GENERIC SCHEME OF A RESTART META-HEURISTIC OPERATOR FOR MULTI-OBJECTIVE GENETIC ALGORITHMS

Christina Brester, Ivan Ryzhikov, Olga Semenkina

Institute of Computer Science and Telecommunications,
Reshetnev Siberian State University of Science and Technology,
e-mails: christina.brester@gmail.com, ivan.s.ryzhikov@gmail.com,
semenkina.olga@mail.ru
Russia

Abstract: We introduce a generic scheme of restarting for multi-objective genetic algorithms and demonstrate that the use of a restart operator leads to a significant improvement in solution quality due to its tendency to explore different regions of a search space and get uniformly distributed points along a true front. The proposed scheme of restarting tells us when to restart an algorithm and how to generate a new population based on previously found solutions. This operator is algorithm-independent, which we show by incorporating it into three multi-objective genetic algorithms based on different heuristics.

Key words: restart meta-heuristic, genetic algorithm, multi-objective optimization, benchmark problems, performance improvement.

1. INTRODUCTION

Two main strategies known as exploration and exploitation push an evolutionary search to promising regions of the search space. Conventional genetic operators (selection, mutation, and crossover) provide an effective combination of these strategies. However, for some complex problems their capabilities seem to be limited and an algorithm (in particular, a genetic algorithm (GA)) is not able to find good solutions. Nevertheless, we may deal with this problem if we note that the search is stuck and the quality of solutions is no longer improving. A possible method is to interrupt the current run and launch the algorithm again taking into account the previously gathered information about the search space.

In this study, we introduce a restarting technique which is implemented as an additional restart operator [1, 2]. Until now, a number of restart meta-heuristics have been developed for one-criterion GAs [3, 4], whereas there are almost no studies devoted to the restart for multi-objective genetic algorithms (MOGAs). Therefore, we develop an algorithm-independent restart operator and investigate its effectiveness in combination with three different MOGAs. With our proposal, we answer two crucial questions: when to apply the restart and how to use the previously found solutions [5].

In the series of experiments, we demonstrate the efficiency of the restart and shed light on its potential capabilities. We also reveal that its performance is affected by the control parameter values, and the appropriate choice of these values may lead to a significant improvement in the solution quality.

The rest of the paper is organized as follows: firstly, we briefly present the MOGAs used and their basic features. Next, we describe our proposal in detail. Test problems and experimental conditions are also introduced. Then, experimental results are given and based on them some conclusions are presented.

2. MOGAS AND THEIR BASIC FEATURES

The common scheme of any conventional MOGA includes the same steps as the scheme of one-criterion GAs (Fig. 1a). However, optimizing several criteria simultaneously results in us obtaining not a single solution, but a set of non-dominated alternatives, i.e. a Pareto set approximation. A representation of this set in the criterion space is a Pareto front approximation. Usually, while solving multi-objective optimization problems, they tend to obtain solutions uniformly distributed along a true front. The found points should portray its shape in all regions of the criterion space.

