

ITMS'2018



The 7th International Scientific Conference Changes in Social and Business Environment (CISABE'2018), 26-27 April 2018, Panevėžys, Lithuania

Hyper-heuristics for structure and parameters tuning in evolutionary algorithms

Pupkov A.^a, Sopov E.^{a,b}, Panfilov I.^{a,b}, Samarin V.^a, Telesheva N.^a, Kuzmich R.^{a,*}

^a *Siberian Federal University, 79 Svobodny pr., Krasnoyarsk 660041, Russia*

^b *Siberian State University of Science and Technologies, Krasnoyarsky Rabochy Av., Krasnoyarsk 660037, Russia*

Abstract

Evolutionary algorithms have proved their efficiency solving many complex optimization problems. Evolutionary algorithms are metaheuristics that provide search in a space of problem solutions. The performance of evolutionary algorithms depends on their structure and parameters, which should be fine-tuned for each optimization problem. In this paper, we discuss hyper-heuristic methods that provide search in a space of evolutionary metaheuristics. Hyper-heuristics are used for solving the problem of automated algorithm selection or design. It is shown that hyper-heuristics are domain-in depended. A selective hyper-heuristic with online learning is proposed. The experimental results and comparison with some well-studied techniques are presented and discussed.

© 2018 A. Pupkov, E. Sopov, I. Panfilov, V. Samarin, N. Telesheva, R. Kuzmich.

Peer-review under responsibility of the Kaunas University of Technology, Panevėžys faculty of Technologies and Business

Keywords: evolutionary algorithms; optimization problem; selection, combination, generation.

1. Introduction

Optimization problems take place in different scientific and applied fields. There exists variety of problem statements, which define different classes of optimization problems and corresponding approaches for solving these problems. Many real-world optimization problems are known as complex (or hard). Usually there is very poor

* Corresponding author. Tel.: +7-908-019-74-33.

E-mail address: RKuzmich@sfu-kras.ru

© 2018 A. Pupkov, E. Sopov, I. Panfilov, V. Samarin, N. Telesheva, R. Kuzmich.

Peer-review under responsibility of the Kaunas University of Technology, Panevėžys faculty of Technologies and Business

information on the objective landscape, the search space and other features of the problem or this information cannot be introduced into optimization algorithms. Such problems are called “black-box” optimization problems (BBOPs).

Although we cannot use many well-studied exact methods of mathematical programming for BBOPs, there exist many efficient heuristics, which are generally based on experts’ knowledge or experience and are domain- or problem-specific. A heuristic produces an optimal solution (or a good enough suboptimal solution) more quickly when classic methods are too slow, or finds an approximate solution when classic methods fail to find any exact solution.

In real-world BBOPs we cannot choose a proper heuristic to the given problem. Moreover, as the heuristic is problem-specific, we need to adapt it to the new instance of a BBOP. A more advanced technique is a metaheuristic search. The term “metaheuristic” was proposed at mid-80s as a family of searching algorithms which are able to define a high level heuristic used to guide other heuristics for a better evolution in the search space. An advantage of a metaheuristic is that it requires no special knowledge on the given optimization problem to be solved. There have been proposed a large variety of metaheuristics for BBOPs. The majority of the approaches and the best results are achieved in a field of nature- and bio-inspired algorithms. In this study will focus on evolutionary (EAs) and genetic algorithms (GAs), which have proved their efficiency with wide range of optimization problems. Unfortunately, when we deal with hard BBOPs, EAs still require fine-tuning of its structure and parameters for sufficient or better performance. Many different approaches, including parameterless EAs, EAs with adaptive parameters, self-tuning (self-adaptation, self-configuration) EAs, have been proposed and discussed. We can found from many papers that the most recent studies are devoted to a problem of automating the design of metaheuristic optimization algorithms. In this case, we can talk about hyper-heuristics.

The term “hyper-heuristics” was first used Schaefer et al. [6] in 2001. The discussed hyper-heuristic was applied for a combinatorial optimization problem and was considered to be a high-level approach that, given a particular problem instance and a number of low-level heuristics, can select and apply an appropriate low-level heuristic at each decision point. In short, we can define a hyper-heuristic as a metaheuristic that designs a metaheuristic.

