jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2015, VOL. 61, NO. 1, PP. 7–16
Manuscript received January 3, 2015; revised March 2015. DOI: 10.1515/eletel-2015-0001

On Some Aspects of Genetic and Evolutionary Methods for Optimization Purposes

Marcin Woźniak and Dawid Połap

***Abstract*—In this paper, the idea of applying some hybrid genetic algorithms with gradient local search and evolutionary optimization techniques is formulated. For two different test functions the proposed versions of the algorithms have been examined. Research results are presented and discussed to show potential efficiency in optimization purposes.**

***Keywords*—computational intelligence, genetic algorithm, heuristic algorithm, optimization**

I. Introduction

ALGORITHMS inspired by nature belong to Computational Intelligence (CI), which have been applied for searching the most accurate solution for optimization problems in the shortest time. These algorithms can be grouped for instance as Evolutionary Algorithms (EA), Heuristic Algorithms (HA), Swarm Algorithms (SA) or hybrids. To the group of Evolutionary Algorithms among others we include Genetic Algorithms (GA), which have origins in 1970s [1]. This has given an impulse to develop many derivative methods simulating other real life into computer procedures to create efficient optimization techniques applicable in various tasks.

Subsequently, CI methods were applied in various engineering problems: active modules optimization [2], dynamic systems work analysis [3] and [4], hydro-pneumatic modules optimization [5] or even entire vehicle simulation and positioning [6]. Another approach is to optimize LAN or queueing systems: [7], [8], [9], [10], [11] and [12]. It is possible to apply CI to participate in the process of learning for Artificial Intelligence (AI) and Decision Support Systems (DSS) [13]. Most recent research results show that CI is also applicable in image processing [14] and [15]. Several CI applications, have led to modifying, improving the original algorithm, and to create efficient improvements tailored for specific purposes. Swarm intelligence is applicable in cast simulation [16], [17], [18]. Some other interesting computing techniques for logic techniques applied in optimization can be found in [19], [20] and [21].

There are many possible applications. In each of these cases CI was used to find optimum values for some phenomena represented in tailored mathematical models. Some of the applied models are representing positioned object states in multi dimensional spaces. Therefore it is important to investigate how GA methods optimize given models. In this paper we try to present and discuss some aspects of GA potential application, it's classic version or modified GA in search of optimum of the multi variable functions. To perform experiments some classic functions were taken and tests were widely applied to verify optimization techniques.

M. Woźniak and D. Połap are with Institute of Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice, Poland (e-mail: Dawid.Polap@gmail.com; Marcin.Wozniak@polsl.pl).

II. Test Functions for Computational Intelligence Optimization Techniques

In verification of examined methods we try to use test functions that represent some classic optimization problems, what is helpful to compare results. We define models at object states represented in points $\mathbf{x}$, that have $i = 1, \ldots, N$ dimensions depending on optimization task.

De Jong functions group some type of benchmark functions designed for efficiency tests. There are known many versions of these functions, among them three basic functions can be given. First De Jong function can be written for multi dimensional model spaces as

$$f_{deJong1}(\mathbf{x}) = \sum_{i=1}^{N} x_i^2, \tag{1}$$

where for each N dimensions object state defined in point $\mathbf{x}$ we have x_i spatial coordinates to represent various aspects to optimize. This function has minimum $f_{deJong1}(\mathbf{x}) = 0$ at point $\mathbf{x} = 0$. In two dimension form $f_{deJong1}(x_1, x_2) = x_1^2 + x_2^2$, it is presented in Fig. 1 Second De Jong function can be written for multi dimensional model spaces as

$$f_{deJong2}(\mathbf{x}) = \sum_{i=1}^{N} \lfloor x_i \rfloor, \tag{2}$$

where notations are similar to (1). In two dimension form $f_{deJong2}(x_1, x_2) = \lfloor x_1 \rfloor + \lfloor x_2 \rfloor$ when pictured on $\langle -5.12, 5.12 \rangle \times \langle -5.12, 5.12 \rangle$ it has minimum at point $\mathbf{x} = -5.12$. This is shown in Fig. 2. Finally third De Jong function in multi dimension form has equation

$$f_{deJong3}(\mathbf{x}) = \sum_{i=1}^{N} i \cdot x_i^4 + N(0,1), \tag{3}$$

where notations are similar to (1) and $N(0,1)$ is normal distribution. This function has minimum $f_{deJong1}(\mathbf{x}) = 0$ at point $\mathbf{x} = 0$. In two dimension form $f_{deJong3}(x_1, x_2) = x_1^4 + 2x_2^4 + N(0,1)$, it is presented in Fig. 3 There are also some hybrid functions that have origins in the above presented De Jong functions. One of them is Rastragin function, which for multi dimension form can be defined as

$$f_{Rastr}(\mathbf{x}) = 10 \cdot N + \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi \cdot x_i)), \tag{4}$$

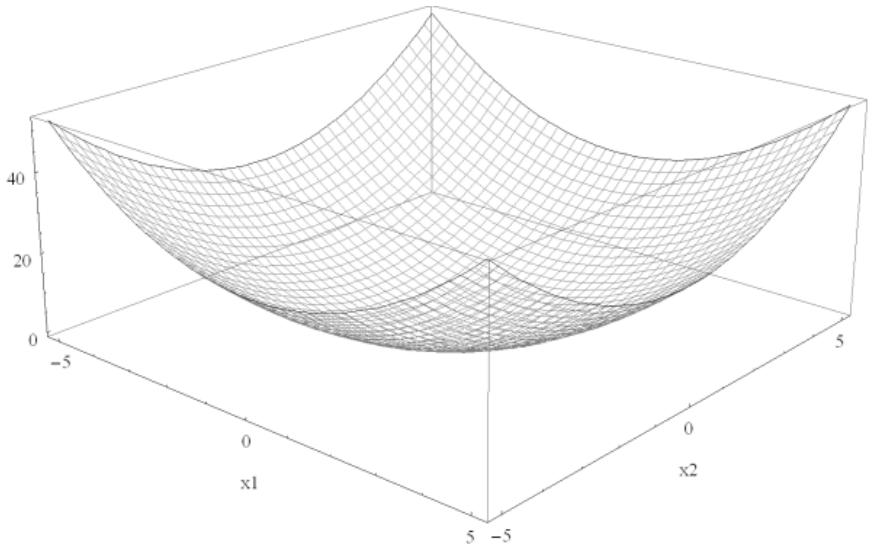

Fig. 1. First De Jong function.