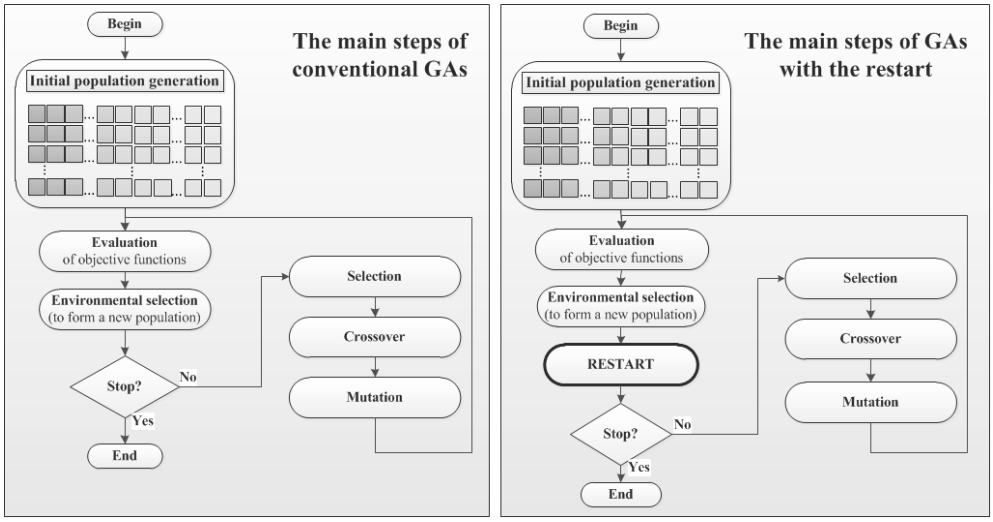

Fig. 1. (a) The common scheme of conventional GAs (left); (b) the scheme of GAs with restart operator.

However, owing to algorithm stagnation or the concentrating of the population in some particular areas, they may achieve a partly approximated Pareto front. Therefore, for that case, when the search is stuck, we suggest no longer performing such a "trapped" algorithm and spending resources in vain, but interrupting the current run and restarting it. This idea has been embodied as a restart meta-heuristic and implemented as an additional operator. Fig. 1b illustrates the modified GA scheme with the restart. In the next section, we present the restart meta-heuristic in detail, but here we would like to introduce the MOGAs used and their main features.

Generally, MOGAs differ from one another in three aspects: fitness assignment, diversity preservation and elitism. In this paper we consider MOGAs which are based on different heuristics: the Non-Sorting Genetic Algorithm II (NSGA-II) [6], the Preference-Inspired Co-Evolutionary Algorithm with goal vectors (PICEA-g) [7], and the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [8].

A while ago, the Pareto-dominance idea was recognised as the main principle of fitness assignment in any MOGA [9]: it proved its effectiveness in many benchmark and real problems and, as consequence, substituted other alternative proposals. Nevertheless, this principle might be implemented in different ways [10]: some algorithms use the dominance rank (the amount of individuals by which the candidate-solution is dominated), others apply the dominance count (the amount of points dominated by a certain individual), and the dominance depth might also be evaluated (a population is divided into several fronts (or niches) and the front which an individual belongs to is determined), and so on.

Then, to sustain variety within the Pareto set and front approximations, diversity preservation techniques are usually incorporated into MOGAs. Some methods use kernel functions to estimate the density of points; nearest neighbour techniques assess a distance between a given point and its k-th nearest neighbour; histograms might be engaged as density estimators (they use a hypergrid to calculate neighbourhoods) [11]. Commonly, distances between points are calculated in the criterion space.

Furthermore, due to the search stochasticity good solutions might be lost. To cope with this problem, elitism is usually applied. In general, there are two ways to implement it: the first way is to combine the parent population with the offspring and select the best individuals from the mating pool based on their fitness values. Another way is to use a special container which is called archive to copy there good solutions at each generation. In NSGA-II, the first described variant of elitism is used, and in SPEA2 the second one is applied, while both of them are employed in PICEA-g.

In Table 1, we give a brief description of the MOGAs used. It might be noted that each algorithm has its own peculiarities.

Table 1. Basic features of the MOGA used

MOGA	Fitness Assignment	Diversity Preservation
PICEA-g	Pareto-dominance (with generating goal vectors)	Nearest neighbor technique
NSGA-II	Pareto-dominance (niching mechanism) and diversity estimation (crowding distance)	Crowding distance
SPEA2	Pareto-dominance (niching mechanism) and density estimation (the distance to the k^{th} nearest neighbour in the objective space)	Nearest neighbour technique

Thus, in this study we investigate the effectiveness of the restart operator for various algorithms and show that it is algorithm-independent.

3. RESTART OPERATOR FOR MOGA

We consider a black-box multi-objective optimization problem:

$$C(a): A \to C_A \subset R^m, \dim(A) = n,$$
$$C(a) = \left(C_1(a) \quad \ldots \quad C_m(a)\right) \to \underset{a \in A}{extrem}, \tag{1}$$

where A is a space of alternatives with dimension n, C_A is a subspace of some Euclidean vector space R^m, $i=\overline{1,m}: C_i(\cdot): A \to C_A^i \subset R, \prod_j C_j(A) = C_A$ are the unknown mappings. We assume that there is a bijection between the alternatives and the binary strings, so every alternative would be represented in such a way.

