UDC 519.85

Vestnik SibGAU
Vol. 16, No. 1, P. 124–130

SELF-CONFIGURING MULTI-STRATEGY GENETIC ALGORITHM FOR NON-STATIONARY ENVIRONMENTS

E. A. Sopov

Siberian State Aerospace University named after academician M. F. Reshetnev
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660014, Russian Federation
E-mail: evgenysopov@gmail.com

Many real-world problems of design and control in a field of the aerospace lead to optimization problems. Such optimization problems are complicated and become a great challenge to many optimization techniques. Moreover, many real-world optimization problems are dynamic and changing over time. Changes occur in the parameters, objectives and/or problem constraints. In this case, search algorithms should have the capability to track moving optima and adapt to a new environment. In past years many approaches for non-stationary optimization were proposed. The best results are achieved using a stochastic population-based search such as evolutionary and genetic algorithms. Unfortunately, real-world non-stationary optimization problems include various types of changes and are poorly predictable, thus there is a problem of choosing a proper optimization technique and tuning its parameters. This study presents a novel approach for designing a multi-strategy genetic algorithm based on a hybrid of the island model, cooperative and competitive coevolution schemes. The approach controls interactions of different genetic algorithms and leads to the self-configuring solving of problems with a priori unknown structure. A short survey on non-stationary optimization problem and methods is presented. The results of numerical experiments for benchmark problems from the CEC competition are discussed. The proposed approach has demonstrated efficiency comparable with other well-studied techniques for non-stationary optimization. And it has significant advantage – it does not require the participation of the human-expert, because it operates in an automated, self-configuring way.

Keywords: dynamic optimization, non-stationary environment, self-configuring, genetic algorithm, coevolution.

Вестник СибГАУ
Т. 16, № 1. С. 124–130

САМОКОНФИГУРИРУЕМЫЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ НА БАЗЕ МНОЖЕСТВА СТРАТЕГИЙ ПОИСКА В НЕСТАЦИОНАРНОЙ СРЕДЕ

Е. А. Сопов

Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660014, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
E-mail: evgenysopov@gmail.com

Многие практические задачи проектирования и управления в аэрокосмической отрасли приводят к задачам оптимизации. Подобные задачи являются сложными и затрудняют применение многих методов оптимизации. Более того, многие практические задачи оптимизации являются динамическими и меняются с течением времени. Изменения происходят в параметрах задачи, целевых функциях и/или ограничениях. В этом случае алгоритмы оптимизации должны иметь возможность отслеживать меняющие положение оптимумы и постоянно адаптироваться к новой среде. Ранее было предложено множество подходов для решения задач нестационарной оптимизации. Наилучшие результаты демонстрируют стохастические популяционные алгоритмы, такие как эволюционные и генетические алгоритмы. Представлен новый подход для проектирования генетического алгоритма, включающего множество стратегий поиска, который основан на гибридизации островной модели, кооперативной и конкурирующей коэволюционных схем. Такой подход осуществляет управление взаимодействием многих генетических алгоритмов, что приводит к самоконфигурируемому решению задач оптимизации с априори неизвестной структурой. Представлен краткий обзор проблемы и методов решения задач нестационарной оптимизации. Приводится анализ результатов численных экспериментов на множестве задач, представленных на соревновании по нестационарной оптимизации в рамках международной конференции CEC. Предложенный подход демонстрирует эффективность, сравнимую с другими хорошо изученными подходами для решения задач нестационарной оптимизации. При этом подход имеет существенное преимущество – он не требует привлечения специалиста, так как является самоконфигурируемым и решает задачу оптимизации в автоматизированном режиме.

Ключевые слова: динамическая оптимизация, нестационарная оптимизация, самоконфигурация, генетический алгоритм, коэволюция.

