Selected Heuristic Algorithms Analysis and Comparison in Solving Optimization Problems

Bartłomiej Szlachta
Faculty of Applied Mathematics
Silesian University of Technology
Kaszubska 23, 44-100 Gliwice, Poland
Email: bartek01570@gmail.com

Kamil Rusin
Faculty of Applied Mathematics
Silesian University of Technology
Kaszubska 23, 44-100 Gliwice, Poland
Email: kamirus323@student.polsl.pl

Łukasz Płaneta
Faculty of Applied Mathematics
Silesian University of Technology
Kaszubska 23, 44-100 Gliwice, Poland
Email: luke.planeta@gmail.com

***Abstract*—Three examples of heuristic algorithms: The Firefly Algorithm, The Artificial Bee Colony Algorithm and The Genetic Algorithm, have been tested in regard to the ability of solving four testing cost functions. In this paper the algorithms are explained and the results are discussed and compared.**

***Keywords*—metaheuristics, heuristic algorithms, firefly algorithm, artificial bee colony algorithm, genetic algorithm, optimization problem**

I. Introduction

Heuristic algorithms are often used for optimization purposes, when a space of solutions is complex and common methods are imprecise these algorithms may help. We can find various application of heuristics in real world problems like image processing [9], [10]; games playability [11]; cognitive aspects of political sciences [12]; terminal slider control [13]; metal annihilating processes [14]. When it comes to the cost function, it is sometimes difficult or it is too time-consuming to find its minimum using mathematical methods. By the minimum of cost we understand a point for which the value of function is the lowest. Heuristic algorithms are very useful for finding solutions to problems which are difficult to solve in a usual way. Heuristic algorithms can be seen as help which randomize a definite number of points and move them in a specific way. As a result, the points group in the area of the minimum of the function. We will never be able to determine the minimal value of the function. However, the determined value should be precise enough.

Each algorithm works in a different way, so one defined algorithm which is able to solve all the optimization problems does not exist. For different functions some of them perform better than others, so the testing results of vary from each other. It is our responsibility to choose an algorithm and its coefficients to be the most suitable to find the best value of objective function. The only way to choose them is to apply trial and error method.

II. Testing functions

To check the results, we use four binary testing functions:

1) *Rosenbrock function:* Considered at a range from -3 to 3, it has a global minimum at f (1, 1) = 0 and is calculated as:

$$f(x,y) = (1-x)^2 + 100\cdot(y-x^2)^2 \tag{1}$$

2) *Beale function:* Considered at a range from -4.5 to 4.5, it has a global minimum at f (3, 0.5) = 0 and is calculated as:

$$f(x,y) = \left(1{,}5 - x(1-y)\right)^2 + \left(2{,}25 - x(1-y^2)\right)^2 + \left(2{,}625 - x(1-y^3)\right)^2 \tag{2}$$

3) *Himmelblau function:* Considered at a range from -10 to 10, it has four global minimums:
f (3, 2) = f (-2,805118, 3,131312) = f (-3,779310, -3,283186) = f (3,584428, -1,848126) = 0 and is calculated as:

$$f(x;\ y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{3}$$

4) *Levy function n. 13:* Considered at a range from -4.5 to 4.5, it has a global minimum at f (0, 0) = 0 and is calculated as:

$$f(x,y) = (\sin(3\pi x))^2 + (x-1)^2(1 + (\sin 3\pi y)^2) + (y-1)^2(1 + (\sin 2\pi y)^2) \tag{4}$$

III. Firefly Algorithm

A. *Fireflies' behaviour / Genesis*

The firefly algorithm (FA) is an example of nature-inspired heuristic algorithm, proposed by Xin-She Yand in [1] with some interesting applications [2], [3]. It is inspired by the behaviour of fireflies, beetles from the family of Lampyridae. Each of them emits a bioluminescent light from abdomen and uses it to attract the less bright fireflies.

B. *Method of operations*

To connect the FA algorithm with the optimization problem, the input of a cost function is interpreted as the co-ordinates of a firefly in a two-dimensional space. Then, for every point of the space the cost function value can be calculated for its co-ordinates. Also, it can be said that a firefly has a ‘f’ value, which means the value for the point the firefly is located at. The main objective of the algorithm is to find the best point – that with the smallest ‘f’.

Firstly, the population of fireflies needs to be generated randomly before the algorithm runs. Then, the algorithm should be launched and repeated a certain number of times. During every iteration, each firefly is moved, one after the other, in the direction of better ones in order to find the best ‘f’ value. There is always a random factor added to the movement. If a firefly is the best of all, the random step is the only factor responsible for moving a firefly.

C. *Pseudocode*

In our implementation, the FA algorithm works in accordance with the pseudocode showed below:

Input:

- Attributes of the A_i fireflies colony:

a) Co-ordinates (x_i, y_i),

b) Value ‘f_i’ of the cost function, calculated for (x_i, y_i), co-ordinates.