Since we consider a population-based optimization tool, let the population at the i-th generation be noted as P_i. Each population consists of different solutions – a set of alternatives – and our aim is not to find the non-dominated set, but the set which maps into a good approximation of the whole Pareto front. In this case, a contradiction can be faced between the need for a search in depth to improve the current solutions and for a search in breadth to approximate the whole front.

To resolve this contradiction we implement the algorithm-independent restart operator. The following operator is applied when the predefined condition is met. We propose a restart condition that is based on the distance between the Pareto front estimations at two consecutive generations. If the distance does not change for some period, the algorithm restarts. A more detailed explanation is given below.

Let $S_i = \left\{ a_j \in A : \nexists k < i, j(k) \le |S_k| : a_j \overset{C}{\mathrm{p}} a_{j(k)}, a_{j(k)} \in S_k \right\}$ be the Pareto set and $F_i = \left\{ C(a_j), a_j \in S_i \right\}$ be the Pareto front estimations at the i-th generation. It is easy to see that $\forall i\ S_i \subset S_{i-1} \bigcup P_i$, so the distance $\rho(F_i, F_{i-1})$ between two different sets F_i and F_{i-1} is performed by the non-dominated solutions found at the current generation. Let F be a set of any limited cardinality $F = \left\{ f_i \in R^m, i = \overline{1,|F|} \right\}$, then

$\rho(F_a, F_b): F \times F \to R^+ \bigcup \{0\}$,

$$\rho(F_a, F_b) = \frac{1}{|F_a|} \cdot \sum_{i=1}^{|F_a|} \min_{j \le |F_b|} \left(\left\| (F_a)_i - (F_b)_j \right\|_{R^m} \right), \tag{2}$$

where $\|\cdot\|_{R^m}: R^m \to R^+ \bigcup \{0\}$ is a norm on the R^m vector space.

To estimate if there is a need for a restart we use a specific variable, which is a queue that consists of metric (2) values of the previous l_{tail} iterations,

$$Tail_i(l_{tail}) = \left\{ \rho(F_j, F_{j-1}) : i - l_{tail} < j \le i \right\}, \tag{3}$$

and the meta-heuristic performs a restart if the following condition is met

$$\max_{j < l_{tail}} \left\{ Tail_i(j) \right\} - \min_{j < l_{tail}} \left\{ Tail_i(j) \right\} \le \delta_{tail}. \tag{4}$$

In equations (3) and (4), the settings of two different operators are presented: the tail length l_{tail} controls the size of the observation period and δ_{tail} is the threshold level. Now, if the restart takes place, we save the current algorithm's run data into the sets, which is a representation of memory. Since all these features are significant for forming the final solution and performing the next algorithm run, the following sets are used:

$Memory_S = Memory_S \bigcup \{S_i\}$, $Memory_F = Memory_F \bigcup \{F_i\}$, $Memory_P = Memory_P \bigcup \{P_i\}$, and $Memory_C = Memory_C \bigcup \{\overline{C_i}\}$, where $\overline{C_i} = \left\{F\left(c_j\right) : c_j = \left(P_i\right)_j,\ j = \overline{1, |P_i|}\right\}$. Memory sets are also used to form the final approximation and evaluate the IGD metric.

First of all, let us describe a possible way in which the new starting population may perform. The generation of the initial population is controlled by two parameters: the probability of each individual in the initial population being randomly generated - α, and the probability of each gene being changed to the opposite - β, in the case of the individual being a mutant of a randomly chosen previously found solution. So, each j-th individual's k-th gene in the initial population is generated in one of the following ways:

$$\left(\left(P_0\right)_j\right)_k = r_{j,k},\ P\left(r_{j,k} = 0\right) = P\left(r_{j,k} = 1\right), \tag{5}$$

with probability α and with probability $1-\alpha$:

$$\left(\left(P_0\right)_j\right)_k = f_c\left(\left(\left(Memory_S\right)_{r_j^1}\right)_{r_j^2}, r_{j,k}^3\right), \tag{6}$$

where k is the number of gene, r_j^1, r_j^2, $r_{j,k}^3$ are the random values: $P\left(r_j^1 = 1\right) = ... = P\left(r_j^1 = \left|Memory_S\right|\right)$, $P\left(r_j^2 = 1\right) = ... = P\left(r_j^2 = \left(Memory_S\right)_{r_j^1}\right)$, $P\left(r_{j,k}^3 = 0\right) = 1 - P\left(r_{j,k}^3 = 1\right) = \beta$, and a special function $f_c\left(v, p\right) = \begin{cases} v, & p = 0 \\ \neg v, & p = 1 \end{cases}$.

