RESTART OPERATOR FOR MULTI-OBJECTIVE GENETIC ALGORITHMS: IMPLEMENTATION, CHOICE OF CONTROL PARAMETERS AND WAYS OF IMPROVEMENT

Christina Brester[1], Ivan Ryzhikov[2], Eugene Semenkin[3]

Institute of Computer Science and Telecommunications,
Reshetnev Siberian State University of Science and Technologies,
e-mails: [1]christina.brester@gmail.com, [2]ryzhikov-88@yandex.ru,
[3]eugenesemenkin@yandex.ru
Russia

Abstract: In this article, we introduce a generic scheme for a restart meta-heuristic for multi-objective genetic algorithms, demonstrate its effectiveness on a set of benchmark problems, and discuss how restart control parameters affect algorithm performance. Based on the experimental results, we come to a conclusion about the peculiarities of the restarting scheme used and propose ways of its improvement.

Key words: Multi-objective optimization, evolutionary algorithm, restart meta-heuristic.

1. INTRODUCTION

An evolutionary search, in particular Genetic Algorithms (GA), might encounter some obstacles, which decrease its performance tremendously. Stagnation and premature convergence to a local optimum are unlikely to be overcome by just applying conventional genetic operators. The only way is to interrupt the current run and start the algorithm again using the gathered information about the search space: exploit previously found good solutions to make them more accurate, penalize attraction basins of local optima and explore new areas.

The described idea has been elaborated so that a restarting technique has been proposed [1, 2] and implemented as an additional restart operator [3, 4]. A number of studies have been devoted to the use of restarting in one-criterion optimization, yet there are almost no examples describing multi-objective heuristics [5, 6] with this additional operator which might be useful in designing a representative approximation of a Pareto front. Therefore, in this article we propose a generic scheme for the restart operator for multi-objective GAs. In our proposal we answer two questions, specifically, when to restart the algorithm and how to generate a new population based on the previously found good solutions.

As a case in point, we incorporate the restart operator in a Preference-Inspired Co-Evolutionary Algorithm using goal vectors (PICEA-g) [7]. This algorithm has demonstrated its high efficiency on a range of benchmark [8] and real [9, 10] multi-objective optimization problems and, therefore, we have chosen it to show that even its high performance might be improved with the use of a restart. In the experiments conducted, we run algorithms on the set of high-dimensional two- and three-objective unconstrained optimization problems from the CEC-2009 competition. Based on the experimental results, we can make an inference about a considerable improvement in performance. We estimate a special metric reflecting the distance between a true Pareto front and its approximations, firstly, obtained by the original PICEA-g and then by its modified version with a restart. After that, we perform a deeper analysis of the "behavioural peculiarities" of the algorithm caused by varying the values of the restart control parameters. This analysis helps us to see the bigger picture and find ways to improve the proposed meta-heuristic.

The rest of the paper is organized as follows: related work is described briefly in Section 2. The scheme for PICEA-g and the details of the restart meta-heuristic are given in Section 3. Section 4 contains the experimental results, their analysis and discussion. The conclusion and future work are presented in Section 5.

2. RELATED WORK

Generally, the idea of restarting is applicable for various iterative methods and algorithms, not only for evolutionary ones. For instance, in Arnoldi's method for finding eigenvalues a similar technique is used [11]. While restarting, it tends to preserve the useful information found about Krylov subspaces on the previous iterations and generate new starting vectors for the next steps based on it [2]. In [1] the authors distinguish between a static and dynamic restart. The static strategy means that a restart schedule is defined in advance, before the algorithm execution (for example, at every *t*-th generation). In contrast, in the dynamic strategy some metrics describing the algorithm progress or its convergence might be introduced to terminate the current run when a predefined threshold is achieved.

In [1] an approach for finding the static restart schedule of a GA for the case of resource-bounded optimization is proposed. Researchers use the algorithm performance on a set of similar problems to evaluate the restart schedule for the current one.