In real world optimization problems, the problem of designing, tuning and applying an appropriate search algorithm is viewed as single-use one. At the same time, in the field of computational intelligence iterative processes are used for training a computational model, which is based on previously solved instances of the problem (called a training dataset), and which is applied after that to new instances (a test dataset or a prediction with new data). The same idea can be used with EAs and GAs. In this study we propose and discuss a hyper-heuristic approach which demonstrates high efficiency within wide range of optimization problems of chosen problem classes.

The rest of the paper is organized as follows. Section 2 describes works related to hyper-heuristics classification. Section 3 describes known and well-studied and the proposed approaches. In Section 4 some results of numerical experiments are discussed. In the Conclusion the results and further research are discussed.

2. Related work

The term hyper-heuristic is relatively new. However, the idea of automating the design of heuristics is not new, and it can be found across Operational Research, Computer Science and Artificial Intelligence. In the field of EAs, the problem of automating the design of an EA’s structure and parameters is also known as the “self-configuration” problem. According to Schaefer et al. [6] the term self-configuration has several meanings.

Deterministic Parameter Control using some deterministic rules without taking into account any feedback from the evolutionary search. For example, the time-dependent change of the mutation rates.

Self-control or self-adaptation of the control parameters of the EA. Usually optimization of the control parameters is viewed as a part of the optimization objective. Values of controlled parameters are included in the population or in the chromosome of individuals. The advantage of the approach is that we solve the only optimization problem to cover two initial problems. Unfortunately, the new search space is larger and the new optimization problem can be more complex.

Meta-heuristics for the EA control that combine the EA with some other adaptive search techniques. Usually the approach is based on the external control procedure. Good results are obtained using the fuzzy-logic technique.

Self-configuration via the automated and adaptive design of an efficient combination of the EA components from the predefined set.

Third and fourth options are the most perspective as they control both an algorithm's structure and values of its parameters. And they both use meta-procedure for providing self-configuration.

More straight-forward approach based on using an EA to tune an EA. According to Freisleben and Härtfelder [3] this can be done using two EA: one for problem solving and another one (so-called meta-evolutionary algorithm) to tune the first one.

These and many other approaches are related to the idea of searching over a space of possible algorithm configurations, and are, therefore, related to hyper-heuristics.

We can define a hyper-heuristic as a high-level approach that, given a particular problem instance and a number of atomic heuristics, selects and applies an appropriate heuristic at each decision point. This definition of hyper-heuristics was expanded to refer to an automated methodology for selecting or generating heuristics to solve hard computational search problems in Burke EK et al. [1].

In the field of machine learning, there is an approach for selecting an appropriate algorithm or for generating new algorithms by model combinations. This approach is similar to the hyper-heuristic approach in optimization, and is called meta-learning. Different types of the meta-learning are presented in Fig. 1 according to Pappa GL et al. [5].