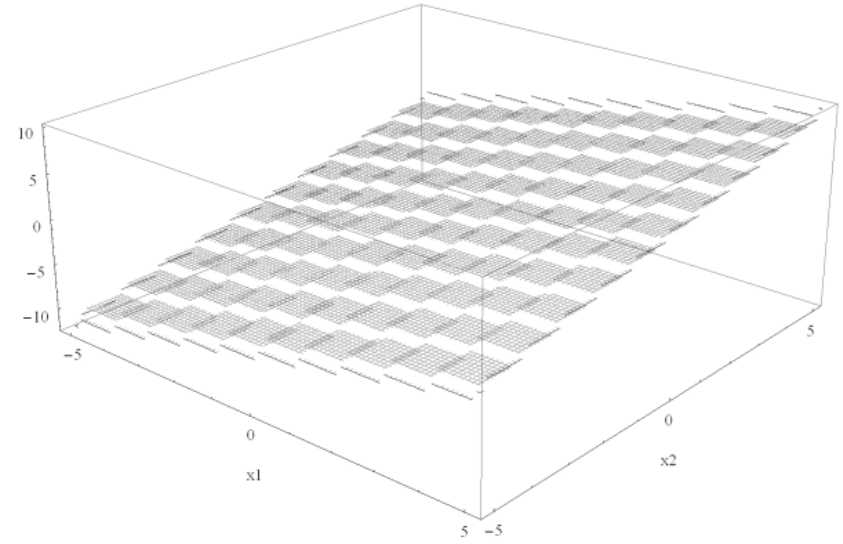

Fig. 2. Second De Jong function.

where notations are similar to (1). This function has minimum $f_{Rastr}(\mathbf{x}) = 0$ at point $\mathbf{x} = 0$. In two dimension form $f_{Rastr}(x_1, x_2) = 10{\cdot}2+(x_1^2-\cos(2\pi{\cdot}x_1))+(x_2^2-\cos(2\pi{\cdot}x_2))$, it is presented in Fig. 4

III. Optimization Techniques

Optimization or solving methods have origins in analytical attempt, which is based on gradient optimization. Analytical attempt is giving precise method that by the use of derivatives is able to calculate strict optimal solution for each function. However it is not possible to have the analytical form of each mathematic function ad-hoc, moreover there are some computational restrictions for large scale functions. These make optimization process complicated and prolonged in time. Therefore CI methods are so important for optimization. Let us discuss classic attempt and proposed CI methods.

A. Gradient Optimization – Derivative Attempt

Gradient descent is one of the local optimization methods. We can define it in algorithmic attempt, that enables us to omit analytical calculations of the derivative form and just use computational power to find the optimum. Let us discuss a numerical algorithm which uses the direction of the search space of applied mathematical model solutions by the negative gradient of the fitness function. In Fig. 5 is presented a sample search operation using gradient attempt for given function over the provided solution space. The algorithm starts with a random selection of a point $\mathbf{x_0}$, which is a first randomly taken optimum position. In the following steps, the negative gradient $-\nabla f_{\mathbf{x_i}}$ for point $\mathbf{x_i}$ is calculated $\frac{\partial f(x_1,\ldots,x_n)}{\partial x_i}$. Gradient $\nabla f_{\mathbf{x_i}}$ is direction of the fastest function growth measured at point $\mathbf{x_i}$ and negative gradient $\nabla f_{\mathbf{x_i}}$ is direction of the fastest function descent measured at point $\mathbf{x_i}$. Then, a new point based on the

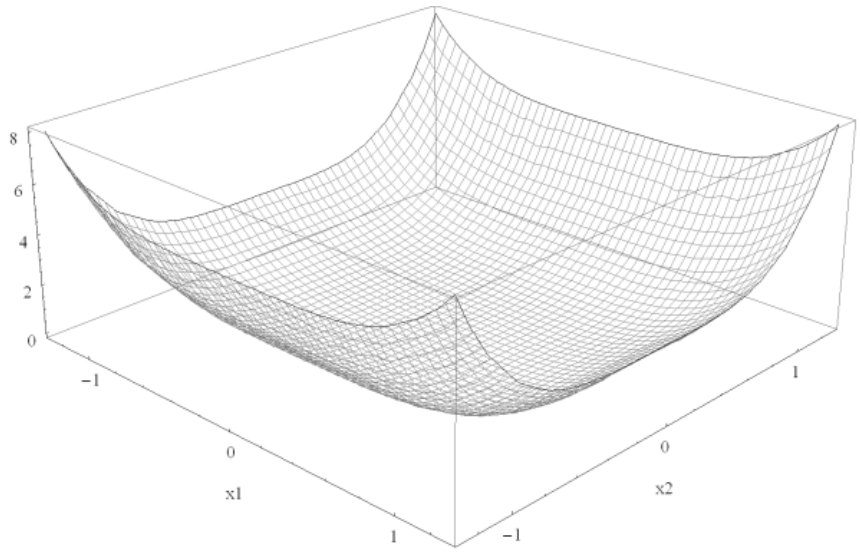

Fig. 3. Third De Jong function.

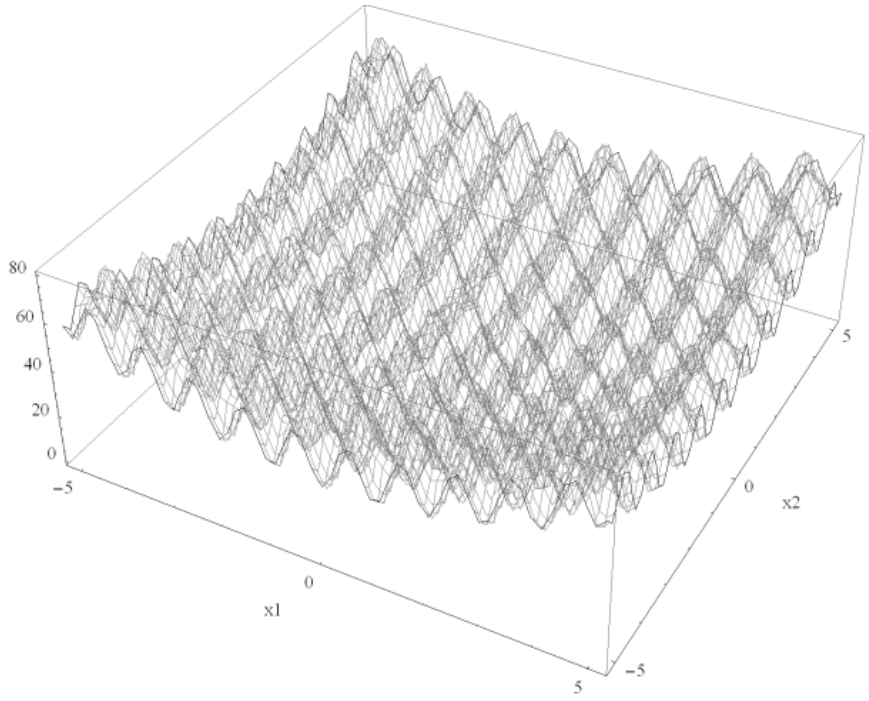

Fig. 4. Rastragin function.