Introduction. Many real-world optimization problems are non-stationary. Changes in the environment can take various forms such as changes in the parameters, objectives or problem constraints. As a result, in such dynamic environments the position of the global and local optima changes. This feature is called the optima drift. Non-stationary optimization problems are also called dynamic optimization problems (DOP) or changing (non-stationary, dynamic) environment optimization [1].

There exist many examples of real-world problems in a field of the aerospace, that are non-stationary. For example, the problem of fuel mixture optimization for jet engines. The dependence of the engine output on the mixture quality changes over time due to degradation of engine components or with component failure. When planning routes of civil aviation, as well as planning landings and takeoffs, it is often required to design a new plan on the fly, for example when the original plan changes or there are changes of weather conditions, etc.

Other examples from other areas are for example in game theory with a change of opponent strategy, in job shop scheduling problems with the addition of a new job, in financial problems with changes in markets, in speech recognition with changes in the acoustic environment, etc.

In the field of mathematical optimization, there exists framework for modelling optimization problems that involve uncertainty. It is called stochastic programming. Stochastic programming is based on a fact that the probability distributions of optimization model parameters are known or can be estimated. In real-world applications, the nature of non-stationarity is not known beforehand, and often is not stochastic. Thus, search optimization methods are preferred and are often the only ones applicable.

In the case of stationary optimization, the main goal is to find the optimal solution to the problem (fig. 1). The efficiency criteria for optimization algorithms are accuracy and the number of objective evaluations (or computational cost).

In the case of non-stationary optimization, the position of optima is changing over time, so the search algorithm should have the capability to track moving optima and adapt to a new environment. Efficiency criteria are accuracy and adaptation speed. Unfortunately, standard search optimization methods are not able to adapt to changes. Moreover, they converge to the best-found solution and lose the previously collected information about search space (fig. 2).

It is obvious that the traditional optimization techniques are not efficient enough with non-stationary problems. Many researchers prefer the population-based approaches that are able to track the optimum position more efficiently as the search involves several parallel solutions. The best results for DOP are achieved using nature-inspired algorithms [2].

Related work. A non-stationary optimization problem or DOP can be defined as:

$$f(\bar{x},t) \to \underset{\bar{x} \in D(t)}{extr},$$

$$g_i(\bar{x},t) \le 0, i = \overline{1,q},$$

$$h_i(\bar{x},t) = 0, i = \overline{q+1,m},$$

$$\bar{x} = (x_1, x_2, \ldots, x_N) \in D(t), D(t) \subseteq S, f : D \to R, t \in T,$$

where S is the search space; $D(t)$ is a set of feasible solutions, constrained with g_i and h_i; t is the time.