It has been shown [2] that in comparison with the conventional GA, the GA with the restart operator requires significantly fewer fitness function evaluations to find an optimum for a number of test functions (Rosenbrock's valley, the Step function, the Himmelblau function, and the Sphere function). In this study, the restart operator is applied at each *t*-th generation (different values of *t* have been tested). Then, an adaptive restarting GA, presented in [12] and using the dynamic restart scenario, outperforms not only the conventional GA but also a number of other optimization algorithms, such as Spiral Dynamic Algorithm (SDA) and Bacterial Foraging Algorithm (BFA), on benchmark functions. In this algorithm, the restart occurs if the best solution obtained is not improved within *i* generations.

The successful use of the restart idea for GAs inspired researchers to incorporate it into Differential Evolution (DE). In [13] the restart mechanism is implemented as a Random Mutation (originally, DE did not use mutation at all). This modified DE has demonstrated its effectiveness on a set of 10-, 30- and 50-dimensional test problems and shown good properties compared with six other self-adaptive DE algorithms.

However, there are not many applications for the restart operator in MOGAs. In [14] authors propose a modified SPEA2 which is used to generate a fuzzy rule base. In their algorithm, the restart operator is problem-specific and intended for tuning linguistic fuzzy systems.

The next section contains a description of our proposal in detail. Firstly, we shed light on the main steps of the PICEA-g algorithm and then introduce the restart meta-heuristic for multi-objective GAs.

3. PICEA-G AND RESTART META-HEURISTIC FOR MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS

3.1. Preference-Inspired Co-Evolutionary Algorithm with goal vectors

The concept of co-evolving the population with decision-maker preferences underlies a class of preference-inspired co-evolutionary algorithms (PICEAs). However, we should not consider "decision-maker preferences" literally as real ones. In this case, they are just a means to compare solutions in the population, in other words, they are co-evolving references which progress as a search is accomplished.

One of the algorithms from this class was presented by Wang in 2013 and called a Preference-Inspired Co-Evolutionary Algorithm using goal vectors (PICEA-g) [7]. Goal vectors serve as auxiliary points generated in the criterion space within the particular bounds at each generation: the more goal vectors a solution dominates, the higher its fitness. In addition, the fitness of a goal vector is determined by the number of solutions which dominate it (the more, the worse) and more effective vectors are chosen. The goal vector bounds depend on the criterion values of non-dominated points found by the current moment and change as this set evolves. Thus, the aim of goal vectors is to direct the population towards the Pareto front co-evolving with it.

PICEA-g includes the following steps:

1. Generate an initial population with N individuals and evaluate the criterion values of each candidate solution. Find non-dominated points in the population, copy them into the archive and estimate the goal vector bounds. Initialize the set of goal vectors as N_g targets randomly generated within the goal vector bounds.

2. Produce N offspring solutions with genetic operators: selection, crossover and mutation. Evaluate criterion values for newly generated candidates.

3. Pool together parents and children (the cardinality of the united population is equal to $2N$).

4. Add to the set of goal vectors N_g new targets randomly generated within the determined bounds (the cardinality of the set of goal vectors is equal to $2N_g$).

5. Assign fitness values to individuals in the united population based on the number of goal vectors they dominate taking into account that candidates which dominate the same goal vector share this fitness contribution.

6. Assign fitness values to goal vectors based on the number of candidate solutions they are dominated.

7. Form the new population by ranging candidates based on their fitness values and selecting the best N solutions. Do the same for the set of goal vectors and so truncate its size by down to N_g.

8. Update the archive with new non-dominated solutions. Update the goal vector bounds.

9. Check the stopping criterion: if it is satisfied, then finish the search with the archive set, otherwise proceed with the second step.

Thus, in Steps 1 and 4 decision-maker preferences are incorporated into the algorithm by using goal vectors. More details might be found in [7].