- Population size ‘n’, a natural number,
- Maximum attractiveness ’β_0’, a real number in the range (0, 1],
- Absorption coefficient ‘γ’, a real number in the range (0, 1],
- Random step size ‘a’, a real positive number.

Output:

- The co-ordinates (x_i, y_i) and the ‘f_i’ value of the best firefly,
- Changes in every firefly’s attributes.

Calculations:

```
i = 0
    while i < n do
    j = 0
        while j < n do
        Counting the distance between $A_i$ and $A_j$
        Assigning an attractiveness to $A_j$
        if fi < fj then
            Moving $A_i$ in the direction of $A_j$
        end if
        Moving $A_i$ randomly.
        Calculating fi value for the new $A_i$ co-ordinates
        end while
    end while
    Comparing ‘$f_i$’ values for each fireflies
end
```

D. *Details*

- The distance between two fireflies A_i and A_j is obtained by using formula:

$$r = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (5)$$

- The A_j attractiveness is determined by equation:

$$\beta = \beta_0 e^{-\gamma r^2} \quad (6)$$

- The distance that A_i is moved in A_j direction is calculated as:

$$s = \beta \cdot r \quad (7)$$

- The random move of A_i is defined by a vector which co-ordinates are random real numbers from the range [-a, a].

E. *Observations*

The consecutive algorithm iterations results can be split into two phases:

1) *Grouping fireflies together:* In this part each iteration of the algorithm results in the fireflies getting closer and closer to each other and the distance between the most distant pair of fireflies declines. The middle of the group moves randomly. There is also a possibility of creating more than one group, each at local minimums.

2) *Movement of the group:* Now the fireflies are moving randomly with the width of the group staying approximately constant. The whole group can move consistently in the direction of the best firefly when a better ‘f’ value is found.

The phases are different for various algorithm coefficients values. Their adequate choice is essential in order to find the global minimum as precisely as possible.

F. *Analysis*

For different coefficients values the algorithm returns different results. That can be analysed in order to find the coefficients values adequate to a function and a need.

The fireflies’ movement consists of two factors: attraction by the others and the random step. The former is very essential for the initial iterations when the distance between fireflies is long. During the latter, fireflies are located so close to each other that the attraction impact is responsible only for keeping the group together and does not help with searching cost function global minima. Those statements can be justified by mathematical calculations.

By connecting the formula (6) with the formula (7) we get a function s(r) which describes the distance the fireflies cover depending on the initial distance between them:

$$s(r) = \beta_0 \cdot r \cdot e^{-\gamma r^2} \quad (8)$$

Value of s(r) function (8) depends, for example, on 'β_0' and 'γ'. Consequently, s(r) can be increased by increasing 'β_0' value and decreasing 'γ' value and vice versa.

Having calculated the first derivative of (8), we can analyse monotonicity of s(r). With the initial distance declining, starting from the zero value in the infinity, it appreciates on value, reaches a peak at the distance r_0 shown below and declines to 0 at the 0 distance.

$$s(r_0) = s\left(\frac{1}{\sqrt{2\gamma}}\right) = \frac{\beta_0}{e^2\sqrt{2\gamma}} \tag{9}$$

So, the distance A_i firefly covers is the longest when A_j firefly is located within r_0 initial distance of Ai. That means that by reducing absorption coefficient 'γ' value, we increase r_0 value. The size of population 'n' has also impact on r_0 because the more fireflies exist, the higher probability of one staying at r_0 is. On the other hand, a big amount of fireflies can slow down a computer because it would need to make many more calculations.

It needs to be mentioned that the s(r) value needs to be much more significant than the random step 'a' value. Otherwise, the fireflies will not attract each other enough and they will split into two or more groups facing the local, not global minimums.

During the second phase fireflies are located so close to each other that s(r) values are insignificant compared to 'a' and are responsible only for keeping the group together – when a firefly moves too far, s(r) soars and brings the firefly back to the group.

When a randomly moved firefly hits the 'f' value better than the found so far, it attracts all other fireflies. As a result, the group is moving in its direction. The global minimum of the cost function is located approximately in the middle of the group and the determination of the minimum 'f' value with an acceptable accuracy depends mostly on luck.

G. *Conclusions*

1) To minimise the time required for the first section, increase 'n' and 'β_0' values, reduce 'a' value and set an appropriate value of 'γ'.
2) To reduce a risk of fireflies splitting into many groups, try increasing 'γ' and 'β_0' values and reducing 'a' value.
3) To reduce the time required for the second section, increase 'a' value.
4) To determine the minimum value with the better accuracy, increase the size of population 'n' and reduce 'a' value, but not exactly to 0.

