples in D after partitioning tuples in D into groups only on its basis.

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j), \quad (9)$$

where *v* is the number of distinct values in the attribute A, |*D*| is the total number of tuples in *D* and $|D_j|$ is the number of repetitions of the j^{th} distinct value.

Information gain (*Gain*(*A*)) is defined as the difference between the original information requirement and new requirement (after partitioning on A) (refer equation 10)

$$Gain(A) = Info(D) - Info_A(D). \quad (10)$$

Information gain applies a kind of normalization to information gain using split information value defined analogously with $Info(D)$ as equation 11:

$$SplitInfo_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right). \quad (11)$$

This value represents the potential information generated by splitting the training data set, D, into *v* partitions, corresponding to the *v* outcomes of test on attribute A. For each outcome, it considers the number of tuples having the outcome with respect to the total number of tuples in D. The gain ratio is defined as in equation (12).

$$GainRatio(A) = Gain(A) / SplitInfo(A). \quad (12)$$

In summary, the feature selection is done in first phase using information gain ratio and then the dataset with reduced number of features is used for automatic training and determination of the parameters of FLANN using DE in the second phase.

In the second phase, we are focusing on the learning of the classifier. Here, differential evolution is employed to reveal the weight of the FLANN. This ensures efficient representation of an individual of DE. Since the performances of the FLANN mainly depend on weight; we just encode the weight into an individual for stochastic search. We have chosen a set of trigonometric functions as the basis function for functional expansion. The reason of choosing trigonometric functions for functional expansion is as follows:

Without loss of generality, for all the polynomials of N^{th} order with respect to an orthonormal system $\{\varphi_i(u)\}_{i=1}^{N}$ the best approximation in the metric space L^2 is given by the N^{th} partial sum of its Fourier series with respect to the system. Thus, the trigonometric polynomial basis functions provide a compact representation of the function in the mean square sense. However, when the outer product terms were used along with the trigonometric polynomials for functional expansion, better results were obtained in the case of learning the classifier.

Suppose the maximum number of trigonometric functions used to expand a particular feature is '*F*' and there are 'L' features selected for input to the network, then the size of the weight vector is defined as $K_{max} = L.(F+1)$, then the length of the individual is $K_{max}+B$. The structure of the individual is represented in Figure 2.

Figure 2: Structure of the Individual.

In other words, each individual has two constituent parts such as weight and bias.

The fitness function which is used to guide the search process is defined in equation (13).

$$E = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e(i)^2}, \quad e(i) = y(i) - \hat{y}(i), \quad (13)$$

where N is the total number of training sample $y(i)$ is the actual output of i^{th} pattern and $\hat{y}(i)$ is the estimated output of FLANN. The error and hence root mean square is denoted as e(i) and E respectively.

The algorithmic framework of FLANN-DE is described as follows:

Initially, a set of n_p individuals (i.e., i=1,2,..,n_p) is the size of the population) pertaining to networks weights and bias are created.

$$x_i^{(t)} = \left\langle x_{i1}^{(t)}, x_{i2}^{(t)}, ..., x_{id}^{(t)} \right\rangle, \quad i = 1, 2, \ldots, n_p$$

where $d = K_{max} + B$ and t is the iteration number.

At the start of the algorithm this n_p set of individuals is initialized randomly and then evaluated using the fitness function $f(.)$.

In each iteration, e.g., iteration t, for individual $x_i^{(t)}$ undergoes mutation, crossover, and selection as follows:

Mutation: for vector $x_i^{(t)}$ a perturbed vector $V_i^{(t+1)}$ called donor vector is generated according to equation (14):

$$V_i^{(t+1)} = x_{r1}^{(t)} + m_f \times (x_{r2}^{(t)} - x_{r3}^{(t)}), \quad (14)$$

where m_f is the mutation factor drawn from (0,2], the indices r_1, r_2, and r_3 are selected randomly from $\{1,2,3,\ldots,n_p\}$.

Crossover: The trial vector is generated as follows (equation (15)):

$$u_i^{(t+1)} = \left\langle u_{i1}^{(t+1)}, u_{i2}^{(t+1)}, \ldots, u_{id}^{(t+1)} \right\rangle,$$
$$u_{ij}^{(t+1)} = \begin{cases} v_{ij}^{(t+1)} & if\ (rand \le c_r)\quad or\ (i = rand(1,2,..,d) \\ x_{ij}^{(t)} & if\ (rand > c_r)\quad and\ (i \ne rand(1,2..,d) \end{cases}, \quad (15)$$

where j=1, 2, …,d, *rand* is a random number generated in the range (0,1) c_r is the user specified crossover constant from the range (0,1) and *rand*(1,2,…,d) is a randomly chosen index from the range (1,2,…,d). The random index is used to ensure that the trial solution vector differs by at least one element from $x_i^{(t)}$. The resulting trial (child) solution replaces its parent if it has higher accuracy (a form of selection), otherwise the parent survives unchanged into the next iteration of the algorithm.

Finally, we use selection operation and obtain target vector $x_i^{(t+1)}$ as follows in equation (16):

$$x_i^{(t+1)} = \begin{cases} u_i^{(t+1)} & if \quad f(x_i^{(t+1)}) \le f(x_i^{(t)}) \\ x_i^{(t)} & otherwise \end{cases}. \quad (16)$$

4 Experimental Study

The data set and experimental parameter setting are discussed in Subsection 4.1. Results are analyzed in Subsection 4.2.

4.1 Description of Dataset Parameters

The data set used to test the proposed method obtained from the UCI machine learning repository [18]. Four balanced and unbalanced datasets have been chosen to validate the proposed method. The details about the four data sets are given below.

Iris: This data set includes 150 instances and each having 4 attributes, excluding the class attribute. The instances are uniformly classified into 3 classes (i.e., every instance either belongs to class 1, or 2, or 3). Class 1 has 50 instances, class 2 contains 50, and remaining instances (i.e., 50) are belong to class 3. None of the attributes contain any missing values. All attributes are continuous.

Wine: This data set includes 178 instances and each having 13 attributes, excluding the class attribute. The instances are classified into 3 classes (i.e., every instance either belongs to class 1 or 2 or 3) in an almost balanced way. Class 1 has 59 instances, class 2 contains 71, and remaining instances (i.e., 48) are belong to class 3. None of the attributes contain any missing values. All attributes are continuous.

Lymphography: This data set includes 148 instances and each having 19 attributes including the class attribute. The instances are classified into 4 classes (i.e., every instance either belongs to class 1, or 2, or 3, or 4). Class 1 has 2 instances, class 2 contains 81, class 3 contains 61 and remaining instances (i.e., 4) belong to class 4. None of the attributes contain any missing values. All attributes are continuous. However, this dataset is purely unbalanced.

Stalog(Heart): There are 270 instances, 13 attributes, and 2 classes in this dataset. Class 1 has 151 instances and Class 2 has 119 instances. None of the attributes contain any missing values. The distributions of samples into different classes are almost balanced.

Pima: This data set includes 768 instances and each having 8 attributes along with one class attribute. The instances are classified into 2 classes (i.e., every instance either belongs to class 1 or 2). Class 1 has 500 instances and class 2 contains 268. None of the attributes contain any missing values. All attributes are continuous. However, the distribution of samples into various classes is not balanced.

In our experiment, every dataset is randomly divided into two mutually exclusive parts: 50% as training sets and 50% as test sets. The parameters' value used for validating our proposed method is listed in Table 1. These parameters were obtained after several rounds of independent runs. However, the number of iterations are varies from dataset to dataset.

Table 1: Parameters used for simulation

Parameter		Iris	Wine	Lympho - graphy	Stalog (heart)	Pima
Population		50	50	50	50	50
Mutation		0.2	0.4	0.3	0.4	0.4
Crossover		0.8	0.8	0.6	0.8	0.8

In the case of Iris dataset, an ideal number of iterations is lies within the range of 40~50. However, it varies from 400~500 in the case of Lymphography, Statlog (Heart), and Pima. In the case of Wine the ideal number of iteration can varies from 1000~1200. Similarly, the parameters setting of the methods ISO-FLANN and PSO-BP have been fixed as suggested in the respective literatures.

4.2 Result Analysis

The experimental test results are presented in Table 2. The accuracy in terms of percentage of test samples correctly classified using proposed method and method ISO-FLANN, PSO-BP, and method proposed in [12] are

given in columns 2, 3, 4, and 5 of Table 2. In all the cases, 1/3rd of the features have been removed.

In our comparison, it is noticed that the accuracy obtained from our proposed method is better than the method proposed in [12], ISO-FLANN, and PSO-BP, moreover a paired t-test has been performed to judge the performance all the algorithms properly. With the 5% significance level, the critical value of t is 0.43 and it is not coming under the specified range, so the null hypothesis is rejected.

Table 2: Testing accuracy of proposed method vs. method proposed in ISO-FLANN (Dehuri, et al., 2012), PSO-BP (Zhang, et al., 2007) and (Dehuri, Mishra, and Cho, 2008).

Dataset	**Proposed Method**	**ISO-FLANN**	**PSO-BP**	**Method Proposed in (**Dehuri, Mishra, and Cho, 2008**)**
Iris	98.33	97.62	97.12	97.33
Wine	93.10	92.32	91.54	90.45
Lymphography	87.50	86.9	85.65	77.08
Statlog (heart)	86.57	85.46	84.77	84.45
Pima	79.20	78.86	76.88	72.14

The features which are identified to remove during the filter process are listed in Table 3. Further, the mapping of numeric and actual name of the features are shown in Table 4 to avoid confusion or ordering.

Table 3: Filtered Attributes of Datasets.

Dataset	**Attributes of the Dataset**	**Attributes Removed**
Iris	1~4	2
Wine	1~13	3,4,5,8
Lymphography	1~18	1,6,9,12,16,17
Statlog(heart)	1~13	1,4,6,7
Pima	1~8	1,3

Table 4: Features ordering of all datasets.