3.2. Restarting Operator Meta-Heuristic

In this study, we consider a black-box multi-objective optimization problem:

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

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 are considering 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, one can face a contradiction 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 $S_i = \left\{a_j \in A : \nexists k < i, j(k) \le |S_k| : a_j \overset{C}{\prec} a_{j(k)}, a_{j(k)} \in S_k\right\}$ 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 in 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 the 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

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

In equations (3) and (4) two control parameters 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 are 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 $Memory_C = Memory_C \bigcup \{\tilde{C}_i\}$, where $\tilde{C}_i = \left\{F\left(c_j\right): c_j = \left(P_i\right)_j,\ j=\overline{1,|P_i|}\right\}$.

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: $P\left(r_j^1=1\right)=\ldots=P\left(r_j^1=|Memory_S|\right)$, $P\left(r_j^2=1\right)=\ldots=P\left(r_j^2=\left(Memory_S\right)_{r_j^1}\right)$, $P\left(r_{j,k}^3=0\right)=1-P\left(r_{j,k}^3=1\right)=\beta$, 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 previous Pareto set estimations, we set it closer to 0 and β closer to 0 too, where β represents the closeness of the new individual to one already found.

Memory sets are also used to perform the final IGD metric (7) and estimate the algorithm efficiency.

4. PERFORMANCE ASSESSMENT

4.1. Test Multi-Objective Problems

In our experiments, we investigate the efficiency of the original PICEA-g and its modification with the restart on a set of high-dimensional benchmark problems proposed by the international scientific community to compare the performance of developed evolutionary algorithms (the CEC 2009 competition [15]).

There are problems with discrete and continuous, convex and non-convex Pareto Sets and Fronts. In this article, we use a number of these test instances which are unconstrained two- and three-objective optimization problems with real variables.

According to the rules of the CEC competition, the metric IGD is applied to estimate the quality of 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 and $d\left(v,A\right)$ is the minimum Euclidean distance between v and the points in A. In short, the $IGD\left(A,P^*\right)$ value is the average distance from P^* to A.

In the next section, we describe the experiments conducted and their conditions, discuss the results obtained and analyse the influence of control restart parameters on the algorithm performance.

4.2. Experimental Results

In all the experiments conducted, we followed the rules of the CEC 2009 competition: the maximal number of function evaluations was equal to 300,000; the maximal number of points in the approximation obtained by any algorithm for the estimation of the IGD metric was 100 and 150 for two-criterion and three-criterion problems respectively.

Firstly, the conventional PICEA-g algorithm with no restart was applied to the set of benchmark problems introduced. For all of the test instances, IGD values were averaged over 25 runs of the algorithm.

We also defined the following settings: binary tournament selection, uniform recombination and the mutation probability $p_m = {}^{1}\!/_{L}$, where L is the length of the chromosome. PICEA-g operated with binary strings and, therefore, we used standard binary coding for the transfer to the real search space.

Then we conducted a series of similar experiments for PICEA-g with the restart operator in which we varied its control parameters:

$$\alpha = 0, 0.1, 0.3, 0.5, 0.9, 1;\ \beta = 0.05, 0.2, 0.5, 0.7;\ l_{tail} = 5, 10, 12, 15;$$

$$\delta_{tail} = 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005.$$

We tended to figure out their best combination on average for the set of benchmark problems and also plunge ourselves into the deeper analysis of the algorithm performance depending on the restart control parameters. We applied the linear normalization to all IGD values (including the results of the conventional PICEA-g) for each problem and found that $\alpha = 0.9$, $\beta = 0.7$, $l_{tail} = 10$, $\delta_{tail} = 0.001$ are the best settings on average. Moreover, we ascertained the best values of the restart control parameters for each problem and noticed that they were different for various problems (Table 1).

Firstly, we would like to discuss the averaged results in relation to the α and β parameters, which control the new population generation process in restarting. In Figure 1 we depict boxplots for diverse combinations of α and β, and corresponding IGD values averaged over all the problems.