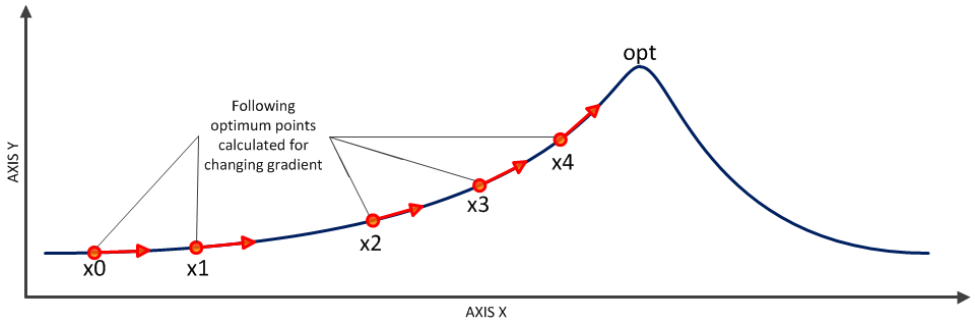

Fig. 5. Sample representation of the gradient change for the following optimal points in numerical calculations.

direction of the d_{x_i} is found using formula

$$\mathbf{x_i^{t+1}} = \mathbf{x_i^t} - \lambda \cdot \nabla f_{\mathbf{x_i}}, \tag{5}$$

where for space points $\mathbf{x_i}$ parameter λ represents the value of the step size and t is time horizon represented by the following iterations. Then, a value of the fitness function for the old and a new points is calculated. For the case of minimization, i.e. if the value of the fitness function for a new point $\mathbf{x_i^{t+1}}$ is smaller than for the old point $\mathbf{x_i^t}$, the new point $\mathbf{x_i^{t+1}}$ replace the old one. The algorithm is performed to comply with stop condition, which for instance are a number of iterations (time horizon t is defined in some way) or the method shall stop after reaching a particular value for the fitness condition. The method is presented in Algorithm 1.

Algorithm 1 Minimization Gradient Descent Algorithm

```
Start,
Define fitness condition φ(·),
Define step size λ and time horizon T,
Randomize start point x_0 for the search space,
t = 1,
while t ≤ T do
    Calculate x_i^{t+1} using (5),
    if φ(x_i^{t+1}) ≤ φ(x_i^t) then
        x_i^t = x_i^{t+1},
    else
        x_i^t = x_i^t,
    end if
    t++
end while
Return x_i^t,
Stop.
```

B. Evolutionary Algorithms

The first implemented Evolutionary Algorithm was Genetic Algorithm created by John Holland in 70s of 20th century [1]. The main idea, which led to create this algorithm, was to make a computer program solving problems in a similar way to the natural evolution process. The first GA purpose was to structure evolution of populations consisted of some amount of individuals. In this population each individual had binary genetic code set to it. These codes are equivalent to chromosomes in organisms, what make their elements equivalent to the genes. Later were introduced some improvements, which were inspired by operations that occur during the biological process of evolution. Those operations consecutively were: mechanism of transforming binary codes in similar way as crossing-over in process of inheritance, mutation and selection. Holland also noticed that usage of logic operators and selection can increase average value of population fitness. Using encoding and genetic operators let to solve problems powered by new technology, like advanced programming for efficient computers what is most popular tendency nowadays.

1) Genetic Algorithm (GA): First step of GA is initialization that leads to create the beginning population. This process consists of setting the size of population, chromosomes encoding and fitness function $\phi(\cdot)$ formulation with set of limitations. Chromosomes are usually presented in two ways. The first of them is a binary code and the second one is a real number. Elements of beginning population are found in a random way. Important parameter is the size of population. Too small number of individuals can quickly finish the algorithm without getting right solution, in second case using too many numbers can make calculations long and be aggravating for the computer.

Next step is reproduction, this is when some individuals are chosen to next operations. There are some ways of reproducing selected ones to be passed to new population. One of them is reproduction based on probability that individual $\mathbf{x_i}$ belongs to selected group. It is taken to next generation with a probability, that is increasing with the higher level of fitness function at this point $\phi(\mathbf{x_i})$. For selected group in an existing population P^t we determine a random variable p_r given by

$$\begin{cases} p_r(\mathbf{x_i^t}) = \frac{\phi(\mathbf{x_i^t})}{\mathbf{x_i^t}} \\ \mathbf{x_i^t} \in P^t \end{cases}. \tag{6}$$

This random variable is sampled few times and distribution function of reproduction probability is introduced

$$P_r(\mathbf{x_i^t}) = \sum_{j=1}^{i} p_r(\mathbf{x_i^t}). \tag{7}$$

Then random number α between $(0, 1)$ is taken. If individual $\mathbf{x_i}$ follows the assumption

$$P_r(\mathbf{x_{i-1}^t}) < \alpha \leq P_r(\mathbf{x_i^t}) \tag{8}$$

then it is reproduced. Reproducing is controlled by genetic operators which should create diverse population. Classical operators are mutation and crossover.

Mutation is a process of changing the chromosomes, like adding a random real ξ from $(0, 1)$ to the chromosome

$$\mathbf{x_i^{t+1}}_{mutated} = \mathbf{x_i^t} + \xi. \tag{9}$$

Moreover, values of vector λ, which length is equal to the amount of individuals in population is randomized. Each chromosome is mutated if it follows the inequality

$$\lambda_i < p_m, \tag{10}$$

where p_m is mutation probability.

Crossover is an operation in which genetic material between two individuals called parents is exchanged. Crossover of two parents creates two individuals which are added to population. The parameter ξ is found as a random real number from $(0, 1)$. Then parental chromosomes are averaged using following formula leading to create new individuals

$$\mathbf{x_i}_{des1}^{t+1} = \mathbf{x_i}^t + \xi(\mathbf{x_{i+1}^t} - \mathbf{x_i^t})\mathbf{x_i}_{des2}, \tag{11}$$

where $\mathbf{x_i}$ and $\mathbf{x_{i+1}}$ are parent chromosomes and ξ is similar to (9). In the next step all individuals must be evaluated how they fit to the environment. New population replace the older one. This process is called succession.

The one of the way of succession is to replace old population with the new one. Another way is to leave some chromosomes of old population and fill in with new individuals.