Artificial Bee Colony Algorithm

H. *Bees' behaviour / Genesis*

Artificial Bee Colony algorithm (ABC) was proposed and implemented by Dervis Karaboga in [4]. He was inspired by a bee swarm behaviour and their way to find the best food source in the neighbourhood. ABC is based on population of a swarm and defined food sources. It is usually used for optimization problems. There are many articles reporting this method in various optimization aspects [5] – [8]. Bees are very cooperative and thanks to it, the swarm is able to perform tasks successfully. The whole population is divided into three groups: employed bees, onlooker bees and scout bees. First of them are looking for food sources. Meanwhile, they are also sharing information with other bees about the best ones they have found so far. The second group of bees observes them and decides to follow them to the food sources with the best fitness. The better the source, the higher possibility of choosing it. Employed bees can also change their food sources. They may be encouraged to follow other bees to their food sources. However, if an employed bee decides to leave its current food source and go to other randomly selected one, then it is named a scout bee. Generally, a population of the swarm in ABC algorithm is divided into half. Employed bees are the first half, whereas onlooking bees are in the other one. Over time, artificial bees discover better and better food sources to finally find the ones with the largest amount of nectar. Then the co-ordinates of its position represent the best solution for our optimization problem.

I. *Pseudocode*

In our implementation, the ABC algorithm works in accordance with the pseudocode showed below:

Input:

- Population of the colony 'n', a natural number,
- Number of iterations, a natural number
- Limit of chances for food source 't', a natural number,
- Number of food sources 'o', a natural number.

Output:

- The co-ordinates (x_i, y_i) and value of the best food source.

Phases of the algorithm:

```
Initialization Phase
Setting number of food sources, population, functions parameters.
Repeat for every iteration
        Employed bees phase
        Onlooker bees phase
        Scouts bees phase
        Memorizing the best solution achieved so far
After
        Writing the co-ordinates and value of the best solution
```

J. *Initialization phase*

Firstly, variables for algorithm must be set, e.g. population of a swarm, limit of chances for each food source. We can also change the number of food sources, which optimal value equals the half of the population. Then, we must adjust parameters for a function we want to optimize, e.g. upper and lower bound. During this phase all food sources are initialized by scout bees. Every food source represents possible solution for our optimization problem. The following equation might be used for initialization:

$$x_i = l_b + rand(0,1) \cdot (u_b - l_b) \tag{10}$$

where ‘l_b’ is lower and ‘u_b’ is upper bound of our parameter ‘x_i’.

K. *Employed bees phase*

Employed bees look for a new food source that might have a greater amount of nectar in a close neighbourhood of their last food source. The value of food source is set using the following equation:

$$v_i = x_i + \sigma \cdot (x_i - x_k) \tag{11}$$

where ‘i’ is a randomly chosen parameter index, ‘x_i’ is a randomly selected food source and ‘σ’ represents a random number from range [-1,1].

Simultaneously, fitness of every food source is being computed. After these processes a bee makes a choice. The fitness value may be calculated using the following formula:

$$f(x) = \begin{cases} \frac{1}{1+f(x_i)}, & f(x_i) \geq 0 \\ 1 + |f(x_i)|, & f(x_i) < 0 \end{cases} \tag{12}$$

where $f(x_i)$ is a value of solution ‘x_i’.

L. *Onlooker bees phase*

Onlooker bees wait in the swarm for information about food sources provided by employed bees. Then a probability is calculated. It is based on the fitness value, given them by employed bees. This probability means that onlooker bees are more willing to exploit food sources with higher fitness value. It can be calculated using a following formula:

$$prob(i) = \frac{fit(i)}{\sum_{i=1}^{s} fit(i)} \tag{13}$$

where ‘fit(i)’ means fitness of food source with index ‘i’ and ‘s’ means a number of food sources. Afterwards a neighbouring source is calculated from (10) and fitness value from (12). The onlooker bee must choose one of two. The more bees are recruited to better food sources, the more positive feedback they have.

M. *Scout bees phase*

In this phase unemployed bees choose a new food source randomly. Scouts are bees, which abandoned their solutions due to chance limit set at the beginning of the algorithm. This means that their solutions could not be upgraded. What is more, they choose next food sources as well as share negative opinions on abandoned food sources, so that it will balance positive ones.

N. *Observations*

The proper number of food sources equals a half of the swarm population. If we keep reducing the number of food sources, we will have worse and worse solutions. When we increase bee population, it does not always go hand in hand with getting a more accurate solution. To obtain optimal chances we limit ‘o’ to 100. There is a slight difference, when we change this parameter.

Genetic algorithm

A. Genesis

Genetic algorithm (GA) was proposed by Alan Turing in 1950. He was inspired by Charles Darwin's theory of natural evolution. It was developed by John Holland and colleagues at the University of Michigan in [18]. Nowadays we can find its various improved versions reported in many articles [15]-[17]. GA is based on population which uses natural selection. It is used for optimization problems. Algorithm relies on biological operations: selection, mutation and crossover, as in natural genetics. Firstly, whole population is selected by their fitness score (degree of adaptation). The higher the score, the higher chance of surviving to the next generation. The next generation is composed of parents, who survived and their children. Children are created by crossover or mutation.

B. Pseudocode

I our implementation, the GA works in accordance with the pseudocode shown below:

Input:

- Population of the generation, a natural number,
- Number of iterations, a natural number,
- Two chromosomes for each individual.