Varying the parameters α and β, we control the initial population generation process. If we want the initial population to be completely randomized, we set α to 1, and if we want it to be in some sense near to the previously estimated Pareto set, we set it closer to 0 and β closer to 0 too, where β represents the closeness of a new individual to a found one.

4. EXPERIEMENTS AND RESULTS

A set of high-dimensional benchmark problems designed by the international scientific community to compare the efficiency of evolutionary algorithms (the CEC 2009 competition [12]) was chosen to investigate the performance of the original MOGAs and their modifications with the restart.

Test instances are different: Pareto sets and fronts might be discrete or continuous, convex or non-convex. In our experiments, we use a number of these benchmark problems which are unconstrained two- and three-objective optimization problems with real variables.

The metric IGD is applied to estimate the quality of the obtained Pareto Front approximations:

$$IGD\left(A, P^*\right) = \frac{\sum_{v \in P^*} d\left(v, A\right)}{\left|P^*\right|}, \tag{7}$$

where P^* is a set of uniformly distributed points along the Pareto Front (in the objective space), A is an approximate set to the Pareto Front, $d(v, A)$ is the minimum Euclidean

distance between v and the points in A. In short, the $IGD(A, P^*)$ value is the average distance from P^* to A. According to the rules of the CEC competition, the final approximation should contain no more than 100 and 150 points for two- and three-criterion problems respectively. The maximal number of vector-function evaluations is equal to 300, 000.

In all the experiments, the following setting were defined: binary tournament selection, uniform recombination and the mutation probability $p_m = 1/L$, where L is the length of the chromosome. We used standard binary coding to transfer binary strings to the real search space. All presented results were averaged over 25 runs of the algorithm.

Firstly, the original PICEA-g algorithm with no restart was run on the set of the CEC problems. Its performance was accepted as a baseline.

Next, we conducted a series of similar experiments for the PICEA-g with the restart operator in which we varied its control parameters: $\alpha = 0, 0.1, 0.3, 0.5, 0.9, 1$, $\beta = 0.05, 0.2, 0.5, 0.7$, $l_{tail} = 5, 10, 12, 15$, $\delta_{tail} = 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005$. In these experiments, our goal was to reveal the values of the control parameters which were the best on average for the set of CEC problems. We applied the linear normalization to all IGD values (including the results of the original PICEA-g) for each problem and found that $\alpha = 0.9$, $\beta = 0.7$, $l_{tail} = 10$, $\delta_{tail} = 0.001$, were the best settings on average. Fig. 2 demonstrates the results obtained. As can be seen, on most of the test problems the PICEA-g with the restart greatly outperforms the original one and only for Problems 4 and 8 does it give slightly worse IGD values.

In spite of the fact that the best settings on average provided us with rather good results, we found that for each problem the best settings were different and by using them we could obtain even better results.

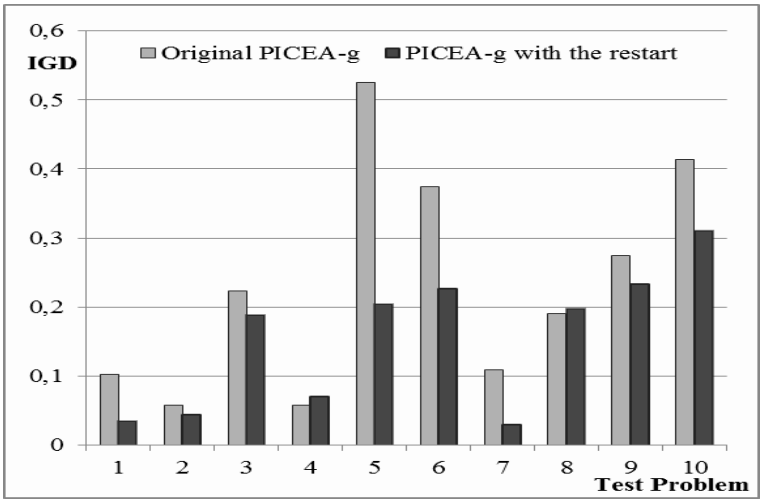

Fig. 2. Experimental results: the IGD values obtained by the original PICEA-g and the PICEA-g with the restart (the best settings on average).