Dataset	**Attributes Ordering**	**Name**
Iris	1	Sepal Length
	2	Sepal Width
	3	Petal Length
	4	Petal Width
Wine	1	Alcohol
	2	Malic acid
	3	Ash
	4	Alcalinity of ash
	5	Magnesium
	6	Total phenols
	7	Flavanoids
	8	Nonflavanoid phenols
	9	Proanthocyanins
	10	Color intensity
	11	Hue
	12	OD280/OD315 of diluted wines
	13	Proline
Lymphography	1	Lymphatics: normal, arched, deformed, displaced
	2	Block of affere: no, yes
	3	Bl. of lymph. c: no, yes
	4	Bl. of lymph. s: no, yes
	5	By pass: no, yes
	6	Extravasates: no, yes
	7	Regeneration of: no, yes
	8	Early uptake in: no, yes
	9	Lym.nodes dimin: 0-3
	10	Lym.nodes enlar: 1-4
	11	Changes in lym.: bean, oval, round
	12	Defect in node: no, lacunar, lac. marginal, lac. central
	13	Changes in node: no, lacunar, lac. margin, lac. central
	14	Changes in stru: no, grainy, drop-like, coarse, diluted, reticular, stripped, faint
	15	Special forms: no, chalices, vesicles
	16	Dislocation of: no, yes
	17	Exclusion of no: no, yes
	18	No. of nodes in: 0-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, >=70
Statlog (heart)	1	Age
	2	Sex
	3	Chest pain type (4 values)
	4	Resting blood pressure
	5	Serum cholestoral in mg/dl
	6	Fasting blood sugar > 120 mg/dl
	7	Resting electrocardiographic results (values 0,1,2)
	8	Maximum heart rate achieved
	9	Exercise induced angina
	10	Oldpeak = ST depression induced by exercise relative to rest
	11	The slope of the peak exercise ST segment
	12	Number of major

		vessels (0-3) colored by flourosopy
	13	Thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
Pima	1	Number of times pregnant
	2	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
	3	Diastolic blood pressure (mm Hg)
	4	Triceps skin fold thickness (mm)
	5	2-Hour serum insulin (mu U/ml)
	6	Body mass index (weight in kg/(height in m)^2)
	7	Diabetes pedigree function
	8	Age (years)

The error rate obtained from the proposed method varies over a number of iterations of IRIS, Lymphography, WINE, Statlog (Heart), and PIMA are illustrated in Figures 3, 4, 5, 6, and 7.

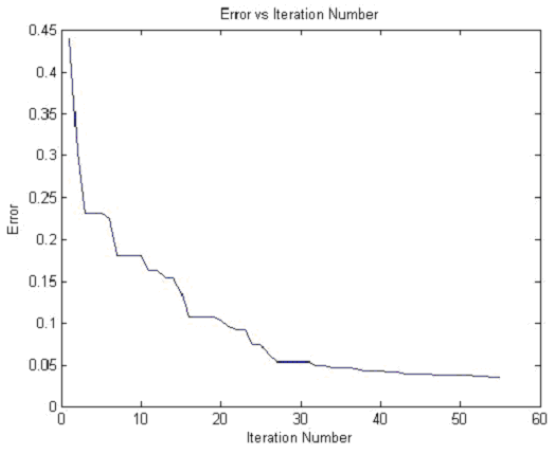


Figure 3. Iteration Number vs Error Obtained from IRIS Dataset.

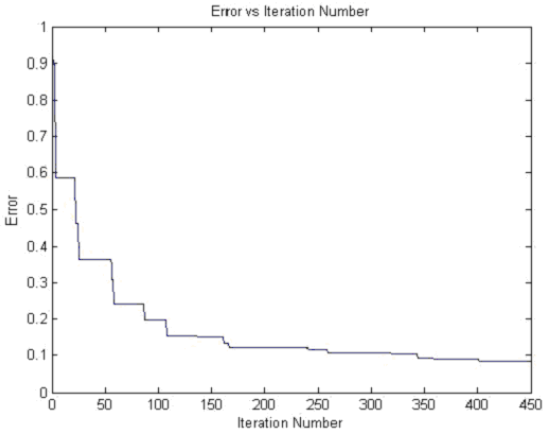


Figure 4. Iteration Number vs. Error Obtained from Lymphography Dataset.

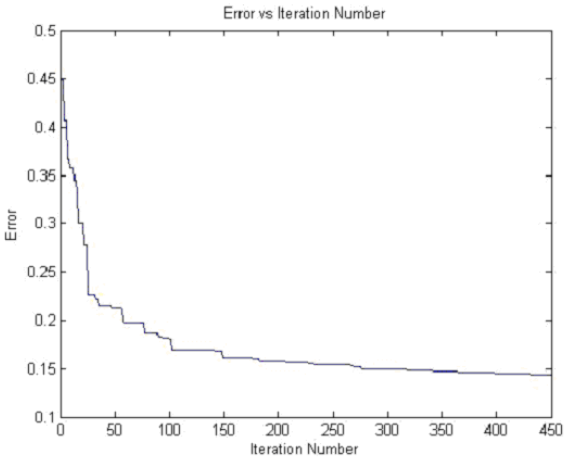


Figure 5. Iteration Number vs. Error Obtained from PIMA Dataset.

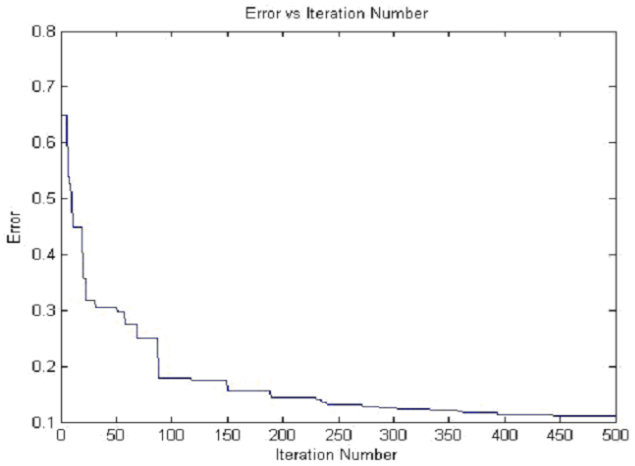


Figure 6. Iteration Number vs Error Obtained from Statlog (Heart) Dataset.

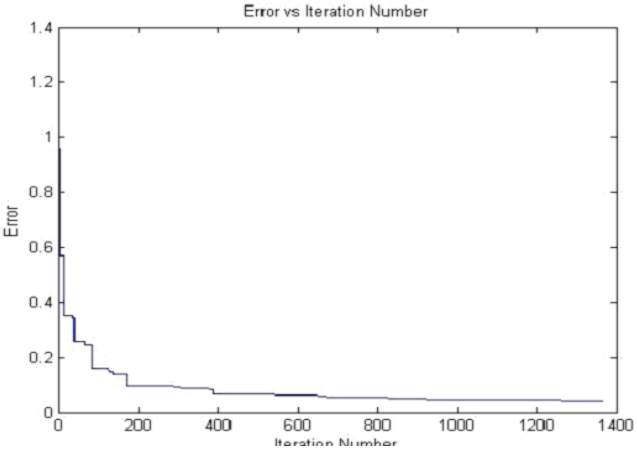


Figure 7. Iteration Number vs. Error Obtained from WINE Dataset.

Further, it is interestingly noticed that if we do not eliminate any features from the dataset then the accuracy of the results hardly makes any difference, except wine dataset. In the case of wine dataset the accuracy obtained with feature selection is poor than without feature selection. With respect to optimal mutation factor the results of proposed method without feature selection for different datasets are described in Table 5. However, it is better than ISO-FLANN, PSO-BP, and method proposed in [12].

Name of Data set	Mutation factor	Accuracy in percentage
Iris	0.3	98.23
Wine	0.4	98.51
Lymphography	0.3	87.50
Stalog(heart)	0.4	86.56
Pima	0.4	78.91

Table 5: Results Obtained from proposed method without feature selection.

Figures 8, 9, 10, 11, and 12 illustrate the performance of the proposed method without feature selection over different mutation rate. In the case of Iris and Lymphography the accuracy of the proposed classifier dropped after the mutation rate 0.3. Similarly, in the case of Wine, Pima, and Statlog the accuracy dropped after the mutation rate 0.4. Hence, our experimental study recommends the mutation rate setup mentioned in Table 5.

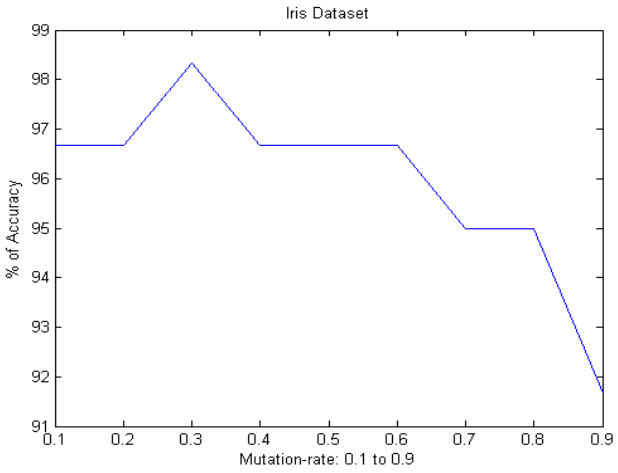


Figure 8: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from IRIS.

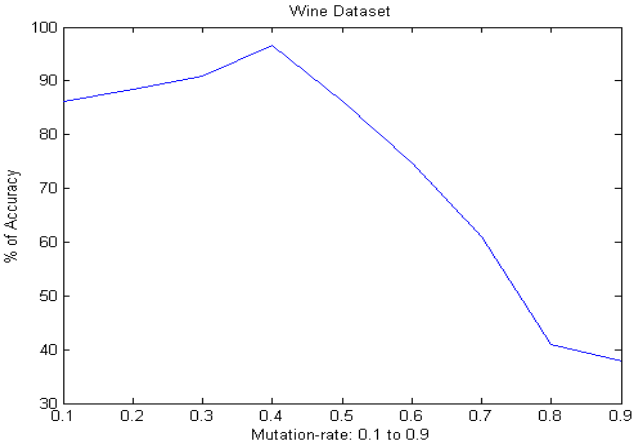


Figure 9: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from WINE.

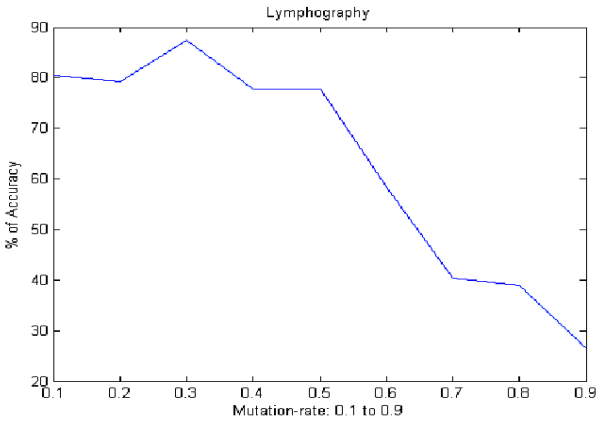


Figure 10: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from Lymhography.

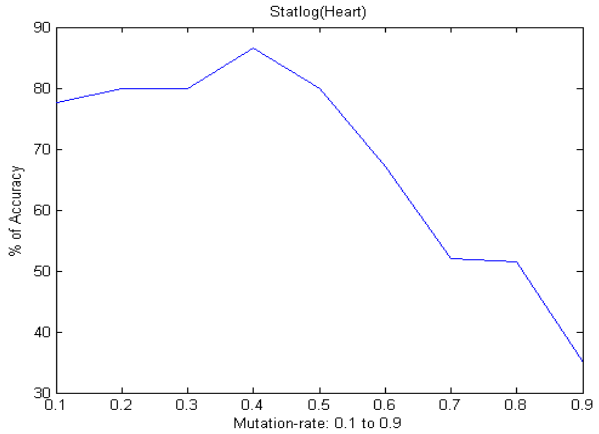


Figure 11: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from Statlog (Heart).