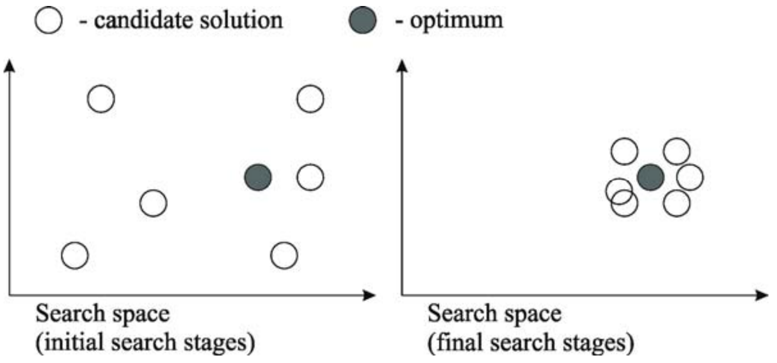

Fig. 1. Search algorithm converges to optimum

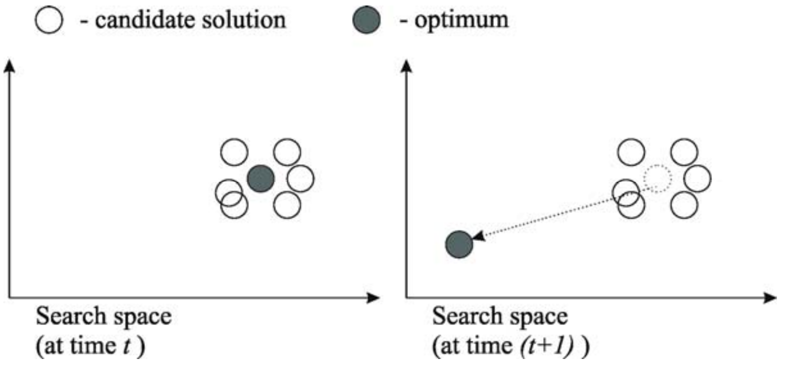

Fig. 2. Environment has changed and optimum has moved

In such a problem statement, the objective and the constraints depend on the solution vector $\overline{x}$ and on the certain value of time *t*. This means that the objective and the feasible region of the search space are both changing over time. However, the problem can be considered as a unique stationary problem at each fixed point of time.

The solution to the problem of non-stationary optimization is a set of global optima for all values of time (in a case of minimization):

$$\overline{x}^*(t) = \{\overline{x} \in D(t) | \forall \overline{x}' \in D(t), f(\overline{x},t) \le f(\overline{x}',t)\}, t \in T,$$

where $\overline{x}^*$ is the global optimum value.

A good survey on DOP methods is proposed in [3]. The range of search techniques is rather wide and contains:

– evolutionary and genetic algorithms (EA and GA);
– evolutionary strategies (ES);
– evolutionary programming (EP);
– cooperative strategies (CS);
– cultural algorithms;
– swarm intelligence (PSO);
– estimation-of-distribution algorithms (EDA);
– immune-based algorithms;
– memetic algorithms;
– ant colony optimization;
– self-organizing scouts;
– neural networks;
– other techniques.

It should be noted that a large proportion of works and of the better results are related to evolutionary algorithms. Other approaches are good in some specific applications.

There exist the following types of environment changes that define certain non-stationary optimization problem [4]:

1. *Coordinate transformations* lead to a new position of optima. Values and the structure of optima do not change. There are the following coordinate transformations:

– *Drifting landscape*. The whole landscape moves in some direction with a constant, variable or random speed value;

– *Rotating landscape*. The landscape is rotating around a point in the search space;

– *Chaotic coordinate changes*.

2. *Landscape rescaling* leads to changes of objective values while maintaining the overall structure of optima. Values may either increase (swelling) or decrease (shrinking).

3. *Landscape stretching* leads to landscape topology changes while maintaining the overall positions of optima.

A known specialist in the field of non-stationary optimization, Shengxiang Yang, notes that most real-world optimization problems have the following properties [5]:

– DOPs are non time-linkage problems;
– changes are assumed to be detectable;
– coordinates and topology are changing;
– DOPs have unpredictable changes;
– DOPs have cyclic/recurrent changes.

Standard evolutionary algorithms have no features that provide adaptation to environment changes. Thus, researchers use different heuristics. There are two main approaches: maintaining the diversity of population and memorizing past search experience. Known techniques include the following:

1. Restarting optimization. If a change in the environment is detected, the search is restarted. As a restart leads to a completely new optimization, there is no influence of a certain change type.

2. Local adaptation. If only slight changes occur, it is reasonable to use local operators to create new solutions in the local region of the current individuals. Most of the known approaches uses a modified mutation operator, for example hypermutation, variable local search (VLS), distance-based mutation in the multinational GA and others. Another way is to use the external local search procedure.

3. Memorizing previous solutions. In a case of time-dependent changes, new landscapes can become very similar to previous ones, so it is a good idea is to memorize previous solutions to guide the search to those previously explored regions in the search space. There are two different approaches: explicit and implicit memory techniques.

The explicit memory uses external storage for previous candidate solutions that can be reevaluated and reinserted into the population if a change in the environment occurs. The implicit memory uses the concept of the diploid chromosomes which contain two sets of genes encoding the solution. Only one set participates in selection and reproduction, and this set is called dominant set. If a change occurs, the dominance of the sets is reevaluated and the dominant set can be substituted by the redundant part of chromosome.