Table 2 shows that for all the problems, the PICEA-g with the restart, whose control parameters have the best values for each problem, always outperforms the original one and for some problems, this improvement is significant. This implies that the control parameters should be tuned for the given problem carefully.

Table 2. Experimental results: the IGD values obtained by the original PICEA-g .

Test Problem	Original PICEA-g		
	Mean	Min	Max
UF1	0.1030	0.0743	0.1827
UF2	0.0582	0.0478	0.0815
UF3	0.2236	0.1664	0.3329
UF4	0.0576	0.0527	0.0679
UF5	0.5254	0.3720	0.7490
UF6	0.3743	0.2135	0.6468
UF7	0.1092	0.0399	0.4206
UF8	0.1902	0.1645	0.2010
UF9	0.2741	0.2102	0.3843
UF10	0.4142	0.2211	0.8590

Table 3. Experimental results: the IGD values obtained by the PICEA-g with the restart (the best settings for each problem).

Test Problem	PICEA-g, the restart with the best settings						
	Mean	Min	Max	α	β	l_{tail}	δ_{tail}
UF1	**0.0341**	**0.0243**	**0.0494**	0.9	0.7	5	0.0005
UF2	**0.0372**	**0.0312**	**0.0446**	0.1	0.7	15	0.0005
UF3	**0.1635**	**0.1272**	**0.2174**	0.1	0.7	10	0.00005
UF4	**0.0536**	**0.0484**	**0.0567**	0.9	0.5	5	0.01
UF5	**0.1597**	**0.1232**	**0.2145**	0.1	0.7	10	0.001
UF6	**0.1878**	**0.0616**	**0.3211**	0.9	0.5	5	0.0005
UF7	**0.0282**	**0.0240**	**0.0394**	0.5	0.05	15	0.0001
UF8	**0.1837**	**0.1331**	**0.1971**	0.3	0.5	5	0.001
UF9	**0.2037**	**0.1470**	**0.2856**	0.1	0.7	5	0.001
UF10	**0.2652**	**0.1968**	**0.4477**	0.9	0.5	5	0.001

Taking into consideration the best settings on average obtained for PICEA-g, we repeated the experiments for NSGA-II and SPEA2. In the beginning, we launched the original versions of NSGA-II and SPEA2. After that, we ran their modifications with the restart. To avoid numerous experiments with different α, β, l_{tail} , and δ_{tail} values, we set them equal to the best values on average which were found for PICEA-g.

It might be noted that for both algorithms, their modified versions are defeated by the original ones in three cases: Problems 3, 4 and 8 (and Problem 10 too for SPEA2). It means that we should adjust the restart control parameters not only for each problem (as was demonstrated for PICEA-g) but also for the MOGA used. We cannot extrapolate the results of one MOGA to another: settings which are good for one algorithm might be ineffective for another.

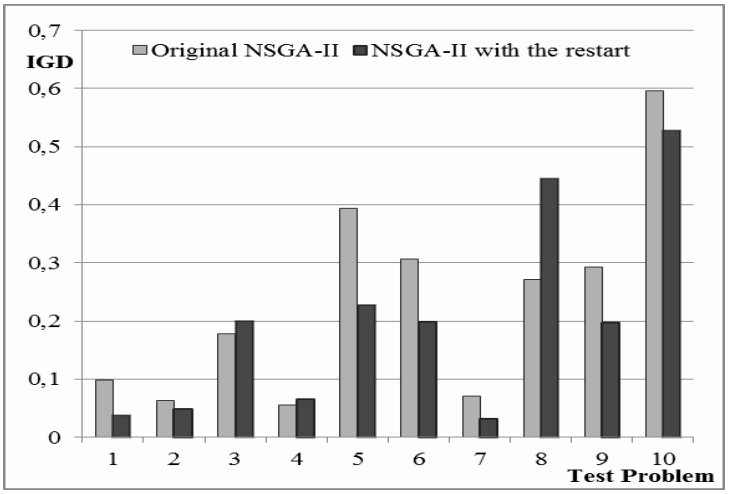

Fig. 3. Experimental results: the IGD values obtained by the original NSGA-II and the NSGA-II with the restart.