Output:

- Two chromosomes of the best individual and fitness value.

```
Phases of the algorithm:
Initialization phase
Setting number of populations, their chromosomes and
function parameters
Repeating for every generation:
        Selection
        Setting the best individual and elite individuals
Crossover
Mutation

After
Write the best fitness score and chromosomes
```

C. Details

- The leader of the population has the highest chance of reproduction. The elite of the population have lower chance which is still high. A typical individual has the lowest chance of reproduction. This algorithm is based on herd selection, where Alpha male, some group of elite individuals and the rest of the population appear. The reproduction probability for the leader can be calculated using a followed formula:

$$prob(leader) = \frac{1}{2} + \frac{1}{elite} + \frac{1}{population} \qquad (14)$$

where:

elite - number of individuals in elite group
population - number of population

- All chromosomes before crossover or mutation are translated into genes. In our case decimal numbers are translated into binary numbers.
- We used 3 crossover variants: 1-Point Crossover, Destructive Crossover and Constructive Crossover. In 1-Point Crossover offspring is created by exchanging the genes of parents.

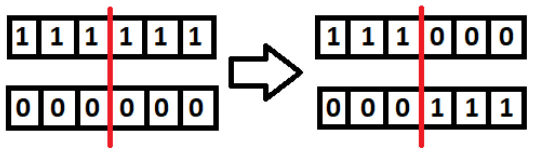

Fig. 1. Typical Crossover

In Destructive Crossover only positive genes in both chromosomes are reproduced.

In Constructive Crossover if one of genes is positive, then offspring's gene is also positive.

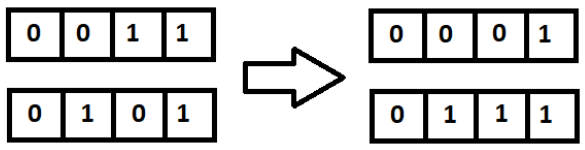

Fig. 2. Destructive and Constructive Crossover

Left side: parents

Right side: up - destructive, down - constructive crossover.

In Crossover, it is also interesting how positive and negative value is inherited. Here , we applied the methodology of inheritance Rh factor from blood group system. When two positive chromosomes cross, the offspring is positive in 15 of 16 cases. But, if one is positive and the other is negative, the positive individual appears 3 times per 4. If both parents are negative, the offspring is always negative.

Operation of mutation inverts random genes. It is used for mutating parents or children that have been created by crossover.

D. Observations

We can find two kinds of behaviour:

- whole population does not evolve for the most of the time
- rare but really efficient 'jumps', where the best individual evolves

That is really interesting and the solution of this may be in combination of two algorithms: destructive crossover and mutation. GA works the best with Rosenbrock function, but it did not look well until 70th generation. The 'jump' occurred where chromosome evolved. In Baele function GA jumps are only in the beginning and are not spectacular. Himmemblau function is definitely the hardest to our GA. After the 10th generation the algorithm slows down and starts again before the end. In Levy's function we can observe quick start and huge jump in the middle generation.

E. Conclusions

1) GA is very useful to find the surroundings of solution, but after some upgrades, it could be more accurate
2) Our version of GA is looking for solution in whole range, not in closer environment
3) A lot of results after mutations and crossover are useless due to thoughtless randomizing algorithm

BENCHMARK TESTS

Plots present the smallest 'f' value found so far depending on the number of iterations done for each testing functions. FA algorithm's results are indicated by green lines, ABC Algorithm's by the blue ones and GA Algorithm's by the yellow ones. [B's sent.]

Green line indicates the values of Firefly algorithm, while blue line represents Artificial Bee Colony. [K's sent.]

FA Algorithm data was taken for following coefficients' values: 'n' = 100; 'β_0' = 0,1; 'γ' = 0,1; 'a' = 0,01.

ABC algorithm data was taken for following coefficients' values: 'n' = 100, 't' = 50, 'o' = 100.

GA algorithm data was taken for following coefficients' values: 'n' = 100, 'o' = 100, crossover probability = 0,5, mutation probability for not-crossover= 0,5, mutation probability for crossover = 0,1, probability of mutation single gene = 0,1, in crossover positive sign is for: two positive = 15/16, two different = ¾, two negative 0.

