SILESIAN UNIVERSITY OF TECHNOLOGY PUBLISHING HOUSE

SCIENTIFIC PAPERS OF SILESIAN UNIVERSITY OF TECHNOLOGY 2023
ORGANIZATION AND MANAGEMENT SERIES NO. 176

COMPARISON OF CERTAIN EVOLUTION-INSPIRED ALGORITHMS

Roman KLUGER

Silesian University of Technology; roman.kluger@polsl.pl, ORCID: 0000-0003-0892-9124

Purpose: This paper aims at making a comparison of three optimization algorithms – standard Genetic Algorithm and its two modifications: Extended Compact Genetic Algorithm and Population-based Incremental Learning.
Design/methodology/approach: To reach the objectives of the paper the solver based on algorithms was developed. Certain test functions were applied to test them and evaluate their performance.
Findings: Modifications of Genetic Algorithm reach optimal values faster and more precisely.
Research limitations/implications: Problem of optimization of certain cost functions frequently occurs in many management problems of organizing the optimal workflow in organizations. It can be used also in engineering problems of designing optimal devices at lowest possible cost.
Practical implications: One can optimize function faster using discussed algorithms than by using standard evolutionary algorithm.
Originality/value: The paper shows results of comparisons of three algorithms, discusses how tuning meta parameters helps to increase their efficiency and accuracy.

Keywords: optimization, genetic algorithms, applied mathematics.

Category of the paper: research paper.

1. Introduction

In the first half of 18th century one of the most notable mathematicians of all times – Leonhard Euler wrote that *nothing at all takes place in the universe in which some rule of maximum or minimum does not appear*. So, from the beginning of development of modern science the importance of the task of optimization of functions was well understood. Many calculus-based methods were developed to deal with this problem. But with the development of applications of mathematics plenty of functions needed to be optimized but they did not satisfy conditions for such methods. That being said new approach had to be developed. Plenty of different numerical methods were developed. They are not perfect, for they do not necessary provide correct answer at first approach, but after certain considerations enable to find local extreme values of some wild functions. Many of such algorithms are heuristics based on some biological phenomena. The algorithms can be inspired by e.g., Ant Colony, Beehive or Wolf Pack. In this paper we shall discuss three algorithms based on central biological theory - Genetic algorithm and its modifications.

The objective of this work is to compare three heuristic algorithms – Evolutionary Algorithm, Extended Compact Genetic Algorithm and Probability-based Incremental Learning based on their performance in the task of optimizing test functions. A thorough discussion of many variants of Evolutionary Algorithms can be found in (Tamilselvi, 2022) and (Vikhar, 2016). Application of variants of EA to engineering problem of optimal design can be found in (Kieszek et al. 2023) Interesting variant of ECGA using properties of machine-precision arithmetic can be found in (Satman, Akadal, 2020). Applications of PBIL are depicted in (Grisales-Noreña et al., 2016). It is even used to optimize nuclear reload (da Silva et al., 2018) As it was shown, the matter of different variant evolution-based algorithms is still a topic of active research. To the best of author's knowledge, no direct comparison of those three algorithms' performance has been published.

2. Methods

Population-based optimization heuristics have been perceived as efficient and universal methods of determining global extremes for a long time. The Evolutionary Algorithm is now considered a classic population-based heuristic. Despite its indisputable advantages, there are still attempts to modify it, aimed at increasing accuracy and reducing the number of assessments of individuals necessary to obtain a solution. The latter feature is particularly important in the case of optimization of complex problems where the evaluation of a single individual is very time-consuming. If individuals with low fitness appear relatively often in the population, their assessment takes time, but does not lead to a solution. Ensuring that low-value individuals appear infrequently is one way to speed up optimization.

In a large group of population-based algorithms, algorithms using probabilistic models have recently started to play an increasingly important role.

These are usually methods with a structure very similar to the structure of the evolutionary algorithm, with the difference that successive generations of individuals/solutions are generated based on a probabilistic model of the population of promising solutions, and not because of crossing or mutation of individuals from the current population.

The population of promising solutions is created from individuals selected because of classical selection (usually a tournament). In such a population there are individuals with a higher-than-average fitness, and the model built on their basis should promote those features of the solution that lead to the optimized goal.

The next generation of solutions is generated in a pseudo-random manner but taking into account the probabilistic model. This means that in methods of this type, the way the model is built is responsible for both the very convergence to the extreme and its pace.