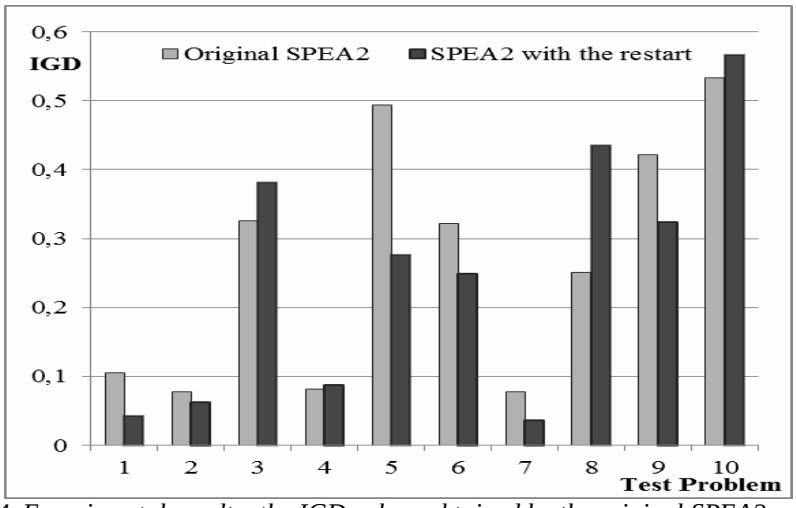

Fig. 4. Experimental results: the IGD values obtained by the original SPEA2 and the SPEA2 with the restart.

In any event, in such experiments, while varying the control parameter values, we may try just a finite set of their combinations. Therefore, we come to the conclusion that the restart operator should be adaptive: the values of its control parameters should be adjusted for the problem considered during the execution of the given MOGA [13, 14]. Only in this case will we be able to use the potential capabilities of the restart operator to their full extent.

5. CONCLUSION

We propose a restart meta-heuristic for MOGAs which is designed as an additional operator. This restart operator differs from conventional ones (selection, crossover, mutation)

because it is applied not at each generation but when a pre-defined criterion is satisfied. In our proposal, this criterion is based on the distance between Pareto front approximations found at consecutive generations. If this distance does not exceed a given threshold for the particular number of the generation, the restart is applied. It is essential that the new population is generated not from scratch but based on the non-dominated solutions previously found.

We investigate the effectiveness of our proposal in combination with three MOGAs which are based on different heuristics (PICEA-g, NSGA-II, SPEA2): the restart operator is incorporated into each of them. As can be noted, on the one hand, the use of the restart leads to a significant improvement in the algorithm performance (as a case in point, PICEA-g). However, on the other hand, it requires a subtle adjustment of the control parameters: the best settings are different not only for the given set of problems, they also cannot be extrapolated from one MOGA to another.

A possible way to cope with this problem is to conduct a series of experiments varying the values of the control parameters. However, it is time-consuming and the values chosen for testing are always limited.

Therefore, to avoid numerous experiments every time we face a new problem or change a MOGA, we should develop an adaptive restart operator whose control parameters are adjusted automatically during the execution of the algorithm. This adaptation may help us to take advantage of all the useful capabilities of the restart.

ACKNOWLEDGEMENTS

The reported study was funded by Russian Foundation for Basic Research, Government of Krasnoyarsk Territory, Krasnoyarsk Region Science and Technology Support Fund to the research project № 16-41-243036.