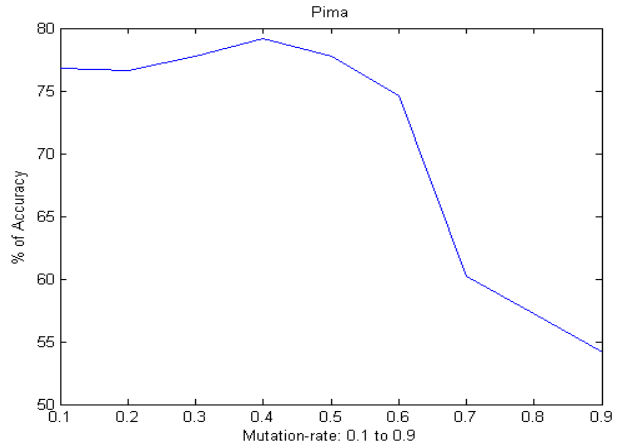


Figure 12: Mutation factor (0.1to 0.9) vs. accuracy (without feature selection) obtained from PIMA.

Figures 13, 14, 15, 16, and 17 demonstrate; how accuracy of the proposed classifier with filter based feature selection varies over different mutation rates.

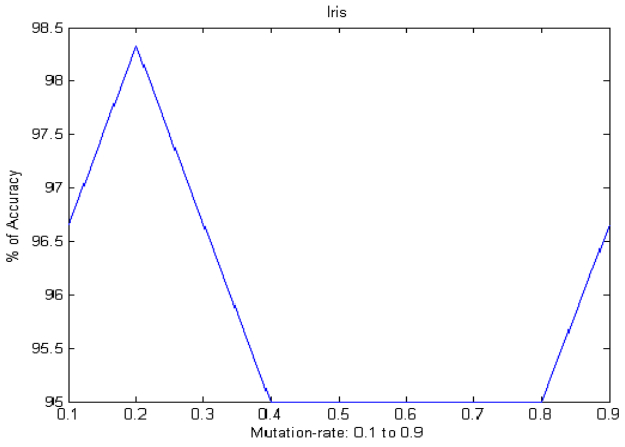


Figure 13. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) Obtained from IRIS.

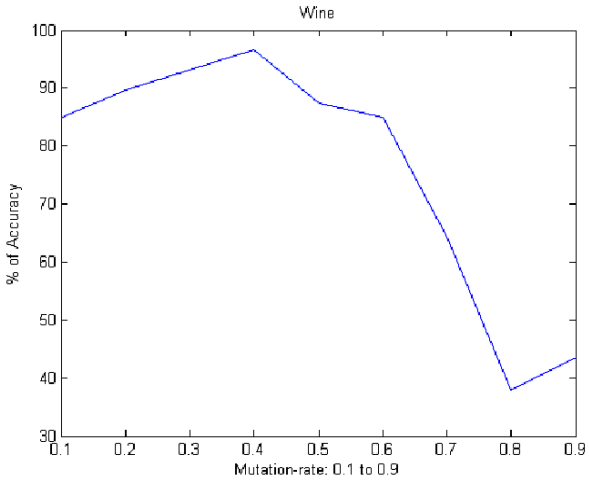


Figure 14. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) obtained from WINE.

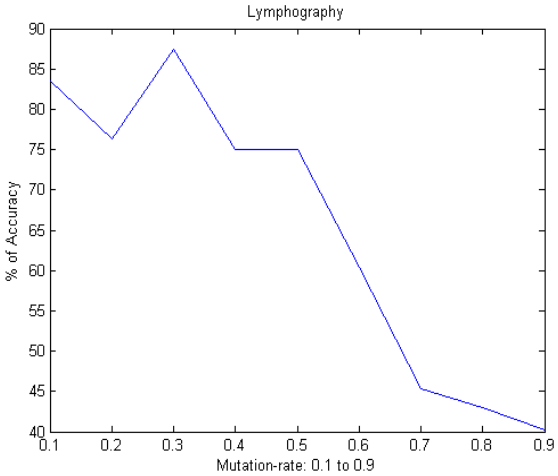


Figure 15. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) obtained from Lympography.

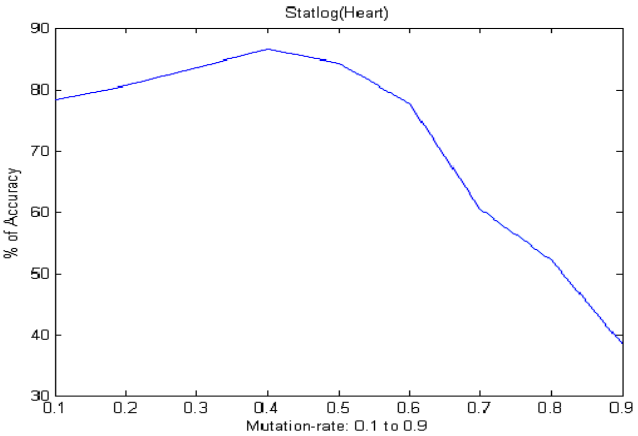


Figure 16. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) obtained from Statlog (heart).

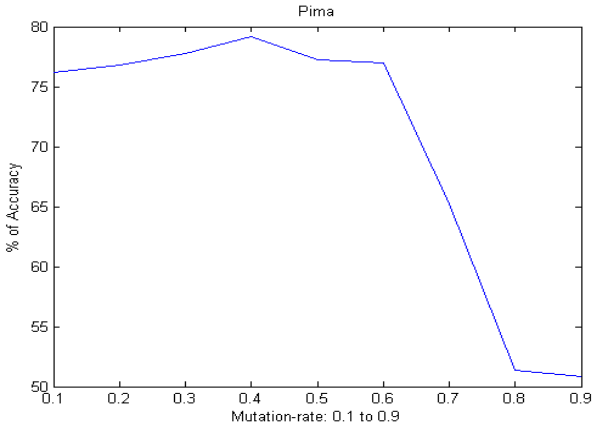


Figure 17. Mutation factor (0.1to 0.9) vs. accuracy (with feature selection) obtained from PIMA.

In the case of Iris the accuracy of the proposed classifier with mutation rate 0.2 is promising. In the case of Wine and Lymphography the accuracy is dropped after the mutation rate 0.3. Similarly the accuracy of Statlog and Pima has been dropped after the mutation rate 0.4.

5 Conclusion

An integrated framework of differential evolution and FLANN along with a filter based feature selection has been crafted in this work to classify unknown patterns. The experimental results confirm that the combination of filter based feature selection and training of FLANN using differential evolution obtains accurate and smooth classification. Further, it has been observed that if we reduce 1/3rd of total attributes by the process of filter approach, then the network shows an improvement and constancy in accuracy. Finally, we have compared the experimental outcomes obtained from this study with its rival proposed by Dehuri et al. [12], ISO-FLANN, and PSO-BP, we then conclude that the accuracy draws a very sharp edge between the proposed method and the rest of the methods consider for comparison. Our future line of research includes: i) validation of its accuracy and scalability in big data analysis and ii) dealing with noise and uncertainty of the dataset by suitable integration with other soft computing paradigm.

References

[1] Abu-Mahfouz, I.-A. (2005). A Comparative Study of Three Artificial Neural Networks for the Detection and Classification of Gear Faults. *International Journal of General Systems*, 34(3), pp. 261–277.

[2] Aruna, S., Nandakishore, L. V., and Rajagopalan, S. P. (2012). A Hybrid Feature Selection Method based on IGSBFS and Naïve Bayes for the Diagnosis of Erythemato-Squamous Diseases. *International Journal of Computer Applications* (0975-8887), 41(7).

[3] Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5(4), 537-550.

[4] Bifet, A. (2013). Mining Big Data in Real Time. *Informatica (Slovenia)*, 37(1), 15-20.

[5] Carvalho, D. R., & Freitas, A. A. (2004). A hybrid decision tree/genetic algorithm method for data mining. *Information Sciences*, 163(1), 13-35.

[6] Chen, C. P., LeClair, S. R., & Pao, Y. H. (1998). An incremental adaptive implementation of functional-link processing for function approximation, time-series prediction, and system identification. *Neurocomputing*, 18(1), 11-31.

[7] Das, M., Roy, R., Dehuri, S., and Cho, S.-B. (2011). A New Approach to Associative Classification Based on Binary Multi-objective Particle Swarm Optimization. *International Journal of Applied Meta-heuristic Computing*, 2(2), pp. 51-73.

[8] Das, S. and Suganthan, P. N. (2011). Differential Evolution: A Survey of the State-of- the- Art. *IEEE Transactions on Evolutionary Computation*, 15(1), pp. 4-31.

[9] Dash, P. K., Liew, A. C., and Satpathy, H. P. (1999). A Functional Link Neural Network for Short Term Electric Load Forecasting. *Journal of Intelligent and Fuzzy Systems*, 7(3), pp. 209–221.

[10] Dash, P. K., Satpathy, H. P., Liew, A. C., and Rahman, S. (1997). Real-Time Short-Term Load Forecasting System Using Functional Link Network. *IEEE Transactions on Power Systems*, 12(2), pp. 675-68.

[11] Dash, S. K., Behera, A., Dehuri, S., and Cho, S.-B. (2013). Differential Evolution Based Optimization of Kernel Parameters in Radial Basis Function Networks for Classification. *International Journal of Applied Evolutionary Computation*, .4(1), pp. 56-80.

[12] Dehuri, S., Mishra, B. B., & Cho, S. B. (2008). Genetic feature selection for optimal functional link artificial neural network in classification. In *Intelligent Data Engineering and Automated Learning–IDEAL 2008*, pp. 156-163.

[13] Dehuri, S. and Cho, S.-B. (2009). A Comprehensive Survey on Functional Link Neural Networks and an adaptive PSO--BP learning for CFLNN. *Neural Computing and Applications*, 19(2), pp. 187-205.

[14] Dehuri, S. and Cho, S.-B. (2010). Evolutionary Optimized Features in Functional Link Neural Networks for Classification. *Expert Systems with Applications*, 37(6), pp. 4379-4391.

[15] Dehuri, S., Roy, R., Cho, S.-B., and Ghosh, A. (2012). An Improved Swarm Optimized Functional Link Artificial Neural Network (ISO-FLANN) for Classification. *The Journal of Systems and Software*, 85, pp.1333-1345.

[16] Derrac, J., Garcia, S., and Herrera, F. (2010). A Survey on Evolutionary Instance Selection and Generation. *International Journal of Applied Meta-heuristic Computing*, 1(1), pp. 60-92.

[17] Forerest, S. (1993). Genetic Algorithms: Principles of Natural Selection Applied to Computation. *Science*, 261, pp. 872-888.

[18] Frank, A. and Asuncion, A. (2010). *UCI Machine Learning Repository http://archive.ics.uci.edu/ml)*, Irvine, CA: University of California, School of Information and Computer Science.

[19] Goldberg, D. E. (1989). Genetic Algorithm in Search Optimization and Machine Learning, *Addison-Wesley*.