Amount of remaining chromosomes from the old population is described by parameter $g \in (0, 100)$. There are several ways of deciding which individuals should be left:

- leaving individuals which are best adapted,
- leaving individuals in random way,
- delete individuals similar to those of next generation,
- succession.

most effective is elite succession. It assumes that the best individuals are left no matter from which generation they came from. This is performed using some "back-up" population, where we store all the best individuals to stay in this and later populations. Individuals from new and old generation are sorted according to value of fitness function and some number of the best is taken to new population. This method of succession let the best ever individuals to survive.

2) Hybrid Genetic Algorithm (HGA): The solution obtained using GA is often good, however usually requires some small, local corrections to become a local optimum. This is one of the most important reasons that the hybrid algorithm was developed: to correct obtained solution. In according to this, hybrids are modifications of the classic version of heuristic algorithms with a better capability of the local search.

In the later 90s for the first time was used the term Memetic Algorithm (MA). The word meme is the analogous term to gene in natural evolution. It refers to the smallest unit of cultural evolution, i.e. pattern of behavior. Memes form create a large system, where are formed cultural patterns. Culture changes in the interaction when certain memes gain an advantage, while others are forgotten. In practice, the environment forces acceptance of the certain memes or oblivion. The main idea of MA is hybridization, namely a combination of heuristics such as GA with local search. In this case, local search is understood as learning individuals. In GA, local search may be implemented between selected genetic operators.

Algorithm 2 Schematic Memetic Algorithm

1: Start,
2: Define fitness condition $\phi(\cdot)$,
3: Randomize initial population of individuals with random solutions,
4: Evaluate individuals,
5: Define ξ, p_m, α and time horizon T,
6: $t = 1$,
7: **while** $t \leq T$ **do**
8: Sort population according to $\phi(\cdot)$ values,
9: Selection,
10: Mutation in population using (9),
11: Crossing in population using (11),
12: Local search using Algorithm 1,
13: Evaluate fitness of the new individuals,
14: Replace the worst individuals of the population with new ones,
15: **end while**
16: Return the solution,
17: Stop.

3) Baldwin Effect: J.M. Baldwin suggested the theory that the skills acquired during the life of the individual are transferred indirectly to offspring [22]. For example, in a given environment the ability to increase chances of survival is better with each year. In the case of individuals with this ability, they will have a larger number of offspring inheriting the genes with peculiar features. The development of language ability in humans can be explained by general conclusion from Baldwins theory, that skills are not inherited but the ability to acquire them. The presented mechanism is called organic selection or Baldwin Effect. This is a learning mechanism, which reflects only in value of fitness function of the individual.

In practice, the Baldwin Effect is simple to implement. Implementation is a simulation by performing local optimization at the selection to determine the adjustment feature. The genetic code remains unchanged and consequently, refers to flattening fitness landscape within the basin of attraction. In the experiments we have implemented local search mechanism using gradient optimization presented in Algorithm 1.

C. Other CI methods

However Evolutionary Computation is still in progress and nowadays we can give a number of newly developed heuristic methods let us give only two selected examples.

1) Cuckoo Search Algorithm (CSA): Cuckoo Search Algorithm is a very efficient gradient free optimization technique, where some Gauss distribution versions are applied in optimization. CSA simulates behavior of cuckoos. These birds, accept for famous sounds, have special nature of breeding. A cuckoo is flying and looking for nest to lay an egg. They are choosing nest, where are already eggs. Moreover these eggs must look similar to their. Cuckoos lay an egg and fly off. When hosts come home they either get rid of intruder egg or just simply accept new situation. This process is modeled and applied as CI algorithm, where we assume:

- Cuckoo is simply $\mathbf{x_i}$ in CSA.
- Each cuckoo has only one egg to lay.
- Best nests containing egg are transferred to next generation.
- Rest of cuckoo population is taken at random.
- Hosts may find that intruder egg is hosted with probability $1-p_\alpha \in \langle 0, 1\rangle$ and get rid of it. In this case a new cuckoo is placed randomly.

We check fitness function for each of cuckoos. The best points are transferred to next round. The rest of population is taken at random to maintain constant level of cuckoos. Since these all operations are completed we start next round in CSA. New generation of cuckoos is placed and we start procedure from the beginning.

In every generation we only model the choice of place to lay an egg with particular equations. This movement has some statistic background. It uses a concept of random walks, what helps to perform non local search in different type solution spaces. Virtual cuckoo movement is modeled with formula

$$\mathbf{x_i^{t+1}} = \mathbf{x_i^t} + \mu \cdot L(\beta, \gamma, \delta), \tag{12}$$

where the symbols are: $\mathbf{x_i^{t+1}}$ – next solution in CSA, μ – length of step in random walk based on normal distribution

$N\left(\frac{\gamma}{cuckoos};0,1\right)$, L($\beta$,$\gamma$,$\delta$) – Lévy flight for a given step length β, δ – length of minimum step for random walk and γ – scaling parameter for Lévy flight.

Lévy flights are made isotropic in random directions [23], according to formula

$$L(\beta,\gamma,\delta)=\begin{cases}\sqrt{\frac{\gamma}{2\pi}}\frac{exp[-\frac{\gamma}{2(\beta-\delta)}]}{(\beta-\delta)^{\frac{3}{2}}}, & 0<\beta<\delta<\infty\\ 0, & \text{other}\end{cases}. \tag{13}$$

Finally we only have to decide if it is "found" by hosts. This decision is modeled with equation

$$H(\mathbf{x_i^{t+1}})=\begin{cases}1-p_\alpha & \text{drop the egg}\\ p_\alpha & \text{the egg stays}\end{cases}, \tag{14}$$

where the symbols are: $H(\mathbf{x_i^{t+1}})$ – decision taken by hosts about intruder, $p_\alpha \in \langle 0,1\rangle$ – chance for cuckoo egg to stay.

Algorithm 3 Cuckoo Search Algorithm

1: Define all coefficients: $p_\alpha \in \langle 0,1\rangle$, β, γ, δ, $bestratio$, number of $cuckoos$ and number of $generations$,
2: Define fitness condition $\phi(\cdot)$,
3: Create at random initial population,
4: t:=0,
5: **while** $t \leq generations$ **do**
6: Move $cuckoos$ according to (12) and (13),
7: Hosts decide if the eggs stay or no – decision according to (14),
8: Sort points (cuckoos) according to the value of criterion function,
9: Evaluate population and take $bestratio$ of them to next $generation$,
10: Rest of $cuckoos$ take at random,
11: Next $generation$ $t++$,
12: **end while**
13: Best $cuckoos$ from the last $generation$ are solution.