In order to take full advantage of the features of the discussed methods, care should be taken to build the probabilistic model in such a way that, with effective convergence, the ability to properly search the space is not lost. If the population influences model changes in subsequent iterations too much, it may lead to rapid unification of the population and improper exploration of space. On the other hand, changing the model too slowly will make the optimization method look like a random search.

The way in which the model will be built is crucial from the point of view of this type of methods. Other elements of the algorithm, such as succession, usually have a classic form (known from GA) and are used to conduct an iterative process.

In the presented work, two optimization methods using the probabilistic model PBIL - Population based incremental learning and ECGA - Extended Compact Genetic Algorithm will be presented. Both methods represent population heuristics belonging to the group of methods considered as EA modifications.

2.1. Genetic Algorithm

Genetic Algorithm is based on evolutionary mechanisms such as reproduction, recombination, mutation, and selection. It mimics the development of species that occurs in real life. That development can be simplified to few observations:

1. There is some population of certain species.
2. Members of population that are best equipped to live in given conditions have the highest chance of reproduction.
3. Members of population that are not best equipped also have a chance of reproduction, but on the smaller scale – this fact is important in preserving some genomes that can be useful in different conditions.
4. Reproduction consists of mixing genomes of parents – it is not just the replication, but something new.
5. During reproduction some random changes in genome can occur.

To translate the observations some assumptions are needed. To simulate the environment a function called fitness function will be used. Members of population will be points inside the domain of the function. Then the algorithm can be expressed as follows:

1. Initialize metaparameters – size of population, probability of crossing, probability of mutation.
2. Randomly build starting population P^0.
3. Set $t = 0$.

```
4. do{
      a. Evaluate population $P^t$ according to fitness function.
      b. Select best members of $P^t$ and make temporary population $T^t$.
      c. Use genetic operators (crossing and mutation) on $T^t$ to produce $O^t$.
      d. Evaluate population $O^t$ according to fitness function.
      e. Produce $P^{t+1}$ by applying succession operators to $O^t$ and $P^t$.
      f. $t++$
      }
      while (stop condition).
5. Return the best fitting element in the last population.
```

The genetic algorithm was applied to continuous, multidimensional problems, so some genes to evolve had to be identified. In d-dimensional problems the coordinate in each of dimensions was treated like one independent gene. For parent vectors $\overrightarrow{\boldsymbol{x_1}}, \overrightarrow{\boldsymbol{x_2}}$ of $\mathbb{R}^d$ the following formula has been applied:

$$\vec{\boldsymbol{y}} = \overrightarrow{\boldsymbol{x_1}} + \vec{\boldsymbol{\xi}}_{U(0,1)} * (\overrightarrow{\boldsymbol{x_2}} - \overrightarrow{\boldsymbol{x_1}})$$

where:

$\vec{\boldsymbol{y}}$ is child vector,

$\vec{\boldsymbol{\xi}}_{U(0,1)}$ is random variable vector with uniform distribution,

$*$ denotes component-wise multiplication. It is called averaging crossing.

Another important genetic operation is random mutation of vector $\boldsymbol{x}$ which is modelled by equation:

$$\vec{\boldsymbol{y}} = \vec{\boldsymbol{x}} + \vec{\boldsymbol{\xi}}$$

where:

$\vec{\boldsymbol{y}}$ is child vector,

$\vec{\boldsymbol{x}}$ – parent vector,

$\vec{\boldsymbol{\xi}}$ – random vector of certain distribution. In our case it was modelled by Gaussian distribution.

Next operator was the selection operator. In the paper the so-called tournament selection was applied. To do this type of selection certain number of competitors had to be chosen from the population. Then the best fitting competitor was found, and it was assigned to the child population. So by number of tournaments the number of elements in child population could be regulated.

Last operator was the succession operator. To do this elitist succession was applied – current populations O^t and P^t were joined and assigned number of best members was set to child population.

2.2. Population-based Incremental Learning

PBIL is a modified Evolutionary Algorithm using a learning process based on "observation" of the current population. In the method, at each iteration step it builds new population using normal distribution with given mean value. In fact, the role of the probabilistic model is played by a vector $\vec{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)$whose components μ_i define mean value of the i-th component of the generated individual.

It is characteristic of the PBIL algorithm that only the best individual in the current generation is used to update the $\vec{\mu}$ vector, which means that the model is based on only one promising individual, marked $\vec{b}$. The value of the standard deviation in one iteration is the same for each component. The population for the next iterative step is generated on the basis of the model adopted in this way.

At the beginning of the iterative process, the components of the vector $\overrightarrow{\mu_0}$ are usually assumed to be random or determined from data about the constraints of the set of feasible solutions.