[20] Haring, B. and Kok, J. N. (1995). Finding Functional Links for Neural Networks by Evolutionary computation. In: van de Merckt, T., et al. (eds) BENELEARN1995, *Proceedings of the Fifth Belgian– Dutch Conference on Machine Learning, Brussels, Belgium*, pp. 71–78

[21] Haring, S., Kok, J. N., and van Wezel, M. C. (1997). Feature Selection for Neural Networks Through Functional Links Found by evolutionary Computation. *In Advances in Intelligent Data Analysis (IDA-97).LNCS* 1280, pp. 199–210.

[22] Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall.

[23] He, X., Zhang, Q., Sun, N., & Dong, Y. (2009). Feature selection with discrete binary differential evolution. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, Vol. 4, pp. 327-330).

[24] Hu, Y.-C. (2008). Functional Link Nets with Genetic Algorithm Based Learning for Robust Non-Linear Interval Regression Analysis. *Neurocomputing*. 72(7). pp. 1808-1816.

[25] Hussain, A., Soraghan, J. J., & Durrani, T. S. (1997). A new adaptive functional-link neural-network-based DFE for overcoming co-channel interference. *Communications, IEEE Transactions on*, *45*(11), 1358-1362.

[26] Karegowda, A. G., Manjunath, A. S., and Jayaram, M. A. (2010). Comparative Study of Attribute Selection Using Gain Ratio and Correlation Based Feature Selection. *International Journal of Information Technology and Knowledge Management*, 2(2), pp. 271-277.

[27] Khusba Rami, N., Ahmed, A., and Adel, A. (2008). Feature Subset Selection Using Differential Evolution. In *Proceedings of 15th International conference on Advances in Neuro-Information Processing, Springer-Verlag, Berlin, Heidelberg, part-I*, pp.103-110,.

[28] Khushaba, R. Al-Ani, A., AlSukker, A., and Al-Jumaily, A. (2008). A Combined Ant Colony and Differential Evolution Feature Selection Algorithm. *Ant Colony Optimization and Swarm Intelligence*, pp. 1-12.

[29] Khushaba, R., Al-Ani, A., and Al-Jumaily, A. (2011). Feature Subset Selection using Differential Evolution and a Statistical Repair Mechanism. *Expert Systems with Applications*, 38(9), pp. 11511-11526.

[30] Klasser, M.S. and Pao, Y. H. (1988). Characteristics of the Functional Link Net: A Higher Order Delta Rule Net. *IEEE Proceedings of 2nd Annual International Conference on Neural Networks, San Diago, CA*, pp.507-513,

[31] Krishnaiah, D., Prasad, D. M. R., Bono, A., Pandiyan, P. M., and Sarbatly, R. (2008). Application of Ultrasonic Waves Coupled with Functional Link Neural Network for Estimation of carrageenan Concentration. *International Journal of Physical Sciences*, 3(4), pp. 90–96.

[32] Liu, H. and Motoda, H. (2002). On Issues of Instance Selection. *Data Mining and Knowledge Discovery*, 6, pp. 115-130.

[33] Majhi, B. and Shalabi, H. (2005). An Improved Scheme for Digital Watermarking using Functional Link Artificial Neural Network. *Journal of Computer Science*, 1(2), pp. 169–174.

[34] Marcu, T. and Koppen-Seliger, B. (2004). Dynamic Functional Link Neural Networks Genetically Evolved Applied to System Identification. In *Proceedings of ESANN'2004, Bruges (Belgium)*, pp. 115–120.

[35] Michalewicz, Z. (1998). *Genetic Algorithm + Data structure = Evolution Programs*. Springer Verlag, New York.

[36] Misra, B. B. and Dehuri, S. (2007). Functional Link Neural Network for Classification Task in Data mining. *Journal of Computer Science*, 3(12), pp. 948–955.

[37] Nayak, S. C., Misra, B. B., and Behera, H.S. (2012). Index prediction with Neuro-Genetic Hybrid network: A Comparative Analysis of Performance Computing. In *Proceedings of International Conference on Communication and Applications (ICCCA)*, pp. 1-6.

[38] Nayak, S. C., Misra, B. B., and Behera, H. S. (2012). Evaluation of Normalization Methods on Neuro-Genetic Models for Stock index Forecasting. In *Proceedings of 2012 World Congress on Information and Communication Technologies (WICT)*, pp. 602-607.

[39] Panagiotopoulos, D. A., Newcomb, R. W., and Singh, S. K. (1999). Planning with a Functional Neural Network Architecture. *IEEE Trans. Neural Network*, 10(1), pp. 115–127.

[40] Pao, Y.-H., Phillips, S. M. (1995). The Functional Link Net Learning Optimal Control. *Neurocomputing*, 9(2), pp. 149–164.

[41] Pao, Y. H. and Takefuji, Y. (1992). Functional Link Net Computing: Theory, system, Architecture and Functionalities. *IEEE Computer Journal*, pp. 76–79.

[42] Pao, Y. H., Phillips, S. M., and Sobajic, D. J. (1992). Neural-net Computing and Intelligent Control Systems. *International Journal of Control*, 56(2), pp. 263–289.

[43] Park, G. H. and Pao, Y. H. (2000). Unconstrained Word-Based Approach for Off-Line Script Recognition using Density Based Random Vector

Functional Link Net. *Neurocomputing*, 31(1), pp. 45–65.

[44] Patra, J. C., Pal, R. N., Baliarsingh, R., and Panda, G. (1999). Non-Linear Channel Equalization for QAM Signal Constellation using Artificial Neural Networks. *IEEE Transactions on Systems, Man, Cybernetics-Part B: Cybernetics*, 29(2), pp. 262–271.

[45] Patra, J. C. and van den Bos, A. (2000). Modelling of an Intelligent Pressure Sensor using Functional Link Artificial Neural Networks. *ISA Transaction*, 39(1), pp. 15–27.

[46] Patra, J. C., Goutam, C., and Subahas, M. (2008). Functional Link Neural Networks-Based Intelligent Sensors for Harsh Environments. *Sensors and Transducers Journal*, vol. 90, pp. 209–220.

[47] Patra, J. C. and Pal, R. N. (1995). A Functional Link Neural Network for Adaptive Channel Equalization. *Signal Processing*, 43(2), pp. 181–195.

[48] Patra, J. C., Pal, R. N. Chatterji, B. N., and Panda, G. (1999). Identification of Nonlinear Dynamic Systems using Functional Link Artificial Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(2), pp. 254-262.

[49] Patra, J. C. and Kot, A. C. (2002). Nonlinear Dynamic System Identification using Chebyshev Functional Link Artificial Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 32(4), pp. 505-511.

[50] Price, K., Storn, R., and Lampinen, J. (2005). Differential Evolution: *A Practical Approach to Global Optimization*, Springer-Verlag.

[51] Purwar, S., Kar, I. N., and Jha, A. N. (2007). On-line System Identification of Complex Systems using Chebyshev Neural Networks. *Applied Soft Computing*, 7(1), pp. 364–372.

[52] Roy, R., Dehuri, S., and Cho, S. B. (2011). A Novel Particle Swarm Optimization Algorithm for Multi-objective Combinatorial Optimization Problem. *International Journal of Applied Meta-heuristic Computing*, 2(4), pp. 41-57.

[53] Sierra, A., Macias, J. A., and Corbacho, F. (2001). Evolution of Functional Link Networks. *IEEE Transactions on Evolutionary Computation*, 5(1), pp. 54–65.

[54] Sing, S. N., Srivastava, K. N. (2002). Degree of Insecurity Estimation in a Power System using Functional Link Neural Network. *European Transactions on Electrical Power*, 12(5), pp. 353–359.

[55] Storn, R., & Price, K. (1995). *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces* (Vol. 3). Berkeley: ICSI.

[56] Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4), 341-359.

[57] Weng, W. D., Yong, C. S., and Lin, R. C. (2007). A Channel Equalizer using Reduced Decision Feedback Chebyshev Function Link Artificial Neural Networks. *Information Sciences*, 177(13), pp. 2642–2654.

[58] Weng, W. D., Yen, C. T. (2004). Reduced Decision Feed-back FLANN Non-Linear Channel Equaliser for Digital Communication Systems. *IEEE Proceedings-Communications*, 151(4), pp. 305–311.

[59] Yan, Z., Wang, Z. and Xie, H. (2008). The Application of Mutual Information Based Feature Selection and Fuzzy LS-SVM based Classifier in Motion Classification. *Computer Methods and Programs in Biomedicine*, vol.90, pp. 275-284.

[60] Zhang, G. P. (2000). Neural Networks for Classification: A Survey. *IEEE Transactions on Systems, Man, Cybernetics-Part C: Application and Reviews*, 30(4), pp. 451–461.

[61] Zhang, J. R., Zhang, J., Lok, T. M., and Lyu, M. R. (2007). A Hybrid Particle Swarm Optimization-Back Propagation Algorithm for Feed-Forward Neural Network Training. *Applied Mathematics and Computation*, vol. 185, pp. 1026-1037.

[62] Ghosh, A., Datta, A., and Ghosh, S. (2013). Self-Adaptive Differential Evolution for Feature Selection in Hyperspectral Image Data. *Applied Soft Computing*, 13(4), pp. 1969-1977.

[63] Hu, Y. C. and Tseng, F. M. (2007). Functional-Link Net with Fuzzy Integral for Bankruptcy Prediction. *Neurocomputing*, 70(16), pp. 2959– 2968.

Multimodal Score-Level Fusion Using Hybrid GA-PSO for Multibiometric System

Cherifi Dalila and Hafnaoui Imane,
Institute of Electrical and Electronic Engineering
University of Boumerdes, Algeria
E-mail: dacherifi@yahoo.fr, hafmane@hotmail.com

Nait-Ali Amine
LiSSi, EA 3956, Paris-Est Creteil (UPEC)
Creteil, France
E-mail: naitali@u-pec.fr

Keywords: multibiometric, multimodal, fusion, score level, genetic algorithm, particle swarm optimization, hybrid, GA-PSO

Received: May 16, 2014

Due to the limitations that unimodal systems suffer from, Multibiometric systems have gained much interest in the research community on the grounds that they alleviate most of these limitations and are capable of producing better accuracies and performances. One of the important steps to reach this is the choice of the fusion techniques utilized. In this paper, a modeling step based on a hybrid algorithm, that includes Particle Swarm Optimization and Genetic Algorithm, is proposed to combine two biometric modalities at the score level. This optimization technique is employed to find the optimum weights associated to the modalities being fused. An analysis of the results is carried out on the basis of comparing the EER accuracies and ROC curves of the fusion techniques. Furthermore, the execution speed of the hybrid approach is discussed and compared to that of the single optimization algorithms, GA and PSO.

Povzetek: Predstavljena je nova optimirna metoda za iskanje uteži pri kombiniranju dveh virov informacij za biometrično prepoznavo.