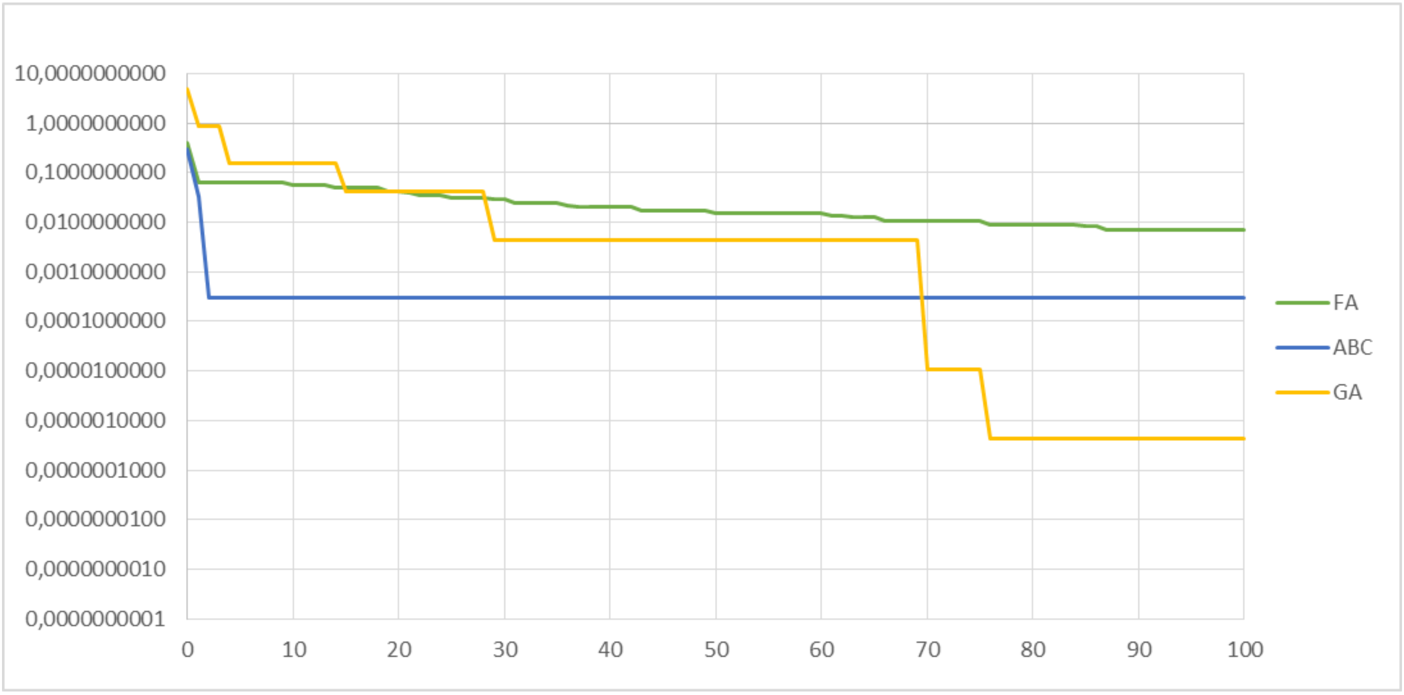

Fig. 3. The best value found during consecutive iterations. (*Rosenbrock function*)