In subsequent iterations, the components of the $\vec{\mu}$ vector are updated as follows:

$$\mu_j^{(t+1)} = (1 - \lambda) \cdot \mu_j^{(t)} + \lambda \cdot b_j$$

where:

$\mu_j^{(t)}$ is the j-th component of the vector $\vec{\mu}$ in generation t,

b_j - the component of the promising vector $\vec{b}$,

λ - the so-called learning rate.

Individuals of the $t + 1$ population are always drawn considering the current probability vector. Contrary to the standard genetic algorithm, PBIL does not retain the best individual in the population, but the specificity of the procedure gives a great chance of selecting it, because it is on its basis that the probabilistic model is modified. However, it is also possible to use elite succession, which transfers a certain number of the best individuals from the current population to the next generation. Drawing the entire population, taking into account the model (represented by $\vec{\mu}$), gives a good chance of appearing more "good" individuals (from the point of view of the objective function) of individuals, usually better than in the previous generation.

The value of the learning coefficient λ is a parameter set at the beginning of the iterative process and affects its course. It should be remembered that its small value slows down the modification of the model, and its too high value may result in too fast unification of the population. The λ coefficient should be chosen to balance the ability for targeted exploration with the ability to exploit space.

Generally speaking, it exchanges genetic operators with probabilistic model. At each iteration step it builds new population it builds new population using normal distribution with given mean value.

The PBIL heuristic pseudocode is as follows:

```
procedure PBIL:
1. Initialize metaparameters – size of population, the starting mean value vector
   $\vec{\mu}_0 = \left(\mu_1^{(0)}, \mu_2^{(0)}, \dots, \mu_n^{(0)}\right)$ and standard deviation $\sigma_0$.
2. Randomly build starting population $P^0$ according to the normal distribution $N(\vec{\mu}_0, \sigma_0)$

3. Evaluate population $P^0$ according to fitness function.
4. Set $t = 0$
5. do {
    a. Select the best member of $P^t$(denoted as $\vec{b}$).
    b. Modification of the mean value vector coordinates according to the formula
       $\mu_j^{(t+1)} = (1 - \lambda) \cdot \mu_j^{(t)} + \lambda \cdot b_j$
    c. Modification of the standard deviation to $\sigma_{t+1}$.
    d. Generation of a new individuals according to the model $N(\vec{\mu}_{t+1}, \sigma_{t+1})$
       to produce $O^t$.
    e. Evaluate population $O^t$ according to fitness function.
    f. Produce $P^{t+1}$ by applying succession operators to $O^t$ and $P^t$.
    g. $t++$
    }
    while(stop condition).
6. Return the best fitting element in the last population.
```

2.3. Extended Compact Genetic Algorithm

The Extended Compact Genetic Algorithm (ECGA) is another variant of the classic evolutionary algorithm.

In this algorithm, because of selection (e.g., tournament) from the current population P^t, a set G is selected, the so-called set of "promising" solutions. Individual components of individuals belonging to the set G are grouped into independent subsets, the number of which is a heuristic parameter. A typical approach is to divide the range to which a given variable belongs into several disjoint sub-intervals. Then, it is determined how many individuals from set G have the observed variable in each of the subintervals. These values are used to determine the boundary probability of each group, and the next generation is generated according to the obtained distributions. Thanks to this, in the progeny generation, the appearance of individuals from the ranges in which the number of promising solutions is significant is greater.

This construction of the heuristic has one disadvantage: in continuous case it does not find optimal value exactly. It rather brackets the point at which optimum occurs. This can be dealt with by slightly modifying the ECGA. Every dozen or so iterations, we narrow the search only to the sub-interval for which the limit probabilities are the highest. This procedure causes that

ECGA, which in the original version copes well with exploration, also begins to exploit space well. Gradually reducing the set of feasible solutions significantly increases the chances of finding an exact solution. This is like a variable-adjacency local search or one of the versions of the ant algorithm.