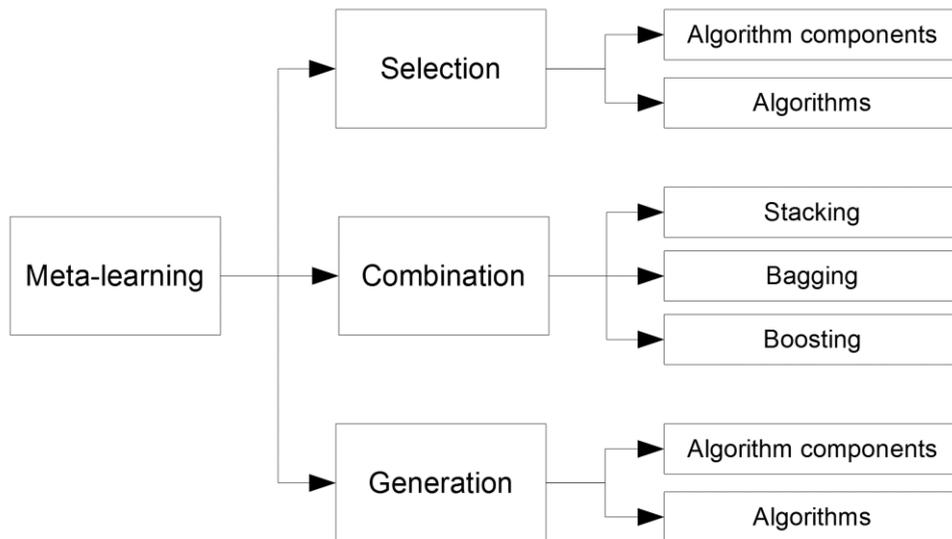


Fig. 1. Classification of meta-learning approaches in machine learning.

In Vrugt et al. [12] similar approach has been proposed for classification of hyper-heuristics in optimization. Burke has proposed the great survey and new classification in the field of hyper-heuristics.

In Cowling et al. [2], a general classification of hyper-heuristics according to two dimensions: the nature of the heuristic search space and the source of feedback during learning are proposed. The first characteristic defines the following two classes:

- Heuristic selection: Methodologies for choosing or selecting existing heuristics.
- Heuristic generation: Methodologies for generating new heuristics from components of existing heuristics.

A second level in this dimension corresponds to the distinction between constructive and perturbative search paradigms. Perturbative methods work by considering complete candidate solutions and changing them by modifying one or more of their solution components, while constructive methods work by considering partial candidate solutions, in which one or more solution components are missing, and iteratively extending them.

With respect to the source of feedback during learning:

- Online learning hyper-heuristics: Learn while solving a given instance of a problem.
- Offline learning hyper-heuristics: Learn, from a set of training instances, a method that would generalize to unseen instances.

- No-learning hyper-heuristics: Do not use feedback from the search process.

The Burke's classification is presented in Fig. 2.

As we can see from surveys, the majority of known hyper-heuristics are proposed for combinatory optimization problem. At the same time, there is a lack of approaches for BBOPs using EAs and GAs.

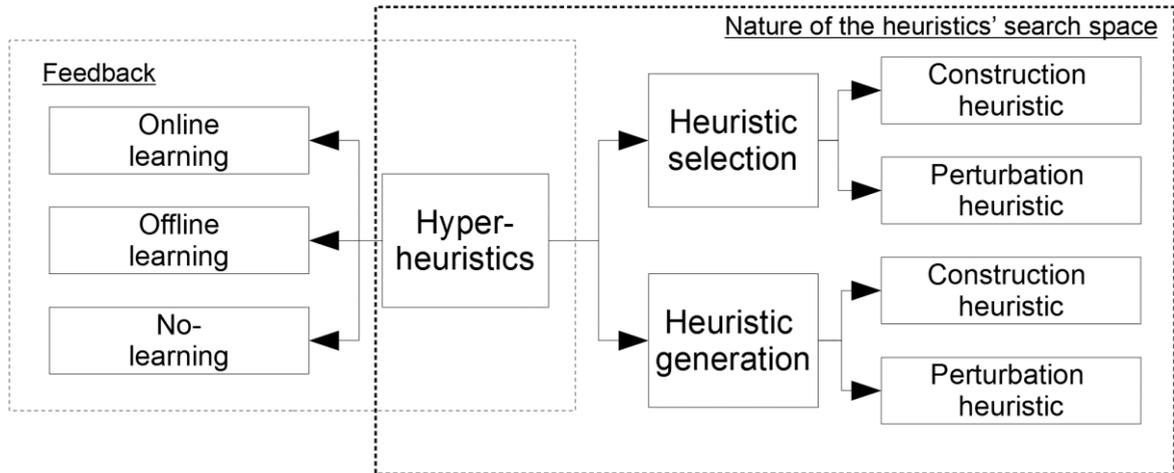


Fig. 2. Classification of hyper-heuristics in optimization.