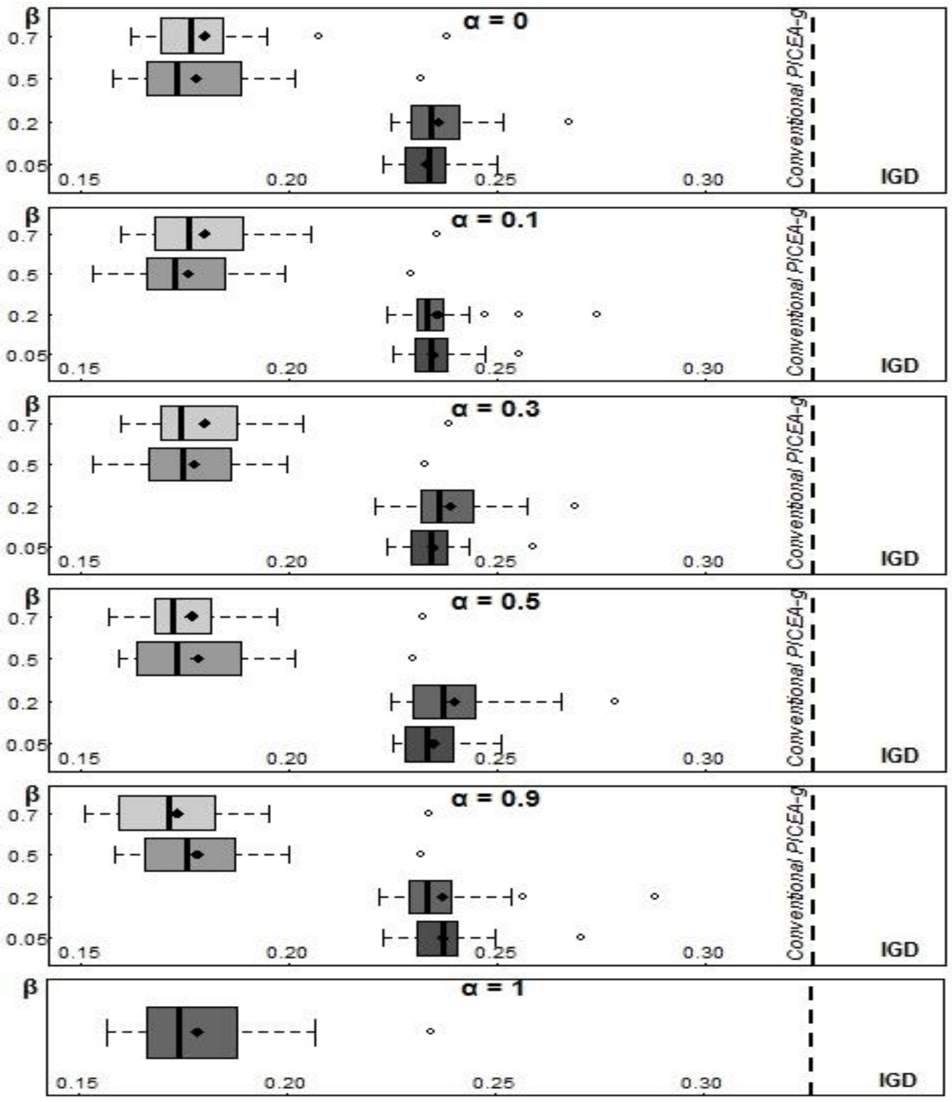

Fig. 1. The influence of α and β control parameters on the algorithm performance.

Each boxplot reflects the experimental results obtained for different combinations of l_{tail} and δ_{tail} and so demonstrates the possible range of the averaged IGD values. As usual in the boxplot, we can see medians, upper and lower quartiles, upper and lower extremes, outliers, and with diamond-shaped marks we depict mean values. It might be noted that for different levels of α the decrease of β leads to a deterioration in the algorithm performance. Why do we get this distribution? And why in the case of the completely randomized new population ($\alpha = 1$) do we have the better result in comparison with IGD values obtained with low levels of α and β (the new population is generated with the use of the previously found non-dominated points)?

Table 1. The best combination of settings for each test problem.