The ECGA pseudocode in the version described above is as follows:

```
procedure ECGA:
1. Initialize metaparameters – size of population and set G, spacing density in each direction.
2. Randomly build starting population $P^0$.
3. Evaluate population $P^0$ according to fitness function.
4. Set $t = 0$.
5. do {
    a. Select the members of the promising set $G$.
    b. Determination of the limiting probabilities for each of the variables of individuals belonging to the set of $G$.
    c. Generation of a new individuals according to the probabilistic model in order to produce $O^t$.
    d. Evaluate population $O^t$ according to fitness function.
    e. Produce $P^{t+1}$ by applying succession operators to $O^t$ and $P^t$.
    f. $t++$.
    g. If necessary, reduce the set of admissible solutions to a subset containing a significant number of individuals from the set of $G$
    }
    while (stop condition).
6. Return the best fitting element in the last population.
```

3. Results

All algorithms were tested against four test functions:

1. Spheric function

$$f(x_1, x_2) = x_1^2 + x_2^2$$

2. Rastrigin function

$$f(x_1, x_2) = 20 + \sum_{i=1}^{2} (x_i^2 - 10\,cos(2\pi x_i))$$

3. Himmelblau function

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

4. Rosenbrock function

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - {x_1}^2)^2$$

The first of the test functions was used to assess the correct operation of own programs. The spherical function optimizes well and each method should find its minimum correctly. Subsequent functions were selected to check how well the tested heuristics cope with various difficulties in optimization. The Rastigin function is a typical multimodal function. Its tests are to show how resistant the method is to being left in local extremes. The Rosenbrock function reaches its minimum at one point lying in a very flat valley and is difficult to optimize due to the small differences between the values of the objective function. The Himmelblau function is a function having four equivalent local minima. Testing heuristics on it is to check whether the optimization method is able to "capture" the existence of many equivalent global minima.

The following assumptions were made in the tests performed for the purposes of this work:

- the size of the population and the maximum number of iterations were the same for each method and amounted to $N = 30$, $max_{Iter} = 500$.
- the halting criterion was no change (by more than 10^{-6}) in 20 consecutive generations.
- tournament selection and succession were used, preserving the best individual from the previous generation.
- each test included 50 independent starts.

Optimisation results with the use of each heuristic for selected test functions are shown in Table 1, Table 2 and Table 3. Vectors $\vec{x}_{opt}$, $\vec{x}_{best}$, $\vec{x}_{mean}$ are respectively the point where the analytical minimum occurs, the best result found in 50 runs, the average result over 50 runs. The tables also contain additional information about the relative error understood as

$$\delta_f = \frac{|f(\vec{x}_{opt}) - f(\vec{x}_{best})|}{1 + |f(\vec{x}_{opt})|} \cdot 100\%$$

and standard derivaions σ in 50 runs.

Table 1.
Results of Evolutionary Algorithm

Function	**Analytic minimum**	**EA**			
		$\vec{x}_{best} = (x_1, x_2)$	δ_f	$\vec{x}_{mean} = (\tilde{x}_1, \tilde{x}_2)$	σ
Spheric	$\vec{x}_{opt} = (0,0)$	$x_1 = 0.0008197$ $x_2 = -0.001555$		$\tilde{x}_1 = -0.00185$ $\tilde{x}_2 = -0.00176$	
	$f(\vec{x}_{opt}) = 0$	$f(x_{best}) = 0.00175$	$\approx 0.175\%$	$\tilde{f} \approx 0.00445$	0.0615
Rastrigin	$\vec{x}_{opt} = (0,0)$	$x_1 = 0.004234$ $x_2 = -0.00268$		$\tilde{x}_1 = 0.072767$ $\tilde{x}_2 = -0.013564$	
	$f(\vec{x}_{opt}) = 0$	$f(x_{best}) = 0.00498$	$\approx 0.498\%$	$\tilde{f} \approx 1.79551$	1.6114
Rosenbrock	$\vec{x}_{opt} = (1,1)$	$x_1 = 1.03732$ $x_2 = 1.06629$		$\tilde{x}_1 = 0.072767$ $\tilde{x}_2 = -0.013564$	
	$f(\vec{x}_{opt}) = 0$	$f(x_{best}) = 0.01087$	$\approx 1.087\%$	$\tilde{f} \approx 6.5147$	17.302

Cont. table 1.

Himmelblau	$\vec{x}_{opt} = \begin{cases} (3,2) \\ (-2.805, 3.131) \\ (-3.779, -3.283) \\ (3.584, -1.848) \end{cases}$	$x_1 = -3.77734$ $x_2 = -3.28137$		$\tilde{x}_1 = 0.072767$ $\tilde{x}_2 = -0.013564$	
	$f(\vec{x}_{opt}) = 0$	$f(\vec{x}_{best}) = 0.00027$	$\approx 0.027\%$	$\tilde{f} \approx 0.7248$	1.3547