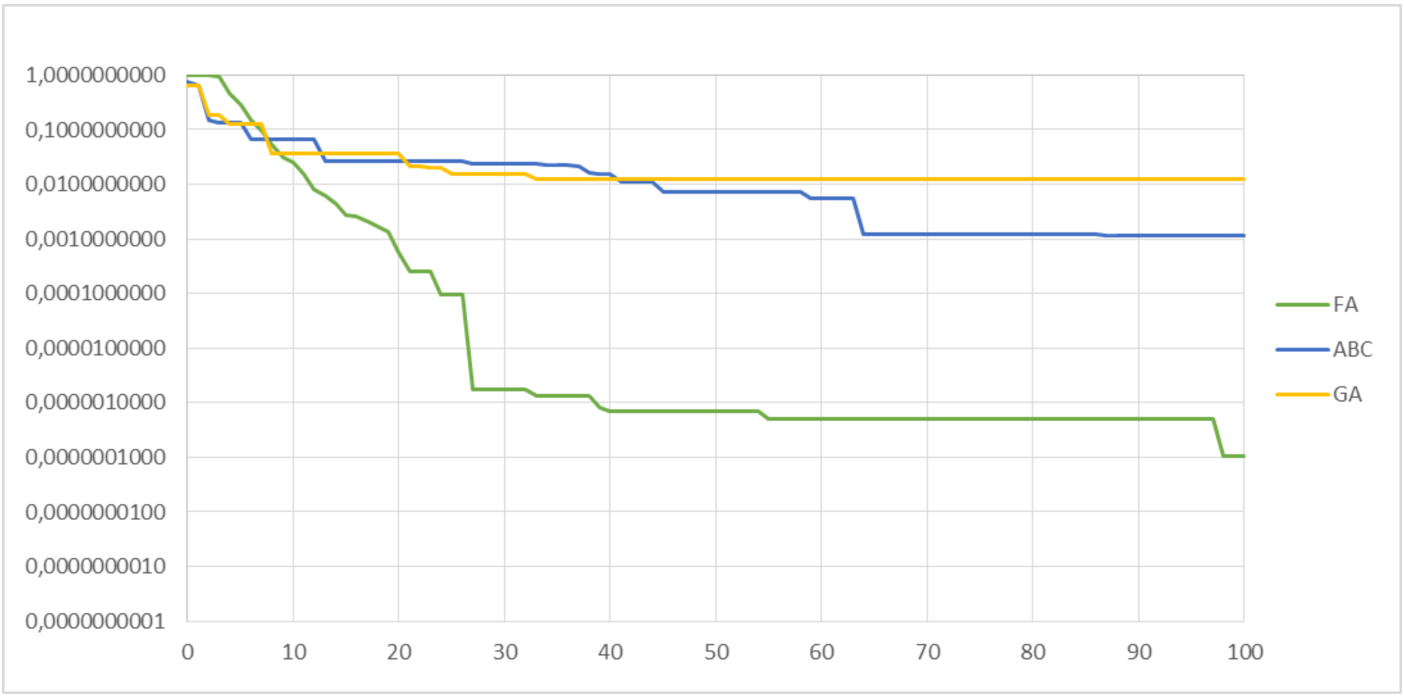

Fig. 4. The best value found during consecutive iterations. (*Beale function*)

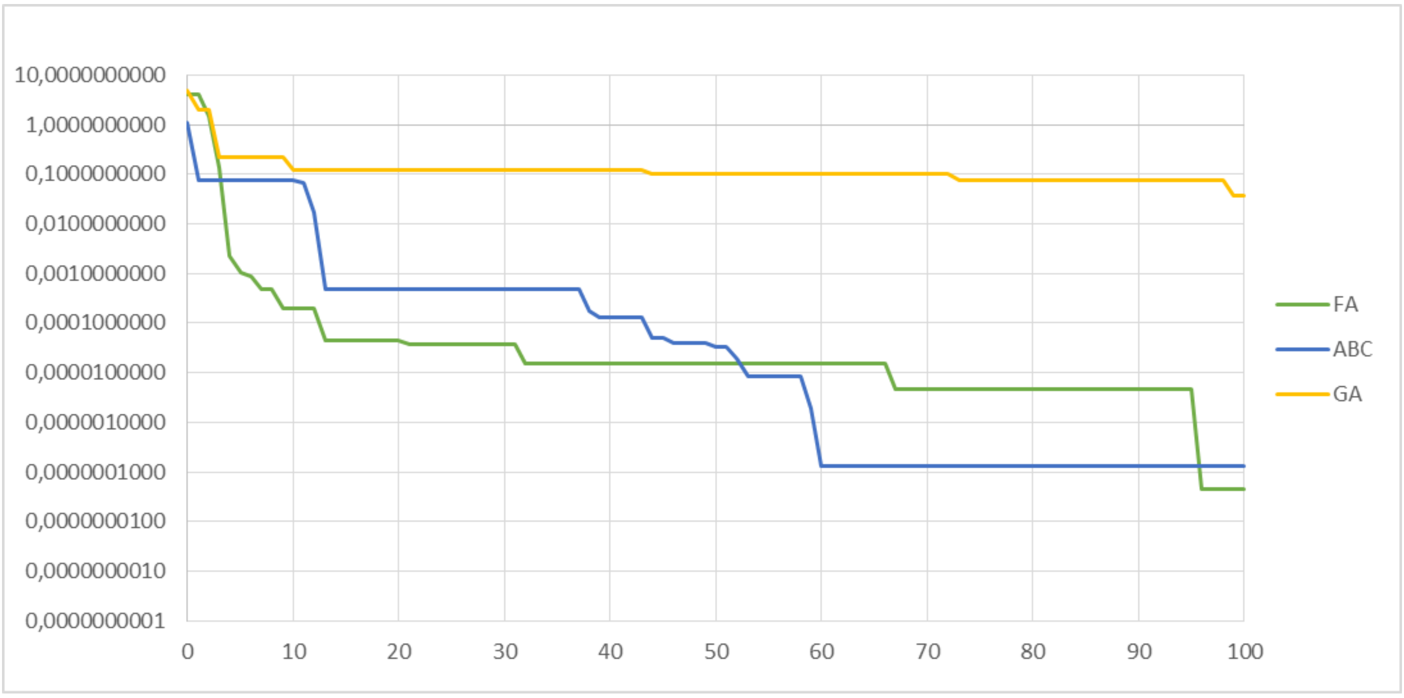

Fig. 5. The best value found during consecutive iterations. *(Himmelblau function)*