2) Firefly Algorithm (FA): FA simulates behavior of flying and blinking insects while searching for a partner. Individuals are described by several biological traits: specific way of flashing, specific way of moving and specific perception of the others. These are mathematically modeled as:

- γ–light absorption coefficient in given circumstances,
- μ–firefly random motion factor,
- β_{pop}–firefly attractiveness factor,
- I_{pop}–light intensity factor for given species,

which implement behavior of different species of fireflies and natural conditions of the environment. Just as in nature, a firefly goes to the most attractive other one by measuring the intensity of flickers over the distance between them characterized by a suitable metric. In FA an average distance r_{ij} between any two fireflies i and j maps the inverse square law. Attractiveness of individuals decreases with increasing distance r_{ij} between them. We also map air absorption of light, which makes fireflies visible to certain distance. In description of FA we assume:

- All fireflies are unisex, therefore one individual can be attracted to any other firefly regardless of gender,
- Attractiveness is proportional to brightness. Thus, for every two fireflies less clear flashing one will move toward brighter one,
- Attractiveness decreases with increasing distance between individuals,
- If there is no clearer and more visible firefly within the range, then each one will move randomly.

Distance between any two fireflies i and j situated at points $\mathbf{x_i}$ and $\mathbf{x_j}$ is defined using metric

$$r_{ij}^t=\|\mathbf{x_i^t}-\mathbf{x_j^t}\|=\sqrt{\sum_{k=1}^{N}(x_{i,k}^t-x_{j,k}^t)^2}, \tag{15}$$

where notations in t iteration are: $\mathbf{x_i^t}$, $\mathbf{x_j^t}$–points in R^N space, $x_{i,k}^t$, $x_{k,j}^t$–k-th components of the spatial coordinates.

Light intensity I_{ij}^t from firefly i that is received by firefly j decreases with increasing distance r_{ij}^t between them. Natural light is absorbed by media, so attractiveness also vary according to absorption and distance between them. In the model, light intensity varies according to

$$I_{ij}^t(r_{ij}^t)=I_{pop}\cdot e^{-\gamma\cdot(r_{ij}^t)^2}, \tag{16}$$

where the symbols are: $I_{ij}^t(r_{ij}^t)$–intensity of light from firefly i that is received by firefly j in t iteration, r_{ij}^t–distance between firefly i and firefly j defined in (15), γ–light absorption coefficient mapping natural conditions.

Attractiveness of firefly i to firefly j decreases with increasing distance. Attractiveness is proportional to intensity of light seen by surrounding individuals and defined as

$$\beta_{ij}(r_{ij}^t)=\beta_{pop}\cdot e^{-\gamma\cdot(r_{ij}^t)^2}, \tag{17}$$

where notations in t iteration are: $\beta_{ij}(r_{ij}^t)$–attractiveness of firefly i to firefly j, r_{ij}^t–distance between firefly i and firefly j, γ–light absorption factor mapping natural conditions, β_{pop}–firefly attractiveness factor.

Movement of individual is based on conditioned distance to other individuals surrounding it. Firefly will go to most attractive one, measuring intensity of flicker over the distance between them. In given model, natural identification of individuals and their attractiveness defined in (17) depends on light intensity defined in (16) and distance separating them defined in (15). In nature fireflies that are closer not only see themselves better, but also are more attractive to each other. Using these features in the model, calculations simulate natural behavior of fireflies. Firefly i motions toward more attractive and brighter (clearer flashing) individual j using information about other individuals denotes formula

$$\mathbf{x_i^{t+1}}=\mathbf{x_i^t}+(\mathbf{x_j^t}-\mathbf{x_i^t})\cdot\beta_{ij}^t(r_{ij}^t)\cdot I_{ij}^t(r_{ij}^t)+\mu\cdot e_i, \tag{18}$$

where the symbols are: $\mathbf{x_i^t}$, $\mathbf{x_j^t}$–points in the picture, $\beta_{ij}^t(r_{ij}^t)$–attractiveness of firefly i to firefly j defined in (17), $I_{ij}^t(r_{ij}^t)$–intensity of light from firefly i that is received by firefly j defined in (16), r_{ij}^t–distance between fireflies i and j defined in (15), γ–light absorption coefficient mapping natural conditions, μ–coefficient mapping natural random motion of individuals in population, e_i–vector randomly changing position of firefly. FA implementation is presented in Algorithm 4.

Algorithm 4 Firefly Algorithm

1: Define all coefficients and number of $fireflies$ and $generation$ in the algorithm,
2: Define fitness condition $\phi(\cdot)$,
3: Create at random initial population of $fireflies$,
4: t:=0,
5: **while** $t \leq generation$ **do**
6: Calculate distance between individuals in population P using (15),
7: Calculate light intensity for individuals in population P using (16),
8: Calculate attractiveness for individuals in population P using (17),
9: Evaluate individuals in population,
10: Move individuals towards closest and most attractive individual using (18),
11: Evaluate individuals in population,
12: Replace $worst\_ratio$ individuals from population with at random,
13: Rest of $fireflies$ take at random,
14: Next $generation$ $t := t + 1$,
15: **end while**
16: Best $fireflies$ from the last $generation$ are solution.

IV. Experimental Results

Numerical tests were performed to search for the optimum of five variables functions using Genetic Algorithm (GA), Hybrid Genetic Algorithm (HGA) with and without the Baldwin Effect, Cuckoo Search Algorithm (CSA) and Firefly Algorithm (FA). Results were examined in velocity of convergence and the precision in comparison to the accuracy results. Two different fitness functions were examined. The first function was Rastragin

$$\phi_1(\mathbf{x_i}) = 50 + \sum_{i=1}^{5} \left({\mathbf{x_i}}^2 - 10\cos(2\pi \mathbf{x_i}) \right), \qquad (19)$$

where $d = 5$ is number of dimensions. The second function was first De Jong

$$\phi_2(\mathbf{x_i}) = -\sum_{i=1}^{5} {\mathbf{x_i}}^2. \qquad (20)$$

In our tests, for both evaluated functions research were done within the hypercube $[-5.12, 5.12]^d$. The analytical solution i.e. for $\phi_2(\mathbf{x}) = (0, 0, 0, 0, 0)$. Research results are shown in Tab. I and Tab. II. Sample results are presented in Fig. 6 and Fig. 7 but of course not in 5 dimension, as it is rather hard to do it, but in lower space. Analyzing values we can see that HGA with Baldwin Effect, CSA and FA can be most precise methods for functions optimization where we do not have many modules with local optimum. Among them CSA is most simple to implement and because of dedicated Gaussian distribution it is self improvable. The experiments, however, show also that for functions with many modules with local optimum examined methods may fail to give precise solution. This means that we still need to work on local search that will not suppress global search, however it is non trivial problem to solve even using sophisticated methods.

V. Final Remarks

The paper is to present research on optimization of common test functions. In the following sections selected methods were discussed. The numerical calculations presented in section IV proved advance of evolutionary computation based methods in common test functions optimization. These methods are easy to implement with possibility to improve them for various purposes presented in section I. Future research will lead to define improvements of discussed algorithms leading to tailored solutions for optimization purposes.