Table 2.
Results of Extended Compact Genetic Algorithm

Function	**Analytic minimum**	**ECGA**			
		$\vec{x}_{best} = (x_1, x_2)$	δ_f	$\vec{x}_{mean} = (\tilde{x}_1, \tilde{x}_2)$	σ
Spheric	$\vec{x}_{opt} = (0,0)$	$x_1 = -2.68 \cdot 10^{-10}$ $x_2 = 4.26 \cdot 10^{-8}$		$\tilde{x}_1 = -0.00076$ $\tilde{x}_2 = -0.00204$	
	$f(\vec{x}_{opt}) = 0$	$f(x_{best}) = 4.3 \cdot 10^{-8}$	$\ll 0.001\%$	$\tilde{f} \approx 0.000983$	0.00375
Rastrigin	$\vec{x}_{opt} = (0,0)$	$x_1 = 6.06 \cdot 10^{-6}$ $x_2 = -3.87 \cdot 10^{-6}$		$\tilde{x}_1 = 0.080108$ $\tilde{x}_2 = -0.09905$	
	$f(\vec{x}_{opt}) = 0$	$f(x_{best}) = 1.03 \cdot 10^{-8}$	$\ll 0.001\%$	$\tilde{f} \approx 0.3138$	0.86817
Rosenbrock	$\vec{x}_{opt} = (1,1)$	$x_1 = 1.0002$ $x_2 = 1.0004$		$\tilde{x}_1 = 0.919493$ $\tilde{x}_2 = 0.884791$	
	$f(\vec{x}_{opt}) = 0$	$f(x_{best}) = 3.31 \cdot 10^{-7}$	$\ll 0.001\%$	$\tilde{f} \approx 0.4694$	0.0876
Himmelblau	$\vec{x}_{opt} = \begin{cases} (3,2) \\ (-2.805, 3.131) \\ (-3.779, -3.283) \\ (3.584, -1.848) \end{cases}$	$x_1 = -3.77932$ $x_2 = -3.283177$		$\tilde{x}_1 = -1{,}10304$ $\tilde{x}_2 = -1.63932$	
	$f(\vec{x}_{opt}) = 0$	$f(\vec{x}_{best}) = 1.1 \cdot 10^{-8}$	$\ll 0.001\%$	$\tilde{f} \approx 0.12695$	0.2396

Table 3.
Results of Population-based Incremental Learning

Function	**Analytic minimum**	**PBIL**			
		$\vec{x}_{best} = (x_1, x_2)$	δ_f	$\vec{x}_{mean} = (\tilde{x}_1, \tilde{x}_2)$	σ
Spheric	$\vec{x}_{opt} = (0,0)$	$x_1 = 1.14 \cdot 10^{-6}$ $x_2 = -2.26 \cdot 10^{-6}$		$\tilde{x}_1 = -0.000996$ $\tilde{x}_2 = 0.0007844$	
	$f(\vec{x}_{opt}) = 0$	$f(x_{best}) = 2.5 \cdot 10^{-6}$	$\ll 0.001\%$	$\tilde{f} \approx 0.003585$	0.00517
Rastrigin	$\vec{x}_{opt} = (0,0)$	$x_1 = -0.00308$ $x_2 = -0.00244$		$\tilde{x}_1 = 1.13311$ $\tilde{x}_2 = 1.11427$	
	$f(\vec{x}_{opt}) = 0$	$f(x_{best}) = 0.00306$	$\approx 0.306\%$	$\tilde{f} \approx 3.451923$	2.1875
Rosenbrock	$\vec{x}_{opt} = (1,1)$	$x_1 = 1.00031$ $x_2 = 1.00063$		$\tilde{x}_1 = 1.2217$ $\tilde{x}_2 = 1.54992$	
	$f(\vec{x}_{opt}) = 0$	$f(x_{best}) = 1.1 \cdot 10^{-7}$	$\ll 0.001\%$	$\tilde{f} \approx 0.10692$	0.1768
Himmelblau	$\vec{x}_{opt} = \begin{cases} (3,2) \\ (-2.805, 3.131) \\ (-3.779, -3.283) \\ (3.584, -1.848) \end{cases}$	$x_1 = 3.00007$ $x_2 = 1.99983$		$\tilde{x}_1 = 2.2454$ $\tilde{x}_2 = 1.7731$	
	$f(\vec{x}_{opt}) = 0$	$f(\vec{x}_{best}) = 4.5 \cdot 10^{-7}$	$\ll 0.001\%$	$\tilde{f} \approx 0.0027$	0.00491

The runs of the best optimizations are presented at Figure 1-Figure **4**.