4. Diversity increasing techniques. Almost every approach in the field of DOP focuses on the maintenance of genetic diversity in the population. There are at least three groups of special techniques.

The first technique increases diversity by introducing random individuals into the population with each generation. This group includes random immigrants, partial hypermutation and hypermutation techniques. The second group is niching techniques. The main idea is to decrease the fitness value for solutions if some local region (called the niche) around them contains other individuals. The third group uses the idea of restricted mating. The population is divided into several subpopulations. The recombination operator is restricted to individuals of the same subpopulation. A more simple way is to restrict the recombination of the closest individuals.

5. Multi-population techniques. The population is divided into competing subpopulations. Each subpopulation responds for a particular region of the search space or may have its own goal. Subpopulations may interact to improve the search. This approach combines diversity, memory and adaptation mechanism. The implementation of the multi-population technique depends on the particular optimization problem that leads to a specific partition of the population and specific goals of subpopulations.

6. Self-adaptive techniques. Self-adaptation is very popular and successful in stationary optimization. In DOP,

the adaption involves learning from past experience. The term "self-adaptation" means that control parameters of evolutionary algorithm are encoded in the chromosome with the representation of the solution. Thus, the evolutionary algorithm implements the search for a solution of the original problem and the adjustment of the algorithm parameters under the current search situation.

7. Algorithms with overlapping generations. A standard evolutionary algorithm forms a new population with every next generation. The previous population is eliminated, but the search experience is saved in implicit form in the genes of the current population. In DOP, better results are obtained with the steady-state model instead of the generational model. The steady-state model uses only a small part of the population (usually only two solutions) on each iteration to reproduce a small number of new solutions.

8. Learning of the underlying dynamics. Many future changes in the environment can be predicted using information about previous changes and the current situation. It is reasonable to use this information to adapt the search according to predictions. Cyclic changes can be predicted in an implicit form using memory techniques. However, arbitrary changes require the development of some external procedures that can be trained on historical data. There exists many approaches using mathematics, statistics or machine learning.

Prediction-based algorithms have disadvantages, mostly due to training errors. The algorithms may need a relatively large set of training data to produce good results, so it means that the prediction can be started after sufficient training data has been collected. If the algorithm has not performed successfully in the previous change periods, the historical data collected by the algorithm might provide the wrong training data. Moreover, not all types of changes can be predicted.

The main disadvantage of dynamics modelling is that this approach uses techniques different from evolutionary algorithms. Choosing the correct method for analysis and predicting the dynamics of changes, their training and implementation are complex problems themselves, comparable in complexity to the original optimization problem.

The experimental investigation of different techniques shows that there is no universal approach to adapt to all types of changes in the environment. As mentioned above, real-world problems of non-stationary optimization include various types of changes and are poorly predictable. Moreover, even if the type of changes is a priori known, one should select the proper structure of evolutionary algorithm and fine tune its parameters.

Thus, there is an actual scientific problem of designing self-configuring techniques that are able to deal with many types of changes in the environment.

Self-configuring multi-strategy genetic algorithm. In the field of statistics and machine learning, ensemble methods are used to improve decision making. On average, the collective solution of multiple algorithms provides better performance than could be obtained from any of the constituent algorithms. This concept can be used in the field of EA. The main idea is to include different search strategies in the ensemble and to design effective control of algorithm interaction.

There are at least two well-studied approaches to the interaction of an EA: the coevolutionary approach and the island model.

The island model was introduced as a parallel version of an EA. In a parallel implementation of the island model, each machine executes an EA and maintains its own population for search. The machines periodically exchange a portion of their populations (it is called migration). In a case of separable search problems, the island model performs better results than the serial single population model.