References

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

[2] A. Nowak and M. Woźniak, "Algorithm for optimization of the active module by the use of genetic algorithm," *Acta Mechanica Slovaca*, vol. 3, no. C, pp. 307–316, 2008.

[3] A. Nowak and M. Woźniak, "Multiresolution derives analysis of module mechatronical systems," *Mechanika*, vol. 6, no. 74, pp. 45–51, 2008.

[4] A. Nowak and M. Woźniak, "Analysis of the active module mechatronical systems," in *Proceedings of Mechanika 2008 – ICM'2008*. Kaunas, Lietuva: Kaunas University of Technology Press, 2008, pp. 371–376.

[5] A. Nowak and M. Woźniak, "Optimization of the active vibroisolation system for operator's cabin with the hydropneumatical element," *Transactions of the Universities of Košice*, vol. 3, no. C, pp. 113–116, 2009.

[6] M. Woźniak, "Fitness function for evolutionary computation applied in dynamic object simulation and positioning," in *Proceedings of the IEEE Symposium Series on Computational Intelligence – SSCI'2014 : 2014 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems – CIVTS'2014*. 9-12 December, Orlando, Florida, USA: IEEE, 2014, pp. 108–114.

[7] M. Gabryel, R. K. Nowicki, M. Woźniak, and W. M. Kempa, "Genetic cost optimization of the $GI/M/1/N$ finite-buffer queue with a single vacation policy," *Lecture Notes in Artificial Intelligence – ICAISC'2013*, vol. 7895, PART II, pp. 12–23, 2013.

[8] M. Woźniak, "On applying cuckoo search algorithm to positioning GI/M/1/N finite-buffer queue with a single vacation policy," in *Proceedings of the 12th Mexican International Conference on Artificial Intelligence - MICAI'2013*. 24-30 November, Mexico City, Mexico: IEEE, 2013, pp. 59–64.

[9] M. Woźniak, W. M. Kempa, M. Gabryel, R. K. Nowicki, and Z. Shao, "On applying evolutionary computation methods to optimization of vacation cycle costs in finite-buffer queue," *Lecture Notes in Artificial Intelligence – ICAISC'2014*, vol. 8467, PART I, pp. 480–491, 2014.

[10] M. Woźniak, "On positioning traffic in nosql database systems by the use of particle swarm algorithm," in *Proceedings of XV Workshop DAGLI OGGETTI AGLI AGENTI – WOA'2014*. CEUR Workshop Proceedings (CEUR-WS.org), RWTH Aachen University, 2014, paper 5.

[11] M. Woźniak, M. Gabryel, R. K. Nowicki, and B. Nowak, "A novel approach to position traffic in nosql database systems by the use of firefly algorithm," in *Proceedings of the 9th International Conference on Knowledge, Information and Creativity Support Systems*, G. A. Papadopoulos, Ed. 6-8 November, Limassol, Cyprus: University of Cyprus Press, 2014, pp. 208–218.

[12] M. Woźniak, W. M. Kempa, M. Gabryel, and R. K. Nowicki, "A finite-buffer queue with single vacation policy – analytical study with evolutionary positioning," *International Journal of Applied Mathematics and Computer Science*, vol. 24, no. 4, pp. 887–900, 2014.

[13] M. Gabryel, M. Woźniak, and R. K. Nowicki, "Creating learning sets for control systems using an evolutionary method," *Lecture Notes in Computer Science – ICAISC'2012*, vol. 7269, pp. 206–213, 2012.

[14] M. Woźniak and Z. Marszałek, "An idea to apply firefly algorithm in 2D images key-points search," *Communications in Computer and Information Science – ICIST'2014*, vol. 465, pp. 312–323, 2014.

[15] M. Woźniak and D. Połap, "Basic concept of cuckoo search algorithm for 2D images processing with some research results," in *Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications – SIGMAP'2014*. 28-30 August, Vienna, Austria: SciTePress – INSTICC, 2014, pp. 164–173.

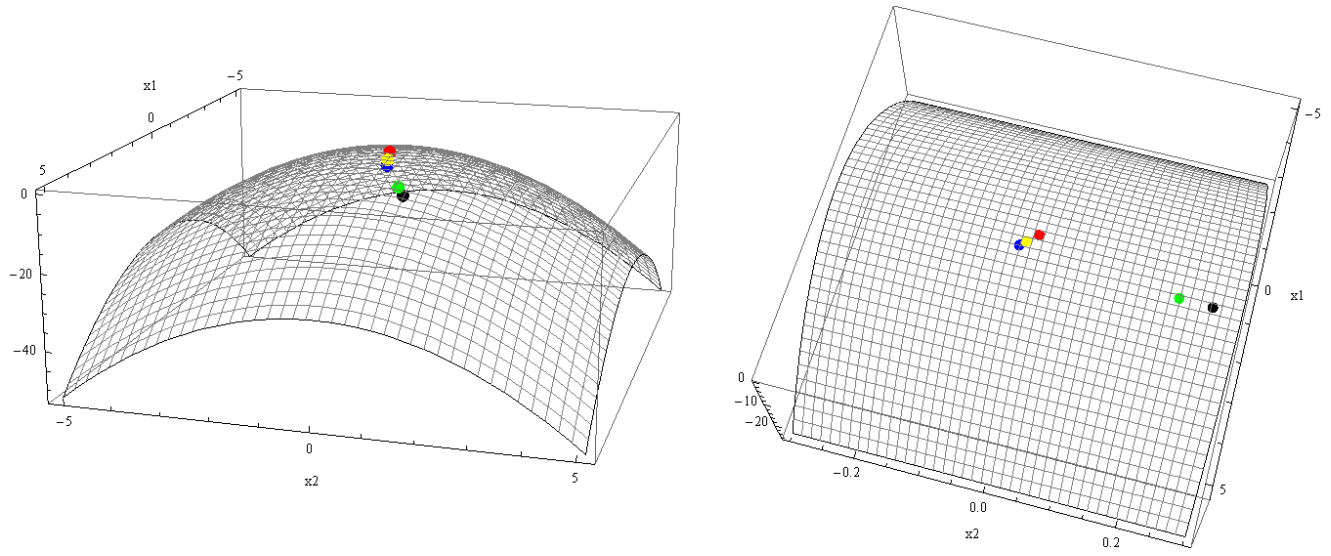

Fig. 6. First De Jong function optimization results: on the left original image and on the right a close-up, both for GA (black), GA+Baldwin Effect (green), HGA(pink), HGA+Baldwin Effect (red), CSA (yellow), FA (blue) calculated for 300 individuals in 100 iterations