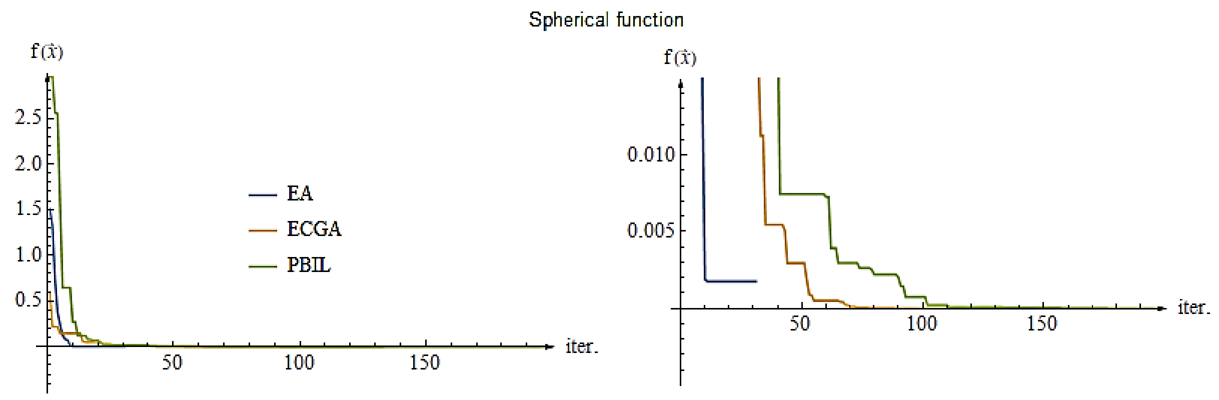

Figure 1. Optimisation runs using the heuristics tested - Spherical function.

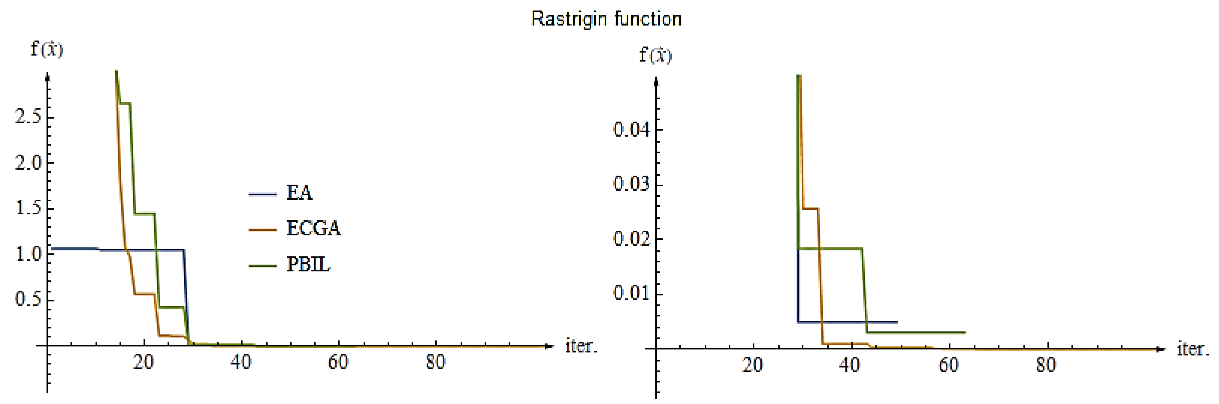

Figure 2. Optimisation runs using the heuristics tested - Ratrigin function.

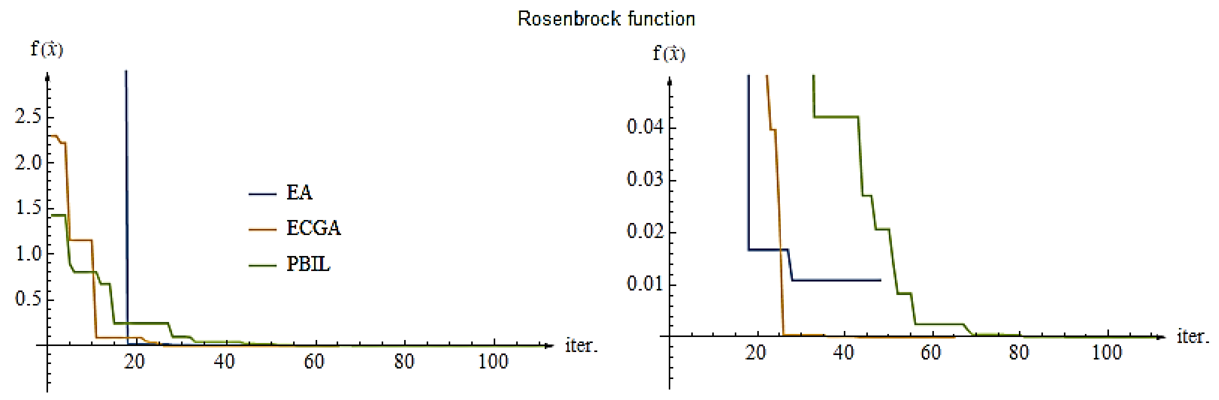

Figure 3. Optimisation runs using the heuristics tested - Rosenbrock function.