1 Introduction

It is becoming increasingly apparent that a unimodal system using a single biometric trait is not sufficient to meet a number of system requirements imposed by several large-scale authentication applications. The limitation of unimodal systems, such as noisy sensor data, intra-classvariations, non-universality, vulnerability to spoof attacks and more, can lower the performance of the system, and make it more susceptible to refusing a legitimate user and jeopardizing personal security. Multibiometric systems seek to alleviate some of these drawbacks by consolidating the evidence presented by multiple biometric sources. These systems are expected to significantly improve the recognition performance of a biometric system besides improving population coverage, deterring spoof attacks, and reducing the failure-to-enroll rate. Multibiometric Fusion can be implemented in different scenarios including the type of fused sources and the level at which the fusion occurs. The sources can be multiple-sensors data, multiple-samples, multiple-algorithms, or multiple-modalities.

As for the levels, Sanderson and Paliwal [1] proposed classifying fusion techniques into two categories: pre-mapping and post-mapping fusion. Pre-mapping fusion techniques, such as sensor-level and feature-level fusion, perform fusion before matching. Post-mapping fusion techniques, such as rank-level, decision-level, and match score-level fusion, perform fusion after matching. In this paper, our work is focused on the fusion of multimodalities at the score level. This scenario is extensively studied in literature because of the relatively easy access to information at this level, and the fusion of the scores output by the different matchers[2]. This offers the best trade-off between accessibility and fusion convenience.

Paper contribution: we propose the use of a hybrid algorithm GA-PSO to optimize the weights assigned to the different biometric modalities used in the fusion at the score level.

The idea of the hybrid GA-PSO is to take advantage of both algorithms so as to gain in time performance and obtain a Multibiometric system with an optimum accuracy.

Paper structure: The rest of the paper is structured as follow: We present some of the previous works in literature that tackled this problem in the next section. Section 3 gives a brief overview of GA and PSO as well as some essential definitions. In section 4, we describe how the hybrid GA-PSO works and how it is used to obtain optimum biometric weights. Section 5 covers our experiments including the results obtained and a brief discussion. Our conclusions are highlighted in section 6.

2 Literature review

In a comparison study, Damousis and al. [3] used four machine learning techniques to fuse face and voice modalities at the matching level; mainly Gaussian Mixture Models (GMMs), Artificial Neural Networks (ANNs), Fuzzy Expert Systems (FESs), and Support Vector Machines (SVMs). Their research concluded that although all four techniques performed well, SVM gave the best accuracies.

The Sum Rule was proposed by Ross et al.[2] to fuse face, fingerprint, and hand geometry modalities. In order to compare this technique, Wang et al. [4] proposed using the Weighted Sum Rule by assigning weights to iris and face score modalities based on their false accept rate (FAR) and false reject rate (FRR). They concluded that the Weighted Sum Rule performs better at increasing the accuracy of recognition than the Simple Sum Rule. Various techniques were studied in order to assign said weights with varying levels of accuracy and performance. A recent trend has been the inclusion of optimization techniques in the fusion process in the hopes of obtaining the optimum of the biometric performance. Genetic algorithms (Gas) have seen a special interest. In the works of Alford and Hansen [5], a fusion of face and periocular biometrics at the score level based on Genetic and evolutionary computations (GEC) was achieved. Their work showed that better accuracies could be reached using this technique. Giot and al. [6] proposed a faster technique to compute the EERs of fused modalities as a fitness function for a Genetic Algorithm. Particle Swarm Optimization (PSO) was used in the works of Raghavendra and al. [7] in order to fuse near infrared and visible images for improved face verification. Mazouni and al. [8] did a comparison in performance of some Multibiometric fusion techniques on face and voice modalities. In their study, GA and PSO were proven to give the best accuracies, especially with degraded datasets. SVM in these cases gave the worst performances.

The work presented in this paper builds on these previous findings and increases the performance of the implemented systems. Since the recognition systems work with thousands of individuals, reducing the computation times is essential. The proposed approach, GA-PSO, strives to achieve this while keeping the performances at their highest. To our knowledge, no previous work employed a hybrid GA-PSO to fuse biometric modalities at the score level in order to gain good accuracies with better computational times.

3 Multimodal score level fusion

During score level fusion, scores are combined to generate a single scalar score which is later used to make the final decision. There are several combination schemes to achieve this. These include statistical rule-based methods such as Simple Sum, Max rule, Min rule, Product rule and Weighted Sum.

3.1 Score Normalization

With the methods mentioned above, score normalization is required before fusion of scores. Anil Jain and al. [9] showed in their work that both min-max and z-score methods are sufficient techniques but they are very sensitive to outliers. On the other hand, *tanh* normalization method, introduced by Hampel et al. [10] is both robust and efficient. For this purpose, and in our work, the *tanh-estimators* normalization rule was employed.

Given a matching score S_i, the normalized score $\tilde{S}_i$ is computed using the following equation:

$$\tilde{S}_i = \frac{1}{2}\left\{tanh\left(0.01\left(\frac{S_i - \mu_{GH}}{\sigma_{GH}}\right)\right) + 1\right\} \quad (1)$$

Where $\mu_{GH}$and σ_{GH} are the mean and standard deviation estimates, respectively, of the genuine score distribution as given by Hampel estimators.

3.2 Genetic Algorithm and Particle Swarm Optimization

In this work, the focus is on finding the optimum weights w_m for fusion of m modalities by weighted sum which is defined by:

$$S_{f_i} = \sum_{m=1}^{M} w_m \times S_i^m \quad (2)$$

Given that $w_m \in [0,1]$ and $\sum_{m=1}^{M} w_m = 1$.

Genetic algorithm is a well-known and frequently used evolutionary computation technique. This method was originally developed by John Holland et al.[11]. The GA is inspired by the principles of genetics and evolution, and mimics the reproduction behavior observed in biological populations.

In GA, a candidate solution for a specific problem is called an *individual* or a *chromosome* and consists of a linear list of genes. GA begins its search from a randomly generated population of designs that evolve over successive generations (iterations). To perform its optimization-like process, the GA employs three operators to propagate its population from one generation to another.

1) ***Selection:*** In which the GA considers the principal of "survival of the fittest" to select and generate individuals that are adapted to their environment.
2) ***Crossover:*** It mimics mating in biological populations. The crossover operator propagates features of good surviving designs from the current population into the future population, which will have a better fitness value on average.
3) ***Mutation:*** It promotes diversity in population characteristics. The mutation operator allows for global search of the design space and prevents the algorithm from getting trapped in local minima [11].

Particle Swarm Optimization is one of the recent evolutionary optimization methods. This technique was originally developed by Kennedy & Eberhart [12] in order to solve problems with continuous search space. PSO uses social rules to search in the design space by controlling the trajectories of a set of independent particles. The position of each particle, x_i representing a particular solution of the problem, is used to compute the value of the fitness function to be optimized. In fact, the main PSO operator is the velocity update, v_i, that takes into account the best position, in terms of fitness value reached by all the particles during their paths during its search g_{best}, and the best position that the agent itself has reached p_{best}, resulting in a migration of the entire swarm towards the global optimum.

At each iteration, the particle moves around according to its velocity and position; the cost function to be optimized is evaluated for each particle in order to rank the current location. The velocity of the particle is then stochastically updated according to [13]

$$v_i^{t+1} = k\begin{pmatrix} \omega v_i^t + C_1 r_1 (p_{best} - x_i^t) \\ +C_2 r_2 (g_{best} - x_i^t) \end{pmatrix} \quad (3)$$

After, the particle position is updated according to

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (4)$$

Where:

ω Inertia weight, a parameter controlling the flying dynamics.

r_1, r_2 random variables in the range $[0, 1]$.

C_1, C_2 positive constants controlling the related weighting of corresponding terms.

k Constriction parameter introduced by Clerc and al. [14].

4 The proposed hybrid GA-PSO approach

Although GAs have been successfully applied to a wide spectrum of problems, using GAs for large-scale optimization could be very expensive due to its requirement of a large number of function evaluations for convergence. Compared to GA, PSO has some attractive characteristics. It has constructive cooperation between particles; that is, particles in the swarm share information among themselves. On the other hand, a drawback of PSO is that the swarm may prematurely converge. The underlying principle behind this problem is the fast rate of information flow between particles, resulting in the creation of similar particles with a loss in diversity that increases the possibility of being trapped in local optima.

To deal with all these misgivings, and seeing as both GA and PSO work with an initial population of solutions and combining the searching abilities of both methods seems to be a reasonable approach, we propose a new algorithm, denoted as GA-PSO, that combines the evolutionary natures and social interactions of both algorithms.

To understand the workings of the algorithm, Figure 1 depicts a schematic representation of the proposed hybrid GA-PSO. As can be seen, GA and PSO both work with the same initial population. The hybrid approach picks N initial individuals that are randomly generated. The N individuals are sorted by fitness, and, according to a user defined probability P_k, the set is divided into two sub-sets$\{\psi_G, \psi_P\}$. The top set ψ_P is used to adjust the particles using the PSO algorithm. The other set ψ_G is fed into the real-coded GA to create new individuals by selection, crossover and mutation operations. Both resulting populations are combined into one single population of N individuals, which are then sorted in preparation for repeating the entire run.

In our experiments, and in terms of multimodal fusion, the hybrid algorithm generates an initial population of size N which consists of the weights $w_i$defined in equation (2). In this work, we will fuse two modalities at a time to create fusion scores which makes $m = 2$.

The fitness function is defined as the Equal Error Rate (EER). As a reminder, the EER is the point at which the error rates FAR and FRR are equal. The goal is to minimize the value of the EER. For every set of individuals$(\mathrm{w}_1, \mathrm{w}_2)$, the EER of the fused scores S_f is computed. Knowing that the best fitness is the one with the smallest EER, the individuals are then rearranged and sorted. The whole set is split into two sets which will go through the selection, crossover and mutation processes in case of GA, and velocity and position update in case of PSO. Evaluation of the fitness costs of the "offspring" is once again run and the weights to produce the minimum EER value is picked as optima. If the stopping criteria are not satisfied yet, this procedure is repeated until one of the conditions is met. This is summarized in **Algorithm 1**.

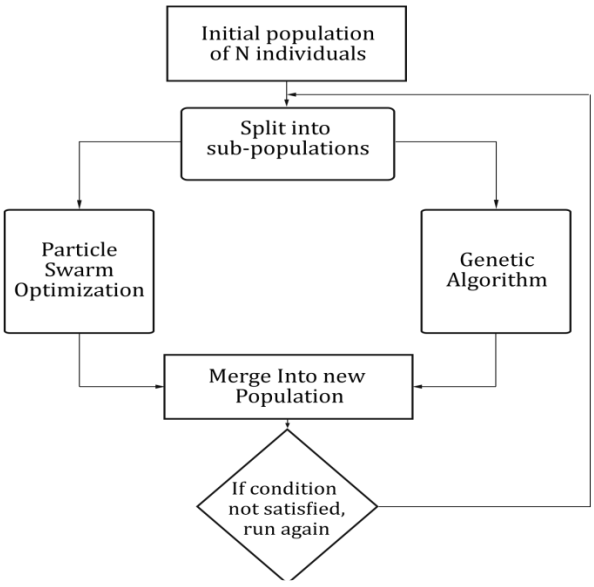


Figure 1: Scheme representation of the Hybrid GA-PSO Algorithm.