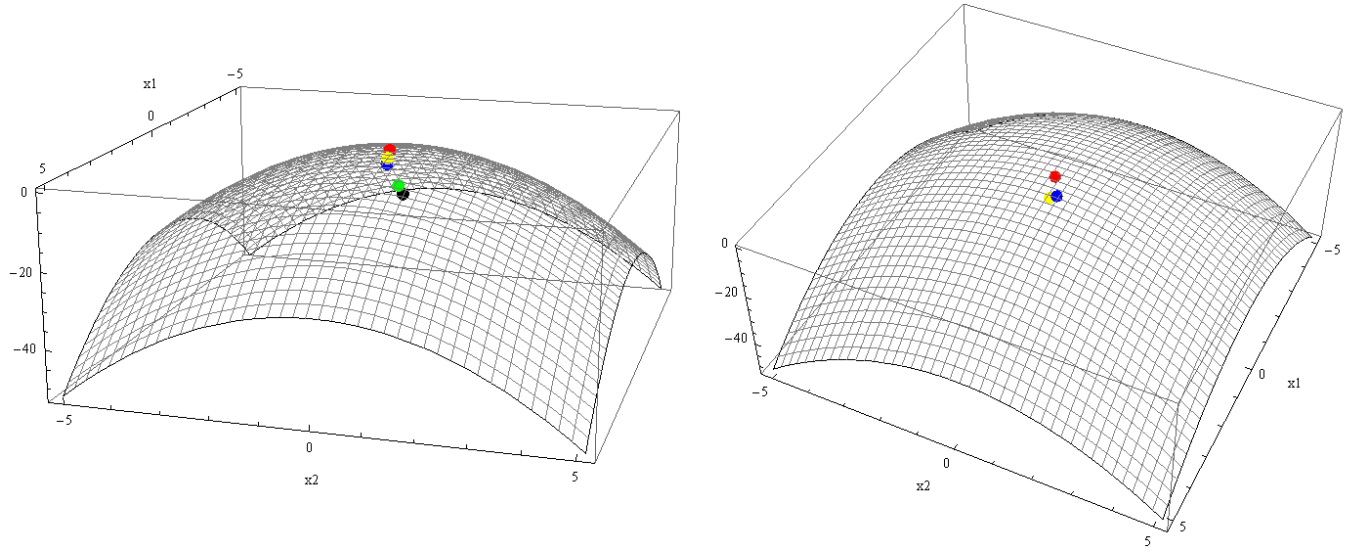

Fig. 7. First De Jong function optimization results; on the left original image and on the right a close-up, both for GA (black), GA+Baldwin Effect (green), HGA(pink), HGA+Baldwin Effect (red), CSA (yellow), FA (blue) calculated for 300 individuals in 5000 iterations

[16] D. Słota, "Reconstruction of the boundary condition in the problem of the binary alloy solidification," *Arch. Metall. Mater.*, vol. 56, pp. 279–285, 2011.

[17] E. Hetmaniok, D. Słota, and A. Zielonka, "Determination of the heat transfer coefficient by using the ant colony optimization algorithm," in *Parallel Processing and Applied Mathematics, Part I*, ser. LNCS, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wa/sniewski, Eds., vol. 7203. Springer, 2012, pp. 470–479.

[18] E. Hetmaniok, I. Nowak, D. Słota, and A. Zielonka, "Determination of optimal parameters for the immune algorithm used for solving inverse heat conduction problems with and without a phase change," *Numer. Heat Transfer B*, vol. 62, pp. 462–478, 2012.

[19] G. Borowik, T. Łuba, and D. Zydek, "Features reduction using logic minimization techniques," *International Journal of Electronics and Telecommunications*, vol. 58, no. 1, pp. 71–76, 2012.

[20] G. Borowik, T. Łuba, and P. Tomaszewicz, "On memory capacity to implement logic functions," *Computer Aided Systems Theory – EUROCAST 2011*, LNCS 6928, PART II, pp. 343–350, 2012.

[21] G. Borowik and A. Krasniewski, "A Tool for Trading-Off On-Line Error Detection Efficiency with Implementation Cost for Sequential Logic Implemented in FPGAs," in *21st International Conference on Systems Engineering – ICSEng 2011*. 16-18 August, Las Vegas, USA: IEEE, 2011, pp. 488–489.

[22] J. Baldwin, "A new factor in evolution," *The American Naturalist*, vol. 30, no. 354, pp. 441–451, 1986.

[23] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics*, vol. 266, no. 2, pp. 1830–1844, 2007.

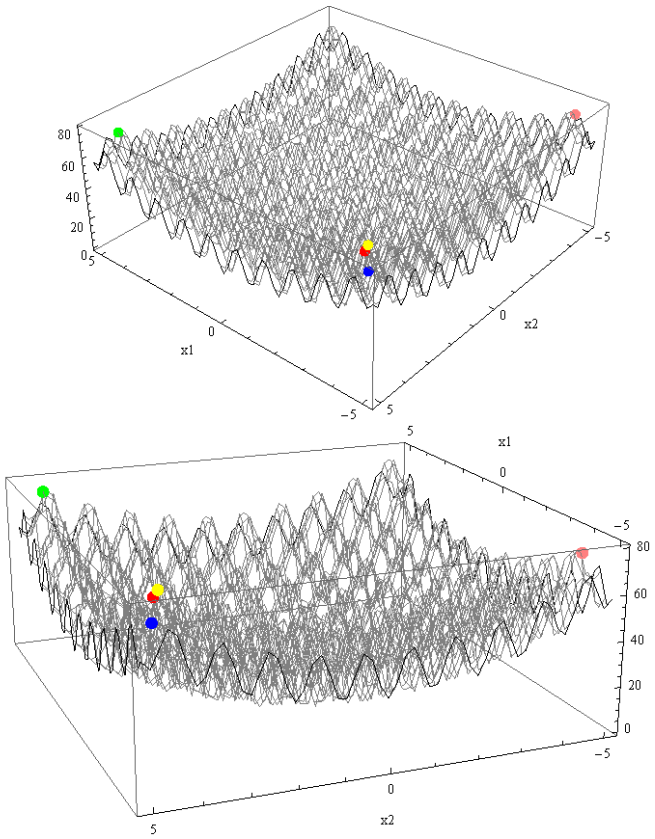

Fig. 8. Rastragin function optimization results: on the left original image and on the right a close-up, both for GA (black), GA+Baldwin Effect (green), HGA(pink), HGA+Baldwin Effect (red), CSA (yellow), FA (blue) calculated for 300 individuals in 100 iterations

TABLE I
RESEARCH RESULTS FOR RASTRAGIN FUNCTION (19)