Test Problem	α	β	l_{tail}	δ_{tail}
UF1	0.9	0.5	5	0.001
UF2	0.9	0.7	5	0.0005
UF3	0.1	0.7	15	0.0005
UF4	0.1	0.7	10	0.00005
UF5	0.9	0.5	5	0.01
UF6	0.1	0.7	10	0.001
UF7	0.9	0.5	5	0.0005
UF8	0.5	0.05	15	0.0001
UF9	0.3	0.5	5	0.001
UF10	0.1	0.7	5	0.001

The tricky thing here is that when we generate the new population based on the previously found non-dominated solutions with a low level of mutation probability ($\beta = 0.05$) we get rather effective candidate solutions. This implies that our values of l_{tail} and δ_{tail} become inappropriate for the restart criterion because the search could not manage to improve the approximation for l_{tail} generations so that the distance $\rho(F_a, F_b)$ would be more than δ_{tail}. When we increase the mutation probability a little ($\beta = 0.2$), we have a similar situation because there is a huge gap between the solutions which are similar to the previously found ones and the solutions which have undergone the mutation. Therefore, again for l_{tail} generations the population could not evolve to produce new non-dominated points so the restart criterion could not be satisfied.

On the one hand, at the current stage the increase of β helps to solve this problem. However, on the other hand, it means that we almost do not use the previously found solutions and just continue to explore the search space randomly. Therefore, we propose using two strategies in the restart. The first one is to generate the new highly randomized population with the use of the same restart criterion. The second one is to generate the new candidates which are close to the previously found ones and try to improve them but using a less strict restart criterion to give the population a chance to evolve. The balanced combination of these strategies and switching between them should help us to improve the current proposal.

In Figure 2 we illustrate the influence of δ_{tail} and l_{tail} control parameters on the algorithm performance for $\alpha = 0.9$, $\beta = 0.7$ (the best settings on average). We may note that for different values of l_{tail} the best values of δ_{tail} vary slightly. While decreasing l_{tail} to 5, we find the best δ_{tail} equalled 0.0005. For $l_{tail} = 10$, $l_{tail} = 12$, $l_{tail} = 15$ the best value of

δ_{tail} is 0.001. The further increase of δ_{tail} leads to the considerable worsening of IGD, especially for the low level of l_{tail}. Also, excessively small values of δ_{tail} lead to the deterioration in IGD. However, using the best settings on average we do not achieve the highest effectiveness because the best settings differ for each problem.

In Table 1 we may even note that for the UF4 and UF8 problems, settings which are quite bad on average are the best for these particular problems: $l_{tail} = 10$, $\delta_{tail} = 0.00005$ are the best for the UF4 problem and $\alpha = 0.5$, $\beta = 0.05$ are the best for the UF8 problem. This implies that we should perform the careful choice of the control parameter values because they greatly affect the algorithm performance. Moreover, the right strategy is to adjust these parameters for the particular problem or, in other words, to make them adaptive. In that case, we will be able to use this adaptive algorithm as an auxiliary optimization tool in different areas (for example, in machine learning [16] to train complex models taking into account several criteria).