Algorithm 1: *Hybrid GA-PSO to find optimized fusion weights*

1. *Initialize parameter values*
2. *Generate random initial population (weights)*
3. ***While*** $k < iter_{max}$ ***do***
4. *Evaluate then sort fitness function for every individual based on EER*
5. ***For*** *i=1:m*
6. *Update particle's personal best* p_{best} *and global best* g_{best}
7. *Update particle's velocity and position according to eqs. (3) and (4)*
8. ***End for***
9. ***For*** *i=m+1: end*
10. *Select parents to reproduce*
11. *Generate children though crossover and mutation*
12. ***End for***
13. *Merge the two resulting sub-populations into one population*
14. ***If*** *(stopping criteria)* ***then***
15. *Go to 18*
16. ***End if***
17. ***End while***
18. ***Return*** *individuals (fusion weights) that give the best EER*

5 Results and Discussion

5.1 Experiment Setup

Three publicly available Multibiometric databases were used in order to validate the fusion techniques. The NIST BSSR1 Set 1 [15] consists of sets of raw output similarity scores from 517 users of faces and both left and right index live-scan fingerprints coming from the same person. XM2VTS database [16] is built on the XM2VTS face and speech multimodal database, respecting the Lausanne Protocols I and II (LP1 and LP2). LP1 has eight baseline systems and LP2 has five baseline systems. In here, we only deal with the LP1 dataset. The BANCA dataset [17] contains matcher scores of face and speech. There are seven different protocols: Mc, Md, Ma, Ua, Ud, G and P.

In order to validate the aforementioned algorithms and their effectiveness when dealing with Multibiometrics, we split the databases into two separate sets:

- **The training set** which serves to compute the biometric reference of each matcher. In other words, we train the algorithms to attain the optimal weights for each matcher.
- **The testing set** which serves to validate the results of the training by computing the performance of the fusion with the obtained weights.

In our experiments, the hybrid algorithm which combines properties of both GA and PSO runs on the parameters summarized in Table.1.

Parameter	Value
Initial Population size	*50*
Maximum iterations	*50*
Splitting probability	P_k*=0.6*
Crossover probability	P_C*=1*
Mutation probability	P_m*=0.05*
Inertia factor	*w=1*
C_1 and C_2	C_i*=2.05*
Constriction factor	*k=0.73*

Table 1: The parameter values used in the hybrid GA-PSO.

5.2 Experimental Results

To compare the performances of the biometric systems, the EER values and Receiver Operating Characteristic (ROC) curves are studied. Table 2 presents the EER values of the single modalities involved in the fusion from each database.

To evaluate its performance, the hybrid is compared to the classical combination rules as well as the single optimization techniques, GA and PSO. Table 3 summarizes the results we obtained from the experiments. Before applying the rules on these scores, they have all been put under the same range {0, 1} using the *tanh-normalization* scheme. The best performance in each of the fused modalities is shown in **bold**.

From the first look, an improvement in accuracy is clearly observed between unimodal and multimodal systems, regardless of the fusion rule applied. In Table 3, even the best matcher in the NIST, Face-C with EER = 4.39%, is outperformed by a simple Max-rule, with an EER = 3.66%.

Figure 2 plots the ROC curves of fused scores using the classical combination rules. We observe that among all these rules, *Simple Sum* gives the better performance even when dealing with degraded data, as is the case of the BANCA Ua subset with (EER = 10.4%).

What interests us is the *Weighted Sum* where the weights associated to the different modalities are optimized through the hybrid GA-PSO. In Figure 3, the ROC curves of fused scores using Simple Sum are plotted against those using GA-PSO. We notice that although *Simple Sum* gave the best results previously, it is outperformed by the hybrid GA-PSO in every dataset. This is not only in terms of EER. From the same figure, we can see that, even when considering the FAR and FRR values, GA-PSO gives better rates.

These results are confirmed in Table 3, where compared to the best EER obtained from Simple Sum in the NIST dataset with the (FaG-FiR) pair (EER = 1.21 %), the improvement in accuracy is quite apparent where the optimizations give a better optimized EER (= 0.43%).

NIST BSSR1				XM2VTS				BANCA					
Face – G	Finger – R	Face – C	Finger – L	Face 1	Speech 2	Face 5	Speech 3	Face G	Speech G	Face Md	Speech Md	Face Ua	Speech Ua
5.69	4.39	5.52	7.91	1.81	6.61	6.57	4.51	11.32	1.98	10.58	4.33	28.5	15.1

Table 2: EER (%) of unimodal Biometric modalities from NIST, XM2VTS and BANCA

	NIST BSSR1		XM2VTS		BANCA		
Fusion Technique	FaG – FiR	FaC – FiL	F1S2	F5S3	F – S (G)	F – S (Md)	F – S(Ua)
Max	5.49	3.66	1.45	3.06	2.19	5.45	15.4
Min	5.52	7.91	1.81	6.46	7.32	5.61	28.5
Product	2.70	4.77	1.64	5.66	2.35	3.36	16.9
Simple Sum	1.21	1.00	1.24	3.67	1.82	3.37	10.4
Genetic Algorithm	0.44	**0.75**	**0.87**	1.88	1.07	**2.24**	**10.4**
Particle Swarm O.	0.62	**0.75**	**0.87**	**1.85**	1.07	**2.24**	11.1
Hybrid GA-PSO	**0.43**	**0.75**	**0.87**	**1.85**	**0.91**	**2.24**	**10.4**

Table 3: EERs (%) of fused scores using fusion techniques

Running Time	Genetic Algorithm	Particle Swarm opt.	Hybrid GA-PSO
Time to run 50 iterations	105	220	315
Time to reach a global min.	76	38	12

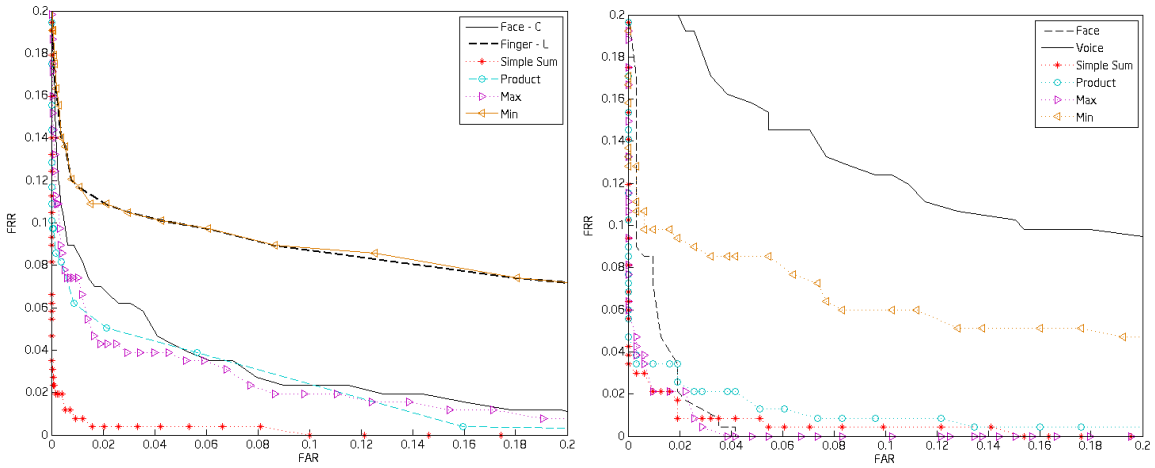
Table 4: Running time of optimization algorithms in (sec).

Although the performances of GA, PSO and the hybrid GA-PSO are closely similar in most datasets, the employment of the hybrid GA-PSO always reaches optimum weights which in turn gives the best EER values, to the contrary of GA and PSO, which sometimes tend to get stuck on local minima. We notice that even with the degraded data, the execution of this hybrid optimization technique provides good performance rates.

5.3 Discussion

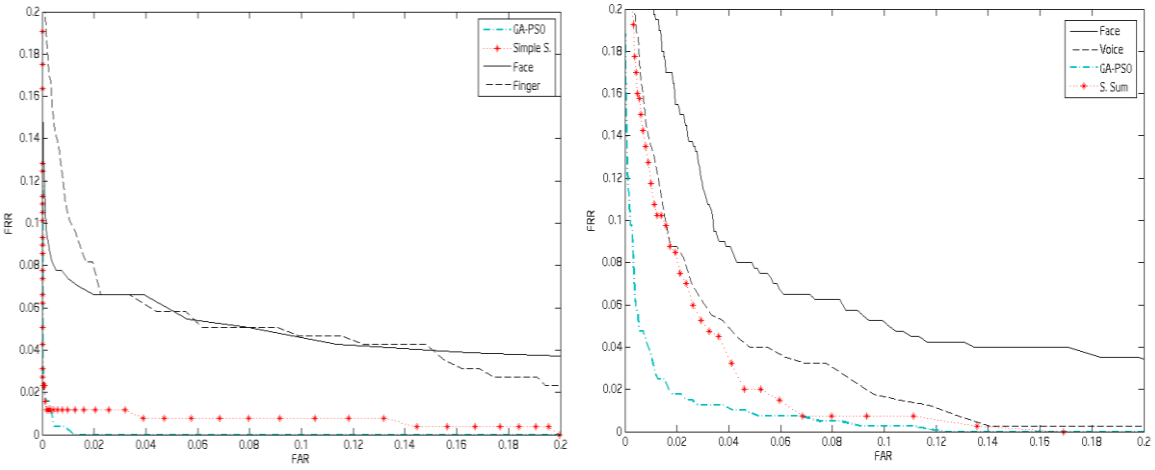
When it comes to comparing the optimization techniques to each other, there are not one but many points to consider. It is clear from the results discussed in the previous section and as can be observed in Figure 4, that GA, PSO, and GA-PSO mostly result in the same best accuracies. But they differ in other aspects such as the *time consumption* (see Figure 5). Genetic Algorithm, due to the fact that it covers large search spaces, has a larger computational time. On the other hand, we have PSO that, as a consequence of its fast operations, consumes less computational time but converges quickly to local minima. The hybrid GA-PSO takes advantage of both algorithms where it gains in computational time, by adding the benefit of fast search property of PSO, and still covers the large search space efficiently. This is observed in Figure 5, where the cost function is plotted against the number of iterations run by all three algorithms. It can clearly be noted, with the XM2VTS dataset, that GA-PSO takes much fewer iterations (#iterations = 2) to reach the global optimum than either PSO (#iterations = 9) or GA (#iterations = 38). Table 4 puts in value the amount of time in CPU-time that each algorithm takes to be executed for 50 iterations and the time to reach a global minimum. It seems, from a first look, that the hybrid algorithm gives the least favorable running time. That is quite logical since GA-PSO computes the cost function three times in one iteration while PSO computes it twice, and GA, once. But when taking into consideration that it takes much fewer iterations to reach a global optimum, it is actually faster than the two other algorithms. After many runs of these programs, it has been noticed that, although GA and PSO mostly give good results, they would occasionally get stuck in local minima, as is the case in Figure.5.b with the NIST dataset. It appears at first glance that GA reached a good place faster than the other algorithms. But in fact, it reached a local minimum and got stuck there. Be that as it may, after giving it more time, it did reach the same global minimum.