In Swan and Woodward [10] it is also mentioned that a hyper-heuristic can be decomposed into hyper- and problem-layers. The problem-layer is domain-specific and contains set of low level heuristics, problem description and fitness-function. The hyper-layer contains methodologies to decide which low level heuristic to apply to which solution based on the history of previously visited solutions and their fitness values. These methodologies can be domain-independent. In this study, we will focus on the hyper-layer.

3. Hyper-heuristic approaches

We will discuss heuristic generation methodologies at first. The defining feature of this class is that the hyper-heuristic searches a space of heuristics constructed from components rather than a space of complete, pre-defined, heuristics.

The best results are achieved with genetic programming (GP) in Koza and Poli [4], which is an evolutionary computation technique that evolves a population of computer programs. The GP is the most common methodology used in the literature to automatically generate heuristics. However, GP is not inherently a hyper-heuristic, as the evolved programs can also directly represent problem solutions. At the same time, the GP can be used for generating low level heuristics for heuristic selection methods (for example, we can generate a distribution of probabilities for operations of selection and mutation in GAs) or a complete EA or GA.

Although the GP is widely applied in the field of computational intelligence, there is a lack of heuristic generation approaches for EAs or GAs using the GP.

Heuristic selection methodologies are wider spread, especially for well-studied techniques. One of the known approaches is an ensemble method, which control operation (select) of many different Gases. According to the above-mentioned classification, the approach is the selection hyper-heuristic as it operates with pre-defined EAs, and it is the online learning hyper-heuristic as it uses feedback from the search process to control interactions of component algorithms in the ensemble. As is known from the field of statistics and machine learning, on average, the collective solution of multiple algorithms provides better performance than could be obtained from any of the constituent algorithms. Thus we will try to reach two goals of automating the EA design and improving the EA performance on complex optimization problems.

The idea of multi-EA search is not new, but there exist only few efficient techniques. Some known approaches are a Multi algorithm genetically adaptive method for single objective optimization (AMALGAM-SO) in Vrugt et al. [12], Population-based algorithm portfolio (PAP) in Tang et al. [11], Multiple evolutionary algorithm (MultiEA) in Yuen et al. [14], Multi-strategy ensemble dynamic multi-objective evolutionary algorithm (MS-MOEA) in Wang and Li [13], and others.

In our previous studies the following selective hyper-heuristic in a form of ensemble have been proposed in Sopov [8]. The main idea of the approach is to include different EAs with different structures and parameters in the ensemble and to design effective control of algorithm interaction. We assume that if the hyper-heuristic operates with very different EAs with fundamentally different properties, it is able to improve the use of their individual advantages and minimize the effects of the disadvantages. Our hypothesis is that different EAs are able to deal with different features of the optimization problem, and the probability of all algorithms failing with the same challenge in the optimization process is low. Moreover, the interaction of algorithms can provide the ensemble with new options for optimization, which are absent in component algorithms.

The proposed hyper-heuristic approaches combine concepts of the island model and cooperative and competitive co-evolutions. We will discuss some general ideas of the proposed hyper-heuristic; more detailed information can be found in Sopov [8].

The general structure of the proposed method is called Self*GA in accordance with a common notation, where “Self” means that the approach is self-configuring, the star sign corresponds to the certain optimization problem.

The total population size (or the sum of populations of all component GAs) is called the computational resource. The resource is distributed between algorithms, which run in parallel and independent over the predefined number of iterations (called the adaptation period). All algorithms have the same objective and use the same encoding (solution representation). All populations are initialized at random. After the distribution, each GA included in Self*GA has its own population which does not overlap with populations of other GAs. At the first iteration, all algorithms get an equal portion of the resource. This concept corresponds to the island model, where each island realizes its own search strategy.

After the adaptation period, the performance of individual algorithms is estimated with respect to the objective of the optimization problem. After that, algorithms are compared and ranked. GAs with better performance increase their computational resource (the size of their populations) by decreasing population sizes of other algorithms in the ensemble. At the same time, all algorithms have a predefined amount of resource that is not distributed to give a chance for algorithms with low performance. This concept corresponds to the competitive co-evolution scheme. On this stage the selection of meta-heuristics from a predefined set is performed.