The coevolution algorithm is an evolutionary algorithm in which fitness evaluation is based on interactions between individuals. The interaction can take place in a single population or in multiple populations, where each population can be processed by its own EA. There are two types of interaction: competitive and cooperative. Usually the coevolution is referred to problem decomposition, and the cooperative scheme is applied.

Coevolution can also be applied to perform the self-configuring. It is known that an EA has many parameters to be tuned. The incorrect values of the parameters lead to low efficiency of the algorithm. The self-configuring coevolutionary algorithm is a hybrid of the island model, competitive and cooperative coevolution. The total population is divided into disjoint subpopulations of equal size. The portion of the population is called the computational resource. Each subpopulation corresponds to certain EA with its own parameters values and evolves independently (corresponds to the island model). After some period, the performance of individual algorithms is estimated and the computational resource is redistributed. EAs with better performance increase their population size (the competitive scheme). Finally, random migrations of the best solutions are presented to equate start positions of EAs for the run with the next period (the cooperative scheme). Such a coevolution technique eliminates the necessity to define an appropriate algorithm for the problem as the choice of the best algorithm is performed automatically during the run [6; 7].

In [8; 9] a new self-configuring coevolutionary technique was introduced. It is based on the idea mentioned previously, but it uses many different search strategies instead of the only EA. The approach was designed and then investigated with complex multi-objective optimization problems. It was called the self-configuring coevolutionary multi-objective genetic algorithm or SelfCOMOGA.

This work presents a novel multi-strategy approach to non-stationary optimization, which is called in a similar way the self-configuring DOP genetic algorithm or SelfDOPGA. The main idea is to include in the self-configuring coevolutionary algorithm many search strategies, which can deal with many different types of changes in the environment. The competitive interaction should lead to an automated choice of the proper DOP algorithm. The cooperative interaction should supply all algorithms with useful information about past search experience collected and is presented in different forms.

In this work, all EAs are assumed to be genetic algorithms with binary representation. GAs are a well-studied technique, so many efficient algorithms can be used as core algorithms for DOP approaches [10; 11]. The binary GA allows to deal with problems that contain variables of various types (real, integer, Boolean, rank values, permutation, etc.).

The general SelfDOPGA scheme is as follows:

Step 1. Define a set of algorithms included in the coevolution.

Step 2. Perform algorithms run over some time until a change in the environment is observed.

Step 3. Estimate the performance for each algorithm over the period.

Step 4. Redistribute the computational resources and perform a new run (go to the step 2).

We will discuss the SelfDOPGA steps in detail.

The first step defines the search strategies. Any combination of DOP algorithms can be included in SelfDOPGA. The combination can be designed using a priori information about certain problem. Otherwise, different (or all available) strategies should be included. In this work, we will use the following list of DOP techniques (varied parameters are shown in brackets):

– Self-adaptive technique (probabilistic GA with self-configuring parameters);

– Restarting optimization (the percentage of the population that is substituted with new randomly generated individuals);

– Local adaptation (the distance-based mutation rate in multinational GA);

– Diversity increasing technique (the size of niche);

– The explicit memory (the size of external storage).

Some the techniques mentioned above were excluded. For example, the multi-population technique is already presented by different populations in the multi-national GA. Populations of GAs in the SelfDOPGA are also different as GAs use different search strategies. The dynamics modelling uses techniques different from evolutionary algorithms, so it cannot be applied in a form of the coevolution.

The second step is the running of algorithms until a change in the environment is observed. Detection of changes can be performed by re-evaluating the fitness of specific solutions (called detectors). The detector can be the current best individual. Also changes can be detected by analysis of algorithm behaviour (in the case the current best was out of the optima). Changes are detected based on monitoring the drop in value of the average of the best-found solutions over a number of generations.

The third step is performance estimation. The performance measure is the offline error [12], which is the average error at every time step over the best solution found by the algorithm since the last change in the environments:

$$OfflineError_i = \frac{1}{T_p}\sum_{t=1}^{T_p} fitness(x_{best}(t),t),$$

where T_p is the number of generations between two consequent changes; $x_{best}(t)$ is the best individual in a population at moment t and i is number of the algorithm.