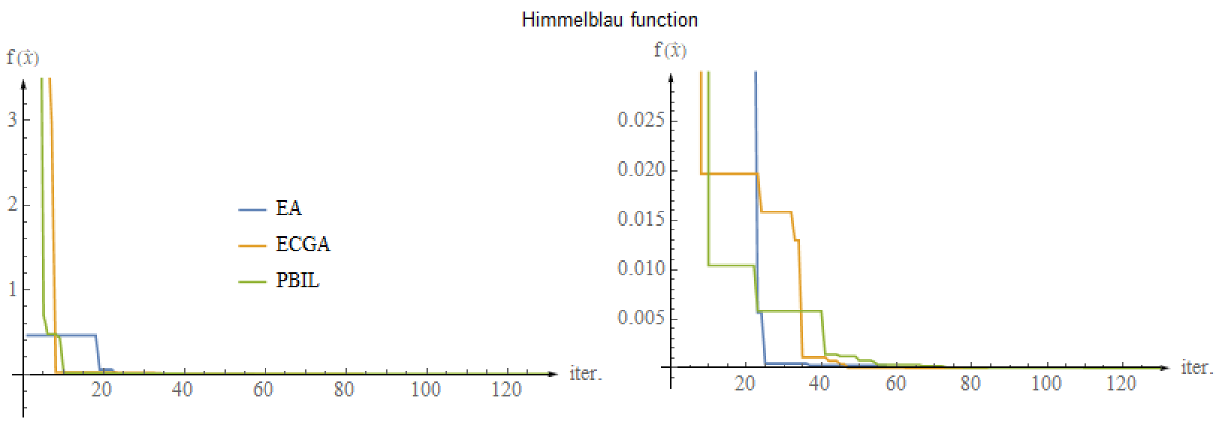

Figure 4. Optimisation runs using the heuristics tested - Himmelblau function.

The graphs show that in the EA, the stop condition intervened the fastest, but at the cost of a loss of accuracy. Both modifications achieve much better results than the original EA. This applies to both the accuracy and the level of standard deviation in 50 runs. Fig. 5 shows that this deviation is the largest for EA, which proves that the best result presented in Tab. 1 is not a typical result. EA is particularly bad at optimizing the Rosenbrock Valley. Although the best result in the test is satisfactory, it should be remembered that it was obtained in 50 runs. A very large standard deviation proves that in many optimizations the result was very inaccurate. We do not observe such an effect in any test of algorithms with probabilistic models.

When it comes to comparing ECGA and PBIL, the results indicate that the second method is worse at dealing with multimodal functions, although still better than the standard EA. PBIL also needs more iterations to get accurate results.

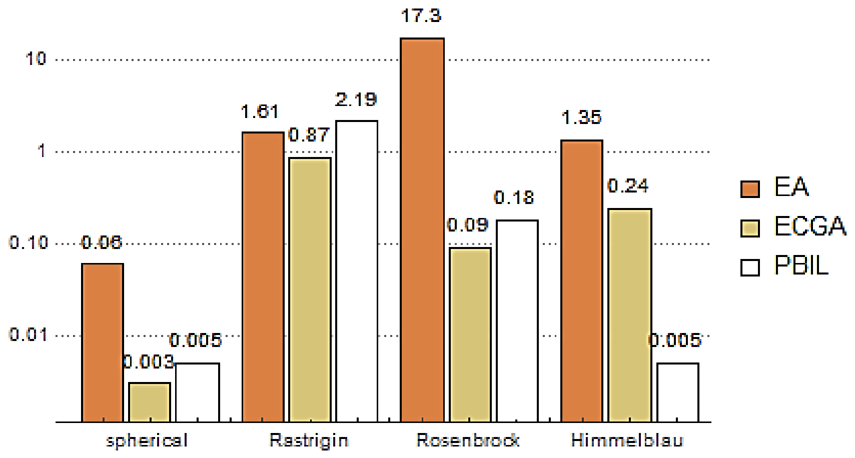

Figure 5. Standard deviation of the objective function values in 50 runs - logarithmic scale.