Finally, migrations of the best solutions are set to equate the start positions of algorithms for the run with the next adaptation period. According to the optimization problem, such a migration can be deterministic, selection-based or random. This concept corresponds to cooperative co-evolution.

Such a technique eliminates the necessity to define an appropriate GA with efficient search strategy for the problem as the choice of the best algorithm is performed automatically and adaptively during the run. Moreover, the hyper-heuristic is domain-independent. A domain-specific knowledge is applied only on low heuristic level with a fitness function evaluation or with genetic operation (if an operation based on problem-specific representation).

We have summarized some previous experimental results for BBOPs of different classes, which are presented and discussed in the following section.

4. Experimental results

4.1 Dynamic optimization problems

Non-stationary optimization problems are also called dynamic optimization problems (DOP) or changing (non-stationary, dynamic) environment optimization. There exist the following types of environmental changes: coordinate transformation, landscape rescaling, landscape stretching. There also exist a great variety of DOP techniques to deal with the certain type of changes. Most of them are based on two basic ideas: the population diversity increasing and the memorizing previous solutions. As is shown in many studies, there is no universal approach to adapt to all types of changes in the environment. At the same time, real-world problems of non-stationary optimization include various types of changes and are poorly predictable.

We will introduce an approach based on the Self*GA concept. It is called SelfDOPGA. In this study, we will use the following list of 5 basic and well-studied DOP techniques in the ensemble: self-adaptive GA, restarting optimization, local adaptation, diversity increasing technique, and the explicit memory. In the SelfDOPGA, the adaptation period is defined by the period between changes in the environment. The detection of changes can be performed by re-evaluating the fitness of the current best solutions (called detectors). After each adaptation period is completed, we need to estimate the performance of single algorithms. The performance measure is the offline error. The redistribution scheme is random migrations. We do not use the standard scheme “the best displaces the worst” to prevent the convergence and the decrease of the population diversity.

The performance of the SelfDOPGA was investigated at solving the following benchmark problems of non-stationary optimization. Two standard non-stationary optimization problems: the moving peaks benchmark (MPB) and the dynamic Rastrigin function. The MPB uses “Scenario 2” with the number of peaks equal to 1, 5, 10 and 20. The dimension for the Rastrigin problem is equal to 2, 5 and 10. Three problems from the CEC’2009 competition on dynamic optimization: Rotation peak function (F1), Composition of sphere's function (F2) and Composition of Rastrigin’s function (F3). The types of changes for F1-F3 are denoted as T1-T6. More details can be found in Sopov [7].

The results of the SelfDOPGA runs are compared with the performance of algorithms from the CEC’09 competition. The algorithms are a self-adaptive differential evolution algorithm (jDE), a clustering particle swarm optimizer (CPSO), Evolutionary Programming with Ensemble of Explicit Memories (EP with EEM), a standard particle swarm optimizer (PSO) and a standard genetic algorithm (GA).

In the case of the F1-F3 problems. The SelfDOPGA outperforms both the GA and PSO algorithms. It also outperforms EP in some cases. But yields to jDE and CPSO. It should be noted that the SelfDOPGA outperforms the average value, and the value of the SelfDOPGA performance is closer to the best value than to the average.

CPSO, jDE and EP are specially designed techniques. Numerical experiments show that their performance can decrease for some types of changes. The SelfDOPGA demonstrates sufficiently good performance within the whole range of types of changes.

4.2 Multi-objective optimization problems

Evolutionary multi-objective (MO) optimization is one of the fastest growing fields of research and application. EAs are highly suitable for MO and are flexible and robust in finding an approximation of the Pareto set. At the same time, many MOPs are still a challenge for EA-based techniques.

There exist various MO search strategies, which are implemented in different certain EAs. All of the techniques have their own advantages and disadvantages. Thus there is a good idea to combine different MO search strategies in a form of ensemble and design Self*GA that can be named SelfMOGA. We use five basic and well-studied techniques: VEGA, FFGA, NPGA, NSGA-II, SPEA-II.