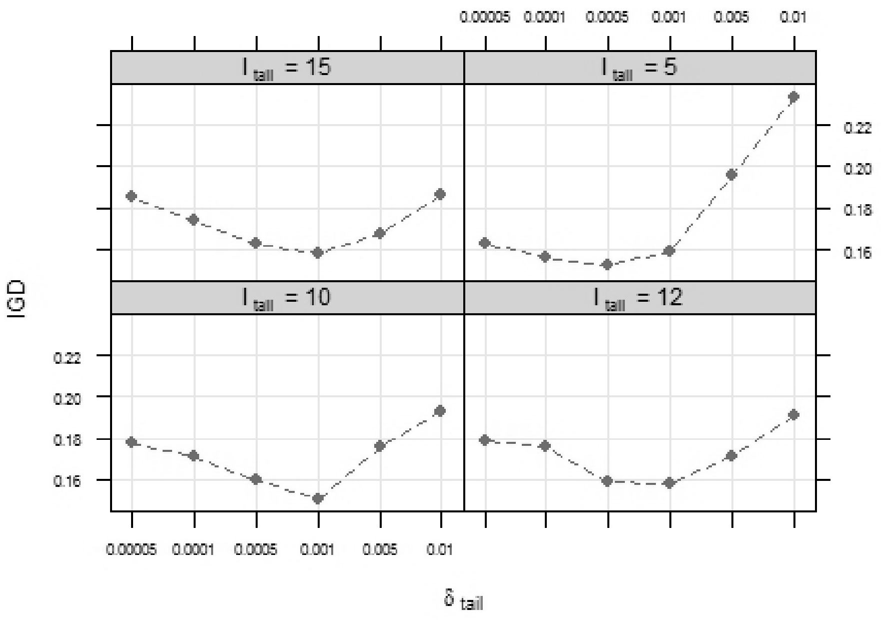

Fig. 2. The influence of δ_{tail} and l_{tail} control parameters on the algorithm performance.

In Figure 3 we demonstrate the difference between the performance of the original PICEA-g (green bars) and the PICEA-g with the restart (blue and pink bars) for each problem (with no averaging over the problems). Here we show two IGD values obtained by the PICEA-g with the restart: blue bars correspond to the algorithm with the best settings on average and pink bars correspond to the algorithm with the best settings for each problem (the best possible combination in our experiments). In this figure, in addition to the mean values, there are minimum and maximum values of IGD for each problem. As we can see, the PICEA-g with the restart almost always outperforms the conventional PICEA-g. Only for two problems (UF4 and UF8) the PICEA-g with the restart (the best settings on average) is defeated by the original algorithm. Nevertheless, in general, we may conclude that the restarting allows the algorithm performance to be increased significantly.

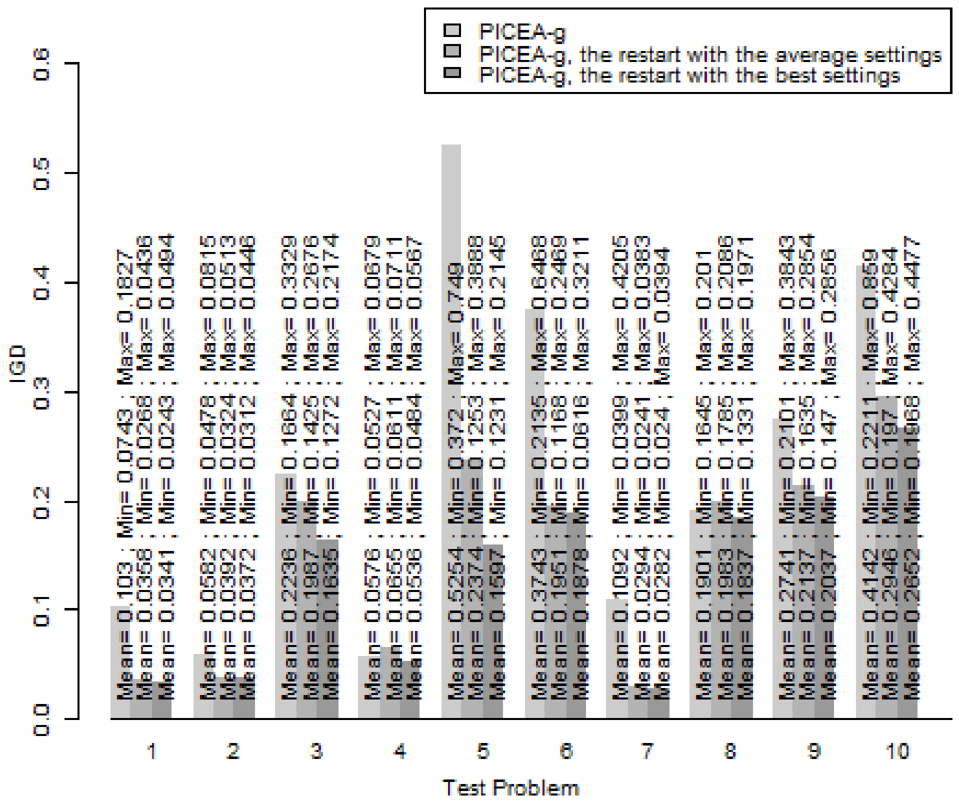

Fig. 3. Experimental results. Comparison of the original PICEA-g and the PICEA-g with the restart.