On the other hand, the hybrid GA-PSO is observed to always converge to a global point in the shortest time.



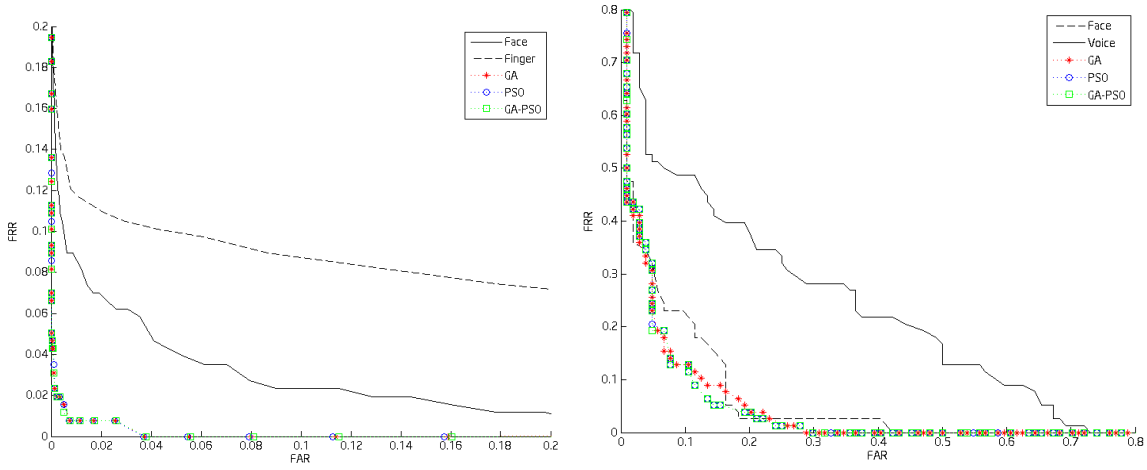
(a) (b)

Figure 2**:** ROC curves of fused scores using classical combination rules from (a) NIST (b) BANCA.



(a) (b)

Figure 3**:** ROC curves of fused scores using hybrid GA-PSO from (a) NIST (b) XM2VTS.



(a) (b)

Figure 4**:** ROC Curves of fused scores from (a) NIST (b) BANCA Databases using Optimization Techniques.

- XM2VTS - - NIST -

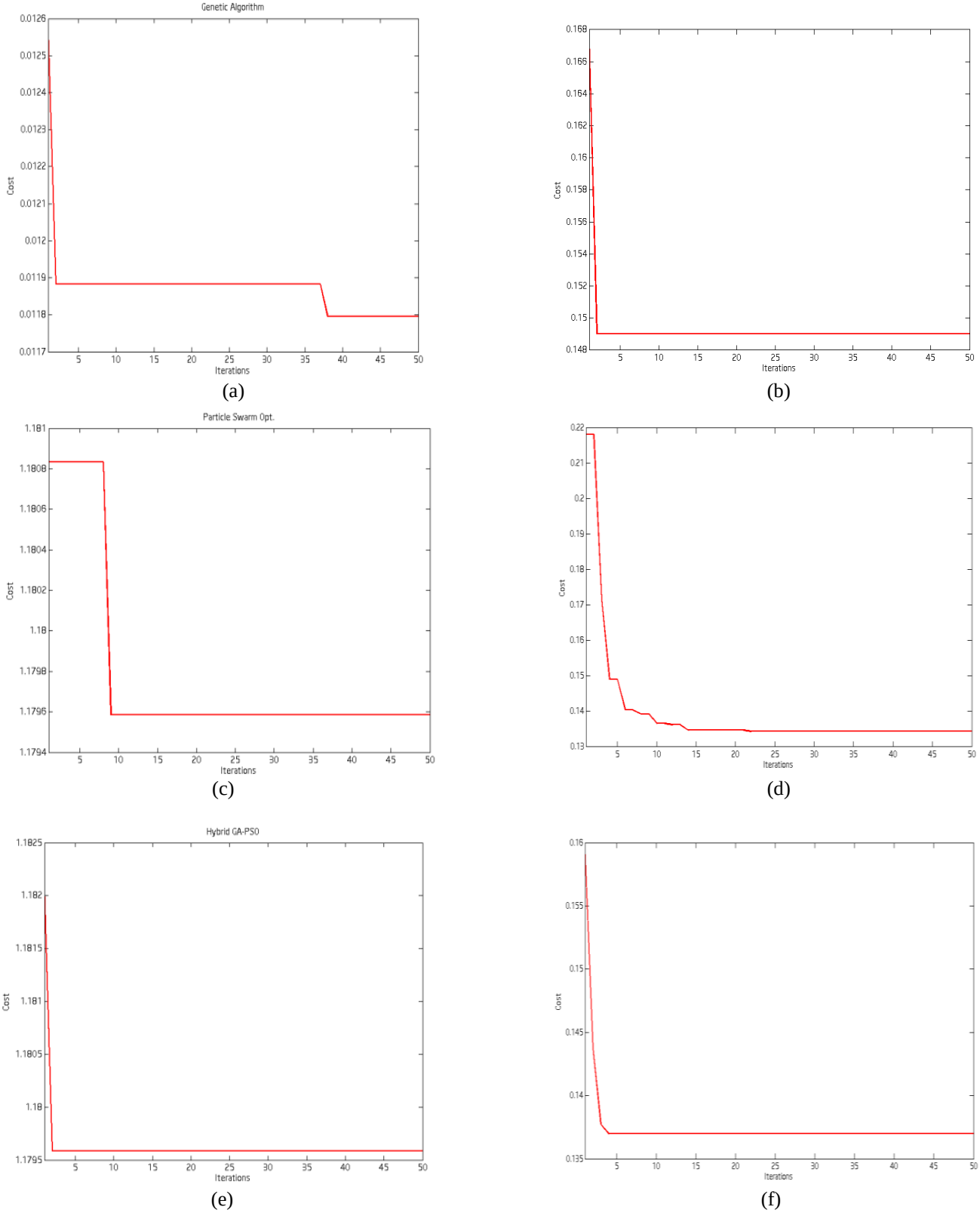


Figure 5**:** Cost Function vs. Number of Iterations for (a) (b) Genetic Algorithm (c) (d) Particle Swarm Optimization (e) (f) Hybrid GA-PSO.

6 Conclusion

This paper proposes a hybrid GA-PSO approach to combining biometric modalities at the score level. With the Weighted Sum rule, the role of the hybrid is to optimize the weights associated with the fused modalities to reach optimum EER values. A normalization based on the tanh-normalization scheme is performed beforehand to put the score modalities on a same unified range. The performance of the hybrid is compared to that of the classical combination rules and the single GA and PSO algorithms. Our results show that the GA-PSO was successful in obtaining much better accuracies on the three different public biometric databases as compared to the classical rules. The time execution of the optimization techniques is also studied. We observe that the GA-PSO outperforms the single GA and PSO in terms of computational time where we find that since the hybrid takes advantage of the properties of both GA and PSO, it assures that the optimum is reached and in the least number of iterations. This makes the hybrid GA-PSO a faster and more robust technique.

References

[1] C. Sanderson and K. K. Paliwal, *"On the Use of Speech and Face Information for Identity Verification,"* IDIAP Research Report 04-10, Martigny, Switzerland, 2004.

[2] A. A. Ross, K. Nandakumar, and A. K. Jain, *Handbook of Multibiometrics*, vol. 6, no. ISBN-13:978–0–387–22296–7.Springer-Verlag, pp. 74–75, 2006.

[3] I. G. Damousis and S. Argyropoulos, *"Four Machine Learning Algorithms for Biometrics Fusion: A Comparative Study,"* Appl. Comput. Intell. Soft Comput., vol. 2012, pp. 1–7, 2012.

[4] Y. Wang, T. Tan and A. K. Jain, *"Combining Face and Iris Biometrics for Identity Verification"*, Proc. 4th Int`l Conf. on Audio- and Video-Based Biometric Person Authentication (AVBPA), pp. 805-813, Guildford, UK, June 9-11, 2003.

[5] A. Alford, C. Hansen, G. Dozier, K. Bryant, J. Kelly, T. Abegaz, K. Ricanek, and D. L. Woodard, *"GEC-based multi-biometric fusion,"* IEEE Congr. Evol. Comput., pp. 2071–2074, Jun. 2011.

[6] R. Giot and C. Rosenberger, *"Genetic programming for multibiometrics,"* Expert Syst. Appl., vol. 39, no. 2, pp. 1837–1847, Feb. 2012.

[7] R. Raghavendra, B. Dorizzi, A. Rao, and G. Hemantha Kumar, *"Particle swarm optimization based fusion of near infrared and visible images for improved face verification,"* Pattern Recognit., vol. 44, no. 2, pp. 401–411, Feb. 2011.

[8] M. Romaissaa and R. Abdellatif, *"On Comparing Verification Performances of Multimodal Biometrics Fusion Techniques"* Int. J. Comput. Appl., vol. 33, no. 7, pp. 24–29, 2011.

[9] A. Jain, K. Nandakumar, and A. Ross, *"Score normalization in multimodal biometric systems,"* Pattern Recognit., vol. 38, no. 12, pp. 2270–2285, Dec. 2005.

[10] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust statistics: the approach based on influence functions*, vol. 114. John Wiley & Sons, 2011.

[11] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[12] J. Kennedy and R. Eberhart, *"Particle swarm optimization,"* in *Proceedings* of ICNN'95-International Conference on Neural Networks, vol. 4, pp. 1942–1948, 1995.

[13] A. T. Al-Awami, A. Zerguine, L. Cheded, A. Zidouri, and W. Saif, *"A new modified particle swarm optimization algorithm for adaptive equalization,"* Digit. Signal Process., vol. 21, no. 2, pp. 195–207, Mar. 2011.

[14] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," IEEE Trans. Evol. Comput., vol. 6, no. 1, pp. 58–73, 2002.

[15] "NIST biometric score set," *National Institute of Standards and Technology*, 2006. [Online]. Available: http://www.itl.nist.gov/iad/894.03/biometricscores/.

[16] N. Poh and S. Bengio, *"Database, protocols and tools for evaluating score-level fusion algorithms in biometric authentication,"* Pattern Recognition, 2006. [Online]. Available: http://personal.ee.surrey.ac.uk/Personal/Norman.Poh/web/fusion.