We have used the following performance evaluations of a single algorithm. We have used the following criteria, combined into two groups. The first group includes the static criteria (the performance is measured over the current adaptation period):

- criterion 1 is the percentage of non-dominated solutions. The pool of all algorithm solutions is created and non-dominated sorting is performed. Finally, the number of non-dominated solutions corresponding to each algorithm is counted.
- criterion 2 is the uniformity (dispersion) of non-dominated solutions. The average variance of distances between individuals is computed using coordinates (for the Pareto set) or criteria (for the Pareto front).

The second group contains the dynamic criteria (the performance is measured in a comparison with previous adaptation periods):

- criterion 3 is the improvement of non-dominated solutions. The solutions of the previous and current adaptation periods are compared. The improvement is completed if the current solutions dominate the previous ones, even if its number has decreased.

The SelfMOGA performance was investigated using 19 benchmark problems. Problems 1-6 are various 2D test functions based on quadric, Rastrigin, Rosenbrock and others. The number of objectives varies from 2 to 4. These problems are good for analysis of the algorithm performance as we can visualize the Pareto set and the Pareto front. Problems 7-19 are taken from the CEC 2009 competition on MO. The following functions are chosen: FON, POL, KUR, ZDT1-4, ZDT6, F2, F5, F8, UP4, and UP7. More details can be found in Sopov [8].

We have estimated the performance of the SelfMOGA, each component algorithm from the ensemble and the mean value (average of all component algorithms). The mean value demonstrates the average performance of the randomly chosen technique. The SelfMOGA can yield to the best algorithm in some cases, but always outperforms the mean value. So we can recommend the SelfMOGA as an efficient and universal technique for complex MO BBOPs.

4.3 Multi-modal optimization problems

Many real-world problems have more than one optimal solution, or there exists only one global optimum and several local optima in the feasible solution space. Such problems are called multimodal. The goal of multimodal optimization (MMO) is to find all optima (global and local) or a representative subset of all optima.

We will discuss the design of a Self*GA for MMO problems that can be named SelfMMOGA. In this study we use six basic techniques, which are well-studied and discussed, and they can be used with binary representation with no modification. Algorithms are Clearing, Sharing, Clustering, Restricted Tournament Selection (RTS), Deterministic Crowding (DC), and Probabilistic Crowding (PC). More detailed information on the algorithms and their specific parameters are presented in Sopov [9].

For MMO problems performance metrics should estimate how many optima were found and how the population is distributed over the search space. Unfortunately, good performance measures exist only for benchmark MMO problems, which contain knowledge of the optima. Performance measures for black-box MMO problems are still being discussed. In this study, the following criteria are used.

The first measure is called Basin Ratio (BR). The BR calculates the number of covered basins, which have been discovered by the population. It does not require knowledge of optima, but an approximation of basins is used. The second measure is called Sum of Distances to Nearest Neighbour (SDNN). The SDNN penalizes the clustering of solutions. This indicator does not require knowledge of optima and basins. Finally, we combine the BR and the SDNN in an integrated criterion K (sum of BR and SDNN).

To estimate the approach performance, we have used the following list of benchmark problems. Six binary MMO problems, which are based on the unimodal functions, and they are massively multimodal and deceptive. Eight real-valued MMO problems are from CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization. We have denoted the functions in the following way: cecF1-cecF8 and binaryF11-binaryF16.

We also have compared the results on binary problems with Ensemble of niching algorithms (ENA). There is no statistically sufficient difference between SelfMMOGA and ENA. We have also compared the results with the average of 6 component algorithms. The average value can be viewed as the average performance of a randomly chosen algorithm. Such an estimate is very useful for black-box optimization problems, because we have no information about problem features and, consequently, about what algorithms to use. If the performance of the SelfMMOGA is better than the average of its component, we can conclude that on average the choice of the SelfMMOGA will be better. The SelfMMOGA always outperforms the average of its stand-alone component algorithms for binary problems.