4. Discussion

The analysis of the obtained results shows that the replacement of genetic operators in the evolutionary algorithm by a probabilistic model built on the basis of promising solutions significantly increases the optimization accuracy. Both PBIL and ECGA determine function minima with much greater accuracy. The slower operation of PBIL and its lower efficiency in the case of multimodal functions result from a very simple procedure of building a probabilistic model. The expectation vector is modified based on one solution, which must affect the rate of optimization. Even such a simple model works better than the standard EA exchange of information based on crossing and mutation.

Of particular importance are the results of optimization of the Himmelblau function. It is a function with four equivalent minima. The population nature of the heuristics in a single run did not capture the existence of several extremes, because in subsequent generations, the population clustered around a single extreme. The existence of several points where the function reaches the same, minimum value was noticed by analyzing the results of several dozen launches. The random nature of all the analyzed heuristics meant that in different runs, extremes of similar value were found, but located at different points. Since the frequency of the points found was approximately the same, the obtained results had to be interpreted as indicating the existence of several equivalent minima.

5. Summary

An attempt to modify the classic Evolutionary Algotithm by replacing genetic operators with a probabilistic model of promising solutions seems to be a good solution. Such heuristics produce more accurate results and require fewer function evaluations to solve at the level found by EA. The use of a probabilistic model results in fewer low-value individuals in the progeny population. This may give a better chance of finding a solution using smaller populations.

The modifications used in the heuristics discussed in the article are not complicated either from the mathematical or programming side and still remain methods with a simple idea. There are also no requirements for the objective function. As in EA, the value of the evaluation function is sufficient for the correct operation of EA with a probabilistic model.

References

1. Chen, Y.P., Chen, C.H., (2010). Enabling the extended compact genetic algorithm for real-parameter optimization by using adaptive discretization. *Evol Comput. 2010 Summer*, pp. 199-228, doi: 10.1162/evco.2010.18.2.18202.
2. Duque, T., Goldberg, D., Sastry, K. (2008). *Improving the Efficiency of the Extended Compact Genetic Algorithm.* GECCO '08, pp. 467-468, doi: 10.1145/1389095.1389181.
3. Grisales-Noreña, L.F., Gonzalez Montoya, D., Ramos-Paja, C.A. (2018). Optimal Sizing and Location of Distributed Generators Based on PBIL and PSO Techniques. *Energies, 11*, 1018. https://doi.org/10.3390/en11041018.
4. Goldberg., D. (2006). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston: Addison-Wesley.
5. Kieszek, R., Kachel, S., Kozakiewicz, A. (2023). Modification of Genetic Algorithm Based on Extinction Events and Migration. *Applied Sciences, 13, 5584,* doi: 10.3390/app13095584.
6. Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin/Heidelberg: Springer, https://doi.org/10.1007/978-3-662-03315-9.

7. Rastegar, R., Hariri, A. (2006). The Population-Based Incremental Learning Algorithm converges to local optima. *Neurocomputing*, pp. 17772-1775, doi: 10.1016/j.neucom.2005.12.116 8.
8. Satman, M.H., Akadal, E. (2020). Machine Coded Compact Genetic Algorithms for Real Parameter Optimization Problems. *Alphanumeric Journal* , *8(1)* , 43-58 . DOI: 10.17093/alphanumeric.576919.
9. da Silva, M.H,, Legey, A.P., de A. Mól, A.C. (2018). The evolution of PBIL algorithm when used to solve the nuclear reload optimization problem. *Annals of Nuclear Energy, Vol. 113*, pp. 393-398, doi: https://doi.org/10.1016/j.anucene.2017.11.043.
10. Tamilselvi, S. (2022). *Introduction to Evolutionary Algorithms*. IntechOpen. doi: 10.5772/intechopen.104198.
11. Verma, A., Llorà, X., Venkataraman, S., Goldberg, D.E., Campbell, R.H. (2010). *Scaling eCGA model building via data-intensive computing*. IEEE Congress on Evolutionary Computation. Barcelona, Spain, pp. 1-8, doi: 10.1109/CEC.2010.5586468.
12. Vikhar, P.A. (2016). *Evolutionary algorithms: A critical review and its future prospects*. International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC). Jalgaon, India, pp. 261-265, doi: 10.1109/ICGTSPICC.2016.7955308.