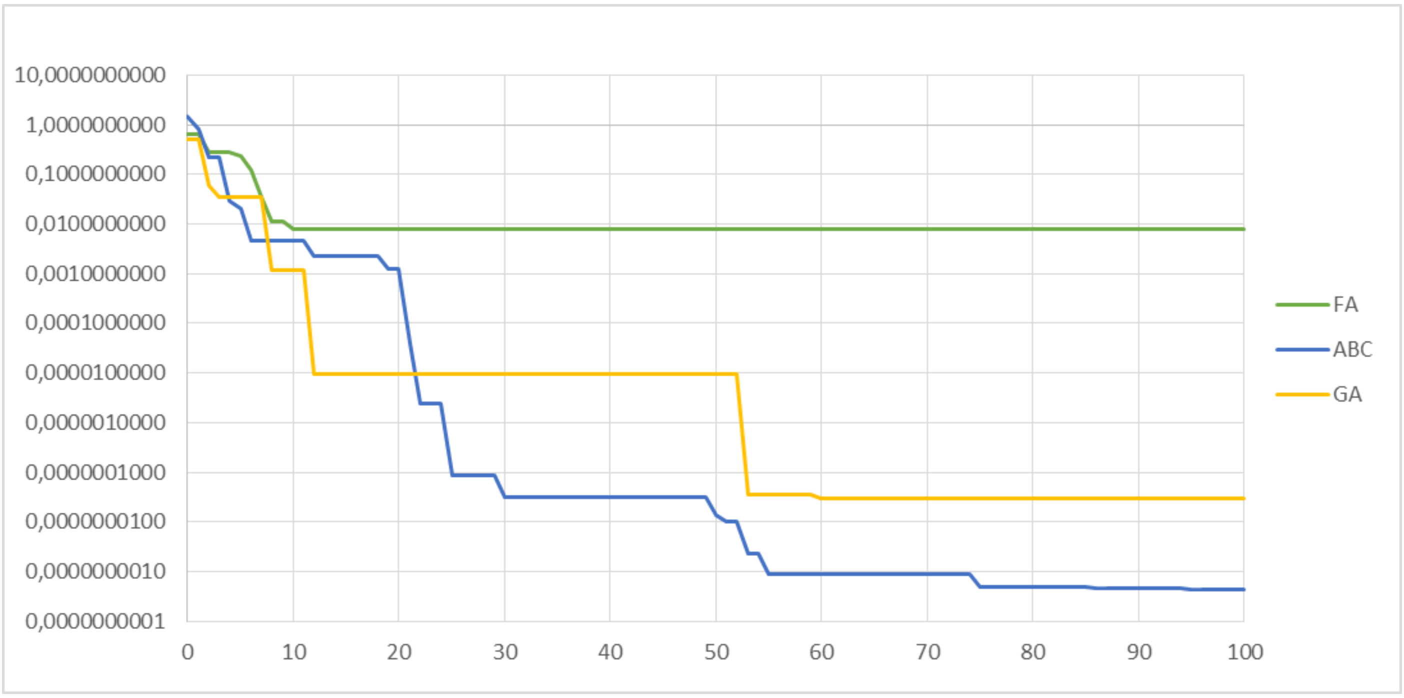

Fig. 6. The best value found during consecutive iterations. (*Levy function n. 13*)

Final remarks

The graphs show the smallest values found for each of the test functions depending on the number of iterations of each algorithm. Population for our algorithms was set to 100 individuals. Other parameters are incomparable and determined differently.

We tested Rosenbrock function first. As we can see FA generated higher value at the beginning and a value at 100s iteration exceeded a final value of ABC algorithm. It was less accurate by almost two decimal places. ABC algorithm line plummeted and at second iteration reached its lowest value. Genetic algorithm performed the best in this function. It was more accurate by three decimal places than ABC.

The second graph shows Beal's function, in which FA dominated over ABC algorithm and Genetic algorithm. Even though it started with a slightly higher function value than the others, it started to decline very quickly and reached highly precise value of seven decimal places. This is an impressive result.

On the third graph we can see comparable results of FA and ABC algorithms. The values declined at different moments. ABC algorithm reached constant value at 60th iteration, whereas FA fell finally at 95th iteration and had a slightly more accurate function value. Genetic algorithm was not precise and its calculation was inaccurate, its precision was by two decimal places, while the ABC and FA algorithms reached values with 7 and 8 decimal places, respectively.

FA did not perform well at Levy's function. At 10 iteration it reached a static value, which was highly inaccurate comparing to ABC algorithm. It had precision of two decimal places, while the second algorithm reached five times better result. Genetic algorithm reached an interesting precision, however, it was still less accurate than ABC algorithm.

Possible application

Heuristic algorithms can be applied in many practical situations when an optimization is needed. To use an heuristic algorithm, first we need to find a cost function formula. Also we must keep in mind that there is a risk that the algorithm will not find a local minimum. Then we should try to apply another algorithm.

Today heuristic algorithms may be used for example in market analysis. It uses the fact that when a bee in ABC algorithm moves to a better solution than the previous one. That behaviour may be easily applied thanks to trend functions of specific stock chart. At the defined range, algorithm knows when it is hitting the lowest and the highest value. Specially designed bot could sell or buy shares of a company we are following. Algorithms may be also used in neuron networks that would learn to solve issues faster than usually. Neurons would correspond with each other and work with better quality so the data would be interpreted more precisely.

Genetic algorithm is used in many processes which use a biological algorithm because of its natural solutions usage. For example, robots are able to learn human behaviour. As a result, now they can perform actions which were reserved for humans only. It advances the moment when robots will replace physical works, which is called evolutionary robotics.

GA is also commonly used in training neural networks. The main benefit of this method is the fact that neuroevolution can find quicker ways of finding solutions. Typical supervised learning algorithms require data of correct input-output pairs. In contrast, genetic algorithm requires only a measure of a network's performance at a task. For example, in games we do not provide strategies (how to play), but only how to increase score. This mechanism makes GA easy to transfer from one program to the other one.