We have also compared the results of the SelfMMOGA runs with some efficient techniques from the competition on continuous problems. The techniques are DE/rand/1/bin and Crowding DE/rand/1/bin, N-VMO, dADE/rand/1, and PNA-NSGAI. The SelfMMOGA shows results comparable with popular and well-studied techniques. It yields to dADE/rand/1 and N-VMO, but we should note that these algorithms are specially designed for continuous MMO problems, and have taken 2nd and 4th places, respectively, in the CEC competition. At the same time, the SelfMMOGA has very close average values to the best two algorithms, and outperforms PNA-NSGAI, CrowdingDE and DE, which have taken 7th, 8th and 9th places respectively.

5. Conclusions and further work

Hyper-heuristic methodologies are fast growing field of evolutionary computation and optimization. Hyper-heuristic methods can be used for automated design of EAs' structure and parameters. The constructive hyper-heuristics can generate new low-level heuristics and meta-algorithms. The selective hyper-heuristics can operate with a predefined list of heuristics. In this study, we have reviewed some results for the proposed approach that is

selective hyper-heuristic with online learning. As experiments have shown that the approach is domain-independent and outperforms the random choice of search technique.

In further work, a constructive method based on GP will be proposed and investigated for generating new low-level heuristics, which after that will be included into a selective hyper-heuristic.

6. Acknowledgements

The research is performed with the financial support of the Ministry of Education and Science of the Russian Federation within the State Assignment for the Siberian State Aerospace University, project assignment 2.1676.2017/ПЧ.

References

- [1] Burke EK, et al. A Classification of Hyper-heuristic Approaches. *Handbook of Metaheuristics, International Series in Operations Research & Management Science* 2010; 146,15: 449-468.
- [2] Cowling P, Kendall G, Soubeiga E. A parameter-free hyper heuristic for scheduling a sales summit. In: *Sousa JP and Resende MGC (eds). Proceedings of the 4th Metaheuristic International Conference 2001; Springer Porto, Portugal:127-131.*
- [3] Freisleben B, Härtfelder M. *Optimization of genetic algorithms by genetic algorithms.* In: *Albrecht RF, Steele NC and Reeves CR (eds). Artificial Neural Nets and Genetic Algorithms 1993; Springer Verlag, Berlin:392-399.*
- [4] Koza JR, Poli R. Genetic programming. In: *Burke E and Kendall G (eds). Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* 2005; Kluwer Boston, MA:127-164.
- [5] Pappa GL, et al. Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. *Genetic Programming and Evolvable Machines* 2014; 15:3-35.
- [6] Schaefer R, Cotta C, Kolodziej J. Parallel Problem Solving from Nature. In: *Proc. PPSN XI 11th International Conference* 2010.
- [7] Sopov E. On the investigation of Coevolutionary Genetic Algorithm for Self-configuring Solving of Non-stationary Optimization Problems. *Proc. International Conference on Computer Science and Artificial Intelligence (ICCSAI 2014)* 2014;202-207.
- [8] Sopov E. Multi-strategy Genetic Algorithm for Self-Configuring Solving of Complex Optimization Problems. *Proc. of the 4th International Congress on Advanced Applied Informatics (IIAI-AAI 2015)* 2015; 556-561.
- [9] Sopov E. Self-configuring ensemble of multimodal genetic algorithms. *Studies in Computational Intelligence* 2017; 669:56-74.
- [10] Swan J, Woodward J. Searching the Hyper-heuristic Design Space. *Cognitive Computation* 2014; 6(1):66-73.
- [11] Tang K, Chen G, Yao X. Population-Based Algorithm Portfolios for Numerical Optimization. *IEEE Transactions on Evolutionary Computation* 2010; 14(5):782-800.
- [12] Vrugt JA, Robinson BA, Hyman JM. Self-Adaptive Multi method Search for Global Optimization in Real-Parameter Spaces. *IEEE Transactions on Evolutionary Computation* 2009; 13(2):243-259.
- [13] Wang Y, Li B. Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. *Memetic Computing* 2010; 2(1):3-24.
- [14] Yuen SY, Chow ChK, Zhang X. Which Algorithm Should I Choose at any Point of the Search: An Evolutionary Portfolio Approach. In: *the Proceedings of the 15th annual conference on Genetic and evolutionary computation* 2013;567-574.