[17] N. Poh, *"BANCA score database."* [Online]. Available: http://info.ee.surrey.ac.uk/Personal/Norman.Poh/web/banca_multi.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 900 staff, has 700 researchers, about 250 of whom are postgraduates, around 500 of whom have doctorates (Ph.D.), and around 200 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of S**love**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

From the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

Part of the Institute was reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project was developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park is a shareholding company hosting an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Higher Education, Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of the Economy, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 251 93 85
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Public relations: Polona Strnad

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit a manuscript to: http://www.informatica.si/Editors/PaperUpload.asp. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible from typing errors to global philosophical disagreements. The chosen editor will send the author the obtained reviews. If the paper is accepted, the editor will also send an email to the managing editor. The executive board will inform the author that the paper has been accepted, and the author will send the paper to the managing editor. The paper will be published within one year of receipt of email with the text in Informatica MS Word format or Informatica LaTeX format and figures in .eps format. Style and examples of papers can be obtained from http://www.informatica.si. Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the managing editor.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1000 Ljubljana, Slovenia. E-mail: drago.torkar@ijs.si

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than twentyone years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

http://www.informatica.si/

Referees from 2008 on:

A. Abraham, S. Abraham, R. Accornero, A. Adhikari, R. Ahmad, G. Alvarez, N. Anciaux, R. Arora, I. Awan, J. Azimi, C. Badica, Z. Balogh, S. Banerjee, G. Barbier, A. Baruzzo, B. Batagelj, T. Beaubouef, N. Beaulieu, M. ter Beek, P. Bellavista, K. Bilal, S. Bishop, J. Bodlaj, M. Bohanec, D. Bolme, Z. Bonikowski, B. Bošković, M. Botta, P. Brazdil, J. Brest, J. Brichau, A. Brodnik, D. Brown, I. Bruha, M. Bruynooghe, W. Buntine, D.D. Burdescu, J. Buys, X. Cai, Y. Cai, J.C. Cano, T. Cao, J.-V. Capella-Hernández, N. Carver, M. Cavazza, R. Ceylan, A. Chebotko, I. Chekalov, J. Chen, L.-M. Cheng, G. Chiola, Y.-C. Chiou, I. Chorbev, S.R. Choudhary, S.S.M. Chow, K.R. Chowdhury, V. Christlein, W. Chu, L. Chung, M. Ciglarič, J.-N. Colin, V. Cortellessa, J. Cui, P. Cui, Z. Cui, D. Cutting, A. Cuzzocrea, V. Cvjetkovic, J. Cypryjanski, L. Čehovin, D. Čerepnalkoski, I. Čosić, G. Daniele, G. Danoy, M. Dash, S. Datt, A. Datta, M.-Y. Day, F. Debili, C.J. Debono, J. Dedič, P. Degano, A. Dekdouk, H. Demirel, B. Demoen, S. Dendamrongvit, T. Deng, A. Derezinska, J. Dezert, G. Dias, I. Dimitrovski, S. Dobrišek, Q. Dou, J. Doumen, E. Dovgan, B. Dragovich, D. Drajic, O. Drbohlav, M. Drole, J. Dujmović, O. Ebers, J. Eder, S. Elaluf-Calderwood, E. Engström, U. riza Erturk, A. Farago, C. Fei, L. Feng, Y.X. Feng, B. Filipič, I. Fister, I. Fister Jr., D. Fišer, A. Flores, V.A. Fomichov, S. Forli, A. Freitas, J. Fridrich, S. Friedman, C. Fu, X. Fu, T. Fujimoto, G. Fung, S. Gabrielli, D. Galindo, A. Gambarara, M. Gams, M. Ganzha, J. Garbajosa, R. Gennari, G. Georgeson, N. Gligorić, S. Goel, G.H. Gonnet, D.S. Goodsell, S. Gordillo, J. Gore, M. Grčar, M. Grgurović, D. Grosse, Z.-H. Guan, D. Gubiani, M. Guid, C. Guo, B. Gupta, M. Gusev, M. Hahsler, Z. Haiping, A. Hameed, C. Hamzaçebi, Q.-L. Han, H. Hanping, T. Härder, J.N. Hatzopoulos, S. Hazelhurst, K. Hempstalk, J.M.G. Hidalgo, J. Hodgson, M. Holbl, M.P. Hong, G. Howells, M. Hu, J. Hyvärinen, D. Ienco, B. Ionescu, R. Irfan, N. Jaisankar, D. Jakobović, K. Jassem, I. Jawhar, Y. Jia, T. Jin, I. Jureta, Đ. Juričić, S. K, S. Kalajdziski, Y. Kalantidis, B. Kaluža, D. Kanellopoulos, R. Kapoor, D. Karapetyan, A. Kassler, D.S. Katz, A. Kaveh, S.U. Khan, M. Khattak, V. Khomenko, E.S. Khorasani, I. Kitanovski, D. Kocev, J. Kocijan, J. Kollár, A. Kontostathis, P. Korošec, A. Koschmider, D. Košir, J. Kovač, A. Krajnc, M. Krevs, J. Krogstie, P. Krsek, M. Kubat, M. Kukar, A. Kulis, A.P.S. Kumar, H. Kwaśnicka, W.K. Lai, C.-S. Laih, K.-Y. Lam, N. Landwehr, J. Lanir, A. Lavrov, M. Layouni, G. Leban, A. Lee, Y.-C. Lee, U. Legat, A. Leonardis, G. Li, G.-Z. Li, J. Li, X. Li, X. Li, Y. Li, Y. Li, S. Lian, L. Liao, C. Lim, J.-C. Lin, H. Liu, J. Liu, P. Liu, X. Liu, X. Liu, F. Logist, S. Loskovska, H. Lu, Z. Lu, X. Luo, M. Luštrek, I.V. Lyustig, S.A. Madani, M. Mahoney, S.U.R. Malik, Y. Marinakis, D. Marinčič, J. Marques-Silva, A. Martin, D. Marwede, M. Matijašević, T. Matsui, L. McMillan, A. McPherson, A. McPherson, Z. Meng, M.C. Mihaescu, V. Milea, N. Min-Allah, E. Minisci, V. Mišić, A.-H. Mogos, P. Mohapatra, D.D. Monica, A. Montanari, A. Moroni, J. Mosegaard, M. Moškon, L. de M. Mourelle, H. Moustafa, M. Možina, M. Mrak, Y. Mu, J. Mula, D. Nagamalai, M. Di Natale, A. Navarra, P. Navrat, N. Nedjah, R. Nejabati, W. Ng, Z. Ni, E.S. Nielsen, O. Nouali, F. Novak, B. Novikov, P. Nurmi, D. Obrul, B. Oliboni, X. Pan, M. Pančur, W. Pang, G. Papa, M. Paprzycki, M. Paralič, B.-K. Park, P. Patel, T.B. Pedersen, Z. Peng, R.G. Pensa, J. Perš, D. Petcu, B. Petelin, M. Petkovšek, D. Pevec, M. Pičulin, R. Piltaver, E. Pirogova, V. Podpečan, M. Polo, V. Pomponiu, E. Popescu, D. Poshyvanyk, B. Potočnik, R.J. Povinelli, S.R.M. Prasanna, K. Pripužić, G. Puppis, H. Qian, Y. Qian, L. Qiao, C. Qin, J. Que, J.-J. Quisquater, C. Rafe, S. Rahimi, V. Rajkovič, D. Raković, J. Ramaekers, J. Ramon, R. Ravnik, Y. Reddy, W. Reimche, H. Rezankova, D. Rispoli, B. Ristevski, B. Robič, J.A. Rodriguez-Aguilar, P. Rohatgi, W. Rossak, I. Rožanc, J. Rupnik, S.B. Sadkhan, K. Saeed, M. Saeki, K.S.M. Sahari, C. Sakharwade, E. Sakkopoulos, P. Sala, M.H. Samadzadeh, J.S. Sandhu, P. Scaglioso, V. Schau, W. Schempp, J. Seberry, A. Senanayake, M. Senobari, T.C. Seong, S. Shamala, c. shi, Z. Shi, L. Shiguo, N. Shilov, Z.-E.H. Slimane, F. Smith, H. Sneed, P. Sokolowski, T. Song, A. Soppera, A. Sorniotti, M. Stajdohar, L. Stanescu, D. Strnad, X. Sun, L. Šajn, R. Šenkeřík, M.R. Šikonja, J. Šilc, I. Škrjanc, T. Štajner, B. Šter, V. Štruc, H. Takizawa, C. Talcott, N. Tomasev, D. Torkar, S. Torrente, M. Trampuš, C. Tranoris, K. Trojacanec, M. Tschierschke, F. De Turck, J. Twycross, N. Tziritas, W. Vanhoof, P. Vateekul, L.A. Vese, A. Visconti, B. Vlaovič, V. Vojisavljević, M. Vozalis, P. Vračar, V. Vranić, C.-H. Wang, H. Wang, H. Wang, H. Wang, S. Wang, X.-F. Wang, X. Wang, Y. Wang, A. Wasilewska, S. Wenzel, V. Wickramasinghe, J. Wong, S. Wrobel, K. Wrona, B. Wu, L. Xiang, Y. Xiang, D. Xiao, F. Xie, L. Xie, Z. Xing, H. Yang, X. Yang, N.Y. Yen, C. Yong-Sheng, J.J. You, G. Yu, X. Zabulis, A. Zainal, A. Zamuda, M. Zand, Z. Zhang, Z. Zhao, D. Zheng, J. Zheng, X. Zheng, Z.-H. Zhou, F. Zhuang, A. Zimmermann, M.J. Zuo, B. Zupan, M. Zuqiang, B. Žalik, J. Žižka,

Informatica

An International Journal of Computing and Informatics

Web edition of Informatica may be accessed at: http://www.informatica.si.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Litostrojska cesta 54, 1000 Ljubljana, Slovenia.
The subscription rate for 2015 (Volume 39) is
– 60 EUR for institutions,
– 30 EUR for individuals, and
– 15 EUR for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut Žnidar.
Printing: ABO grafika d.o.o., Ob železnici 16, 1000 Ljubljana.

Orders may be placed by email (drago.torkar@ijs.si), telephone (+386 1 477 3900) or fax (+386 1 251 93 85). The payment should be made to our bank account no.: 02083-0013014662 at NLB d.d., 1520 Ljubljana, Trg republike 2, Slovenija, IBAN no.: SI56020830013014662, SWIFT Code: LJBASI2X.

Informatica is published by Slovene Society Informatika (president Niko Schlamberger) in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Janez Perš)
Slovenian Artificial Intelligence Society (Dunja Mladenić)
Cognitive Science Society (Urban Kordeš)
Slovenian Society of Mathematicians, Physicists and Astronomers (Andrej Likar)
Automatic Control Society of Slovenia (Sašo Blažič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Vojteh Leskovšek)
ACM Slovenia (Andrej Brodnik)

Informatica is financially supported by the Slovenian research agency from the Call for co-financing of scientific periodical publications.

Informatica is surveyed by: ACM Digital Library, Citeseer, COBISS, Compendex, Computer & Information Systems Abstracts, Computer Database, Computer Science Index, Current Mathematical Publications, DBLP Computer Science Bibliography, Directory of Open Access Journals, InfoTrac OneFile, Inspec, Linguistic and Language Behaviour Abstracts, Mathematical Reviews, MatSciNet, MatSci on SilverPlatter, Scopus, Zentralblatt Math

Volume 39 Number 2 June 2015 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **39** (2015) Number 2, pp. 103–217