REFERENCES

[1] Ryzhikov I., Semenkin E. Restart Operator Meta-heuristics for a Problem-Oriented Evolutionary Strategies Algorithm in Inverse Mathematical MISO Modelling Problem Solving. *IOP Conference Series: Materials Science and Engineering*, vol. 173. 2017. DOI: 10.1088/1757-899X/173/1/012015

[2] Ryzhikov I., Semenkin E., Sopov E. A Meta-heuristic for Improving the Performance of an Evolutionary Optimization Algorithm Applied to the Dynamic System Identification Problem. *IJCCI (ECTA)*, 2016, pp. 178–185.

[3] Fukunaga A.S. Restart scheduling for genetic algorithms. In: Eiben A.E., Bäck T., Schoenauer M., Schwefel HP. (eds) Parallel Problem Solving from Nature – PPSN V. PPSN 1998. *Lecture Notes in Computer Science*, vol. 1498. Springer, Berlin, Heidelberg. 1998.

[4] Beligiannis G.N., Tsirogiannis G.A. and Pintelas P.E. Restartings: a technique to improve classic genetic algorithms' performance. *International Journal of Computational Intelligence*, vol. 1, 2004, pp. 112–115.

[5] Brester Ch., Ryzhikov I., Semenkin E. On Performance Improvement Based on Restart Meta-Heuristic Implementation for Solving Multi-objective Optimization Problems. *ICSI* (2), 2017, pp. 23-30.

[6] Deb K., Pratap A., Agarwal S., Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6 (2), 2002, pp. 182-197.

[7] Wang R. *Preference-Inspired Co-evolutionary Algorithms*. A thesis submitted in partial fulfillment for the degree of the Doctor of Philosophy. University of Sheffield, 2013, 231 pp.

[8] Zitzler E., Laumanns M., Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. *Evolutionary Methods for Design Optimisation and Control with Application to Industrial Problems EUROGEN 2001*, 3242 (103), 2002, pp. 95-100.

[9] Goldberg D. *Genetic algorithms in search. optimization. and machine learning*. Addison-wesley. 1989.

[10] Zitzler E., Laumanns M., Bleuler S. A Tutorial on Evolutionary Multiobjective Optimization. In: Gandibleux X.. (eds.): *Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems*, vol. 535. 2004.

[11] Silverman B. *Density estimation for statistics and data analysis*. Chapman and Hall. London. 1986.

[12] Zhang Q., Zhou A., Zhao S., Suganthan P.N., Liu W., Tiwari S. *Multi-objective optimization test instances for the CEC 2009 special session and competition*. University of Essex and Nanyang Technological University. Tech. Rep. CES-487. 2008.

[13] Guliashki V., Kirilov L., Genova K. An interactive evolutionary algorithm for multiple objective integer problems. *International Journal on Information Technologies & Security*, vol. 5, 2013, pp. 45-54.

[14] Kasabov, N. From Multilayer Perceptrons and Neuro-Fuzzy Systems to Deep Learning Machines: Which Method to Use? – A Survey. *International Journal on Information Technologies and Security*, No. 2 (vol. 9), 2017, pp. 3-24.

Information about the authors:

Christina Brester is a docent at the Department of Higher Mathematics of the Reshetnev Siberian State University of Science and Technology (Krasnoyarsk, Russia). She received her PhD in Computer Science in 2016 from the Siberian Federal University and the Institute of Computational Modeling of Siberian Branch of Russian Academy of Sciences. Her research interests include evolutionary computation, neuro-evolutionary algorithms, machine learning and speech analysis.

Ivan Ryzhikov is a research fellow at the Reshetnev Siberian State University of Science and Technology (Krasnoyarsk, Russia). He received his PhD in Computer Science in 2016 from the Reshetnev Siberian State Aerospace University. His areas of research include global black-box optimization, meta-heuristics for natural computing extremum seeking algorithms, inverse mathematical modeling, dynamical system identification.

Olga Semenkina is a professor at the Department of Higher Mathematics of the Reshetnev Siberian State University of Science and Technology (Krasnoyarsk, Russia). Doctor of technical sciences, professor. She received her PhD in Computer Science in 1995 and, then, her DSc in Engineering and Habilitation in 2002 from the Reshetnev Siberian State Aerospace University (Krasnoyarsk, Russia). Her scientific interests are modelling and optimization of complex systems, discrete optimization, data analysis.

Manuscript (Final version) is received on 04 April 2018