5. CONCLUSION

The restarting idea is a simple but effective technique to improve the performance of GAs. In this article, we propose a generic scheme of its implementation by means of an additional restart operator.

The presented meta-heuristic is designed for multi-objective GAs and we investigate its effectiveness using the example of the PICEA-g algorithm. According to the experimental results, the PICEA-g with the restart operator greatly outperforms the original one on the set of high-dimensional benchmark problems.

However, this operator requires subtle tuning of its control parameters. Therefore, in this study, we fulfil a deep analysis of the algorithm performance and its peculiarities depending on the values of the restart control parameters. This analysis reveals the ways of how we may improve the current proposal. In particular, it is reasonable to elaborate algorithmic schemes to make the restart adaptive: its control parameters should be tuned for the problem automatically during the algorithm execution. Moreover, the restart operator should provide us with an option not only to explore unknown regions of the search space with new randomly generated candidate solutions but also to exploit previously discovered areas more carefully.

ACKNOWLEDGEMENTS

This research is performed with the financial support of the Ministry of Education and Science of the Russian Federation within state project assignment № 2.1680.2017/ПЧ.

REFERENCES

[1] 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, pp. 257-366.

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

[3] 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.

[4] 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.

[5] Janssens, G.K. Recent challenges in the use of evolutionary algorithms for multi-objective optimization. *International Journal on Information Technologies & Security*, (vol. 1), 2009, pp. 3-12.

[6] 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.

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

[8] Brester, C., Semenkin, E. Cooperative Multi-Objective Genetic Algorithm with Parallel Implementation, *Advances in Swarm and Computational Intelligence*, Part I, LNCS 9140, 2015, pp. 471–478.

[9] Brester, C., Semenkin, E., Sidorov, M., Minker, W. Self-adaptive multi-objective genetic algorithms for feature selection, *Proceedings of International Conference on Engineering and Applied Sciences Optimization*, Kos Island, Greece, 2014, pp. 1838–1846.

[10] Brester, C., Semenkin, E., Sidorov, M., Kovalev, I., Zelenkov, P. Evolutionary feature selection for emotion recognition in multilingual speech analysis, *Proceedings of IEEE Congress on Evolutionary Computation* (CEC2015), Sendai, Japan, 2015, pp. 2406–2411.

[11] Petition, S.G. Parallel subspace method for non-Hermitian eigenproblems on the connection machine (CM2), *Applied Numerical Mathematics*, (vol.10), 1992, pp. 19–36.

[12] Dao, S.D., Abhary, K., Mariam, R. An adaptive restarting genetic algorithm for global optimization, *Proceedings of the World Congress on Engineering and Computer Science*, WCES 2015, Oct 21-23, San Francisco, USA, 2015.

[13] Mohamed, A.W. RDEL: restart differential evolution algorithm with local search mutation for global numerical optimization, *Egyptian Informatics Journal*, (vol. 15), no. 3, 2014, pp. 175–188.

[14] Gacto, M.J., Alcala, R., Herrera F. An Improved Multi-Objective Genetic Algorithm for Tuning Linguistic Fuzzy System, *Proc. of 2008 International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (IPMU 2008), 2008, pp. 1121–1128.

[15] 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.

[16] 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*, ISSN 1313-8251, No. 2 (vol. 9), 2017, pp. 3-24.

Information about the authors:

Christina Brester is an Associate Professor at the Department of Higher Mathematics, Reshetnev Siberian State University of Science and Technologies (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 Technologies (Kranoyarsk, 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, and dynamical system identification.

Eugene Semenkin is a Professor at the Department of System Analysis and Operations Research, Siberian State University of Science and Technologies. He received his PhD in Computer Science from Leningrad State University (Leningrad, USSR) in 1989 and his Dr. Sc. in Engineering and Habilitation from the Siberian State Aerospace University (Krasnoyarsk, Russia) in 1997. His areas of research include the modelling and optimization of complex systems, computational intelligence and data mining.

Manuscript received on 07 October 2017