Genetic Algorithm

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-4,65E+00;-4,62E+00;-4,51E+00;4,53E+00;-4,46E+00)	1,96E+02
1000	(4,54E+00;-4,48E+00;-4,50E+00;-4,52E+00;4,51E+00)	2,01E+02
50000	(4,52E+00;-4,52E+00;4,53E+00;4,53E+00;-4,52E+00)	2,02E+02

Hybrid Genetic Algorithm

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-4,65E+00;-4,62E+00;-4,51E+00;4,53E+00;-4,46E+00)	1,96E+02
1000	(4,54E+00;-4,48E+00;-4,50E+00;-4,52E+00;4,51E+00)	2,01E+02
50000	(4,52E+00;-4,52E+00;4,53E+00;4,53E+00;-4,52E+00)	2,02E+02

Genetic Algorithm with Baldwin Effect

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-4,46E+00;4,64E+00;4,51E+00;-4,51E+00;-4,43E+00)	1,97E+02
1000	(4,53E+00;-4,51E+00;-4,57E+00;-4,47E+00;4,50E+00)	2,01E+02
50000	(-4,53E+00;4,52E+00;4,53E+00;4,53E+00;-4,52E+00)	2,02E+02

Hybrid Genetic Algorithm with Baldwin Effect

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-4,46E+00;4,64E+00;4,51E+00;-4,51E+00;-4,43E+00)	1,97E+02
1000	(4,53E+00;-4,51E+00;-4,57E+00;-4,47E+00;4,50E+00)	2,01E+02
50000	(-4,53E+00;4,52E+00;4,53E+00;4,53E+00;-4,52E+00)	2,02E+02

Cuckoo Search Algorithm

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-4,49E+00;4,56E+00;4,53E+00;-4,52E+00;-4,61E-01)	1,81E+02
1000	(4,45E+00;4,97E+00;-4,55E+00;-4,53E+00;4,39E+00)	1,82E+02
50000	(-4,60E+00;-4,48E+00;4,77E+00;4,59E+00;-4,51E+00)	1,90E+02

Firefly Algorithm

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-4,67E+00;4,75E+00;4,23E+00;4,32E+00;4,80E+00)	1,58E+02
1000	(-4,44E+00;4,88E+00;5,03E+00 4,79E+00;4,43E+00)	1,60E+02
50000	(4,54E+00;-4,43E+00;4,97E+00;4,49E+00;4,68E+00)	1,80E+02

TABLE II
RESEARCH RESULTS FOR DE JONG FIRST FUNCTION (20)

Genetic Algorithm

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-8,32E-02;2,57E-01;-9,30E-02;3,11E-02;-2,26E-02)	-8,32E-02
1000	(-3,89E-02;4,29E-02;-1,76E-02;-4,57E-02;5,02E-02)	-8,27E-03
50000	(9,67E-05;-2,11E-03;4,94E-04;1,23E-03;1,74E-03)	-9,26E-06

Hybrid Genetic Algorithm

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(6,72E-16;-2,08E-15;7,52E-16;-2,51E-16;1,83E-16)	-5.4E-30
1000	(-3,07E-13;3,39E-13;-1,39E-13;-3,13E-13;3,97E-13)	-5,17E-25
50000	(6,40E-19;-1,40E-17;3,27E-18;8,12E-18;1,15E-17)	-4,05E-34

Genetic Algorithm with Baldwin Effect

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(8,61E-02;2,07E-01;-1,76E-01;5,81E-02;-1,49E-01)	-1,07E-01
1000	(-8,19E-02;2,24E-03;-7,69E-02;5,18E-02;9,12E-02)	-2,36E-02
50000	(-1,88E-03;1,59E-03;9,06E-03;-5,49E-04;-2,51E-03)	-9,47E-05

Hybrid Genetic Algorithm with Baldwin Effect

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-6,46E-14;-1,56E-13;1,32E-13;-4,36E-14;1,12E-13)	-6,02E-26
1000	(-5,29E-14;1,45E-15;-4,97E-14;3,35E-14;5,90E-14)	-9,88E-27
50000	(1,83E-12;-1,54E-12;-8,82E-12;5,34E-13;2,44E-12)	-8,97E-23

Cuckoo Search Algorithm

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-1,83E-01;-3,97E-01;1,28E-01;-5,39E-01;-2,59E-01)	-5,66E-01
1000	(-5,71E-01;-3,49E-01;-8,84E-02;6,01E-01;9,33E-02)	-8,26E-01
50000	(-4,29E-01;2,15E-01;1,14E-01;2,36E-02;1,45E-01)	-2,65E-01

Firefly Algorithm

iterations	optimum $\mathbf{x}$	$\phi_1(\mathbf{x})$
100	(-8,47E-01;5,79E-01;-3,21E-02;9,97E-02;1,04E-01)	1,07E+00
1000	(-3,50E-01;-6,58E-01;-3,39E-01;1,01E-03;2,43E-02)	6,72E-01
50000	(-4,03E-01;-6,93E-01;-3,98E-01;6,12E-03;8,15E-02)	8,07E-01

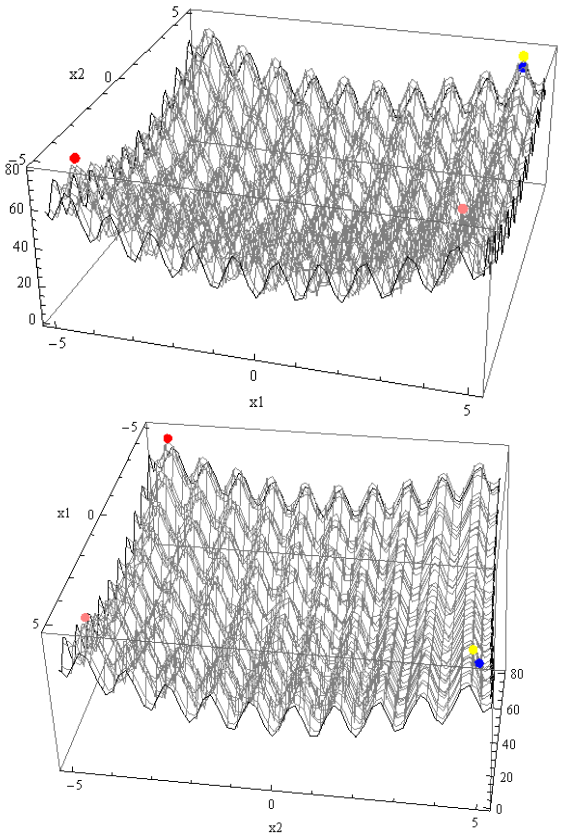

Fig. 9. Rastragin function optimization results: on the left original image and on the right a close-up, both for GA (black), GA+Baldwin Effect (green), HGA(pink), HGA+Baldwin Effect (red), CSA (yellow), FA (blue) calculated for 300 individuals in 5000 iterations