The fourth step performs the redistribution of populations. All algorithms give to the "winner" algorithm a certain percentage of their population size. Each algorithm has a minimum guaranteed resource that is not distributed.

Experimental results. To investigate the performance of the SelfDOPGA, two test problems are used: the moving peaks benchmark (MPB) and the dynamic Rastrigin function. The MPB is the standard for testing DOP algorithms, as it is thought to be a simulation of various real-world optimization problems. The static Rastrigin function is a complex optimization problem with a large number of deceptive local optima. The dynamic version of the Rastrigin function simulates a complex dynamic problem. Both problems are well-studied, so we can compare the performance with many DOP techniques [12–14].

The MPB defines an n-dimensional landscape with a pre-set number of peaks (p). Peaks are defined with specific locations (denoted as X), heights (H) and widths (W). The peaks are distributed randomly in a pre-defined area. Peaks may vary its height, width and location over time. The fitness function for the MPB is formulated as follows:

$$f_{\mathrm{MPB}}(\bar{x},t) = \max_{i=1,\dots,p}\left(\frac{H_i(t)}{1+W_i(t)\cdot\sum_{j=1}^{n}(x_j - X_j(t))^2}\right).$$

A shift of a single peak can be defined as:

$$X_i(t) = X_i(t-1) + v_i(t),$$

$$v_i(t) = \frac{s}{|r + v_i(t-1)|}((1-\lambda)\cdot r + \lambda\cdot v_i(t-1)),$$

where r is a random shift vector; s is a parameter regulating the length of the movement (the severity). A parameter λ sets a balance between the random and directed movement. The value of $\lambda = 0.0$ results in a completely random direction of movement. The value of $\lambda = 1.0$ makes every move direction depend on the direction of the previous move.

The set of the MPB parameters is called the "scenario". The majority of the research in the field of non-stationary optimization uses Scenario 2 (tab. 1) [15].

Table 1

Parameter settings for MPB

Parameter	Settings
Dimensions (n)	5
Coordinates range ($x_{\min}$, $x_{\max}$)	[0,100]
Peak heights (H)	[30,70]
Peaks widths (W)	[1,12]
Evaluations between changes	5000
The severity (s)	1.0
Correlation coefficient (λ)	0.0

The dynamic Rastrigin function is defined as follows:

$$f_{Rastrigin}(\bar{x},t) = \\ = \sum_{i=1}^{D}\left[(x_i + \Delta(t))^2 - 10\cos(2\pi x_i + 2\pi\Delta(t)) + 10\right],$$

where $x_i \in [-5.12, 5.12]$; $\Delta(t)$ is a random number changed over times within $[-0.8, 0.8]$.

The SelfDOPGA settings are:

– Population size – 500 (all algorithms start with 100 individuals);

– Chromosome length (the MPB) – 80 bits (discretization step is 1.5×10^{-3});

– Chromosome length (dynamic Rastrigin function) – 10 bits per a dimension (discretization step is 10^{-2});

– All results are the averages of 100 independent runs;

– All algorithms are self-configuring in a way described in [11];

– The size of external storage for the explicit memory technique – 20.

The results of numerical experiments are shown in tab. 2 and 3. The results of the SelfDOPGA runs are compared with the performance of 5 individual algorithms and its average.

Table 2

Offline error for the MPB

Algorithm	$p = 1$	$p = 5$	$p = 10$	$p = 20$
Self-adaptive	0.5	3.27	9.96	8.72
Restarting optimization	1.25	5.03	10.87	9.35
Local adaptation	0.96	4.85	10.84	11.5
Diversity increasing	0.4	3.93	9.01	7.97
The explicit memory	1.98	7.63	16.25	17.82
Average	1.018	4.942	11.386	11.072
SelfDOPGA	0.82	