Genetic algorithm is also used in music record production. Algorithm uses known songs to create new ones. It is based on schemes that are most popular in the songs, recombining them to the fresh chanson.

REFERENCES

[1] YANG, Xin-She. Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. Springer, Berlin, Heidelberg, 2009. p. 169-178.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] WOŹNIAK, Marcin; MARSZAŁEK, Zbigniew. An idea to apply firefly algorithm in 2d image key-points search. In: International Conference on Information and Software Technologies. Springer, Cham, 2014. p. 312-323.

[3] NAPOLI, Christian, et al. Simplified firefly algorithm for 2d image key-points search. In: Computational Intelligence for Human-like Intelligence (CIHLI), 2014 IEEE Symposium on. IEEE, 2014. p. 1-8.

[4] KARABOGA, Dervis; BASTURK, Bahriye. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In: International fuzzy systems association world congress. Springer, Berlin, Heidelberg, 2007. p. 789-798.K. Elissa, “Title of paper if known,” unpublished.

[5] KARABOGA, Dervis. Artificial bee colony algorithm. scholarpedia, 2010, 5.3: 6915.

[6] Witanowski, Łukasz & Lampart, Piotr. (2015). Using of a bee algorithms for searching process of minimum of objective function. Mechanik. 570/971-570/978. 10.17814/mechanik.2015.7.318.

[7] Behzad Nozohour-leilabady, Babak Fazelabdolabadi, On the application of artificial bee colony (ABC) algorithm for optimization of well placements in fractured reservoirs; efficiency comparison with the particle swarm optimization (PSO) methodology, Petroleum, Volume 2, Issue 1, 2016, Pages 79-89, ISSN 2405-6561.

[8] NOZOHOUR-LEILABADY, Behzad; FAZELABDOLABADI, Babak. On the application of artificial bee colony (ABC) algorithm for optimization of well placements in fractured reservoirs; efficiency comparison with the particle swarm optimization (PSO) methodology. Petroleum, 2016, 2.1: 79-89.

[9] WOŹNIAK, Marcin; POŁAP, Dawid. Adaptive neuro-heuristic hybrid model for fruit peel defects detection. Neural Networks, 2018, 98: 16-33.

[10] Capizzi, G., Sciuto, G. L., Napoli, C., Shikler, R., & Woźniak, M. (2018). Optimizing the Organic Solar Cell Manufacturing Process by Means of AFM Measurements and Neural Networks. Energies, 11(5), 1-13.

[11] Napoli, C., Pappalardo, G., Tina, G. M., & Tramontana, E. (2016). Cooperative strategy for optimal management of smart grids by wavelet rnns and cloud computing. IEEE transactions on neural networks and learning systems, 27(8), 1672-1685.

[12] Bonanno, F., Capizzi, G., Coco, S., Napoli, C., Laudani, A., & Sciuto, G. L. (2014, June). Optimal thicknesses determination in a multilayer structure to improve the SPP efficiency for photovoltaic devices by an hybrid FEM—cascade neural network based approach. In International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 (pp. 355-362). IEEE.

[13] WOŹNIAK, Marcin; POŁAP, Dawid. Bio-inspired methods modeled for respiratory disease detection from medical images. Swarm and Evolutionary Computation, 2018.

[14] DESURVIRE, Heather; CAPLAN, Martin; TOTH, Jozsef A. Using heuristics to evaluate the playability of games. In: CHI'04 extended abstracts on Human factors in computing systems. ACM, 2004. p. 1509-1512.

[15] LAU, Richard R.; REDLAWSK, David P. Advantages and disadvantages of cognitive heuristics in political decision making. American Journal of Political Science, 2001, 951-971.

[16] REZAIE, Behrooz; GHASEMI, Hossein; RAHMANI, Zahra. Terminal Sliding Mode Controller Tuned Using Evolutionary Algorithms for Finite-Time Robust Tracking Control in a Class of Nonholonomic Systems. Information Technology And Control, 47.1: 26-44.

[17] BROCIEK, Rafal; SLOTA, Damian. Application of real ant colony optimization algorithm to solve space and time fractional heat conduction inverse problem. Information Technology And Control, 2017, 46.2: 171-182.

[18] DEB, Kalyanmoy, et al. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: International Conference on Parallel Problem Solving From Nature. Springer, Berlin, Heidelberg, 2000. p. 849-858.

[19] HORN, Jeffrey; NAFPLIOTIS, Nicholas; GOLDBERG, David E. A niched Pareto genetic algorithm for multiobjective optimization. In: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on. Ieee, 1994. p. 82-87.

[20] MORRIS, Garrett M., et al. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. Journal of computational chemistry, 1998, 19.14: 1639-1662.

[21] GOLDBERG, David E.; HOLLAND, John H. Genetic algorithms and machine learning. Machine learning, 1988, 3.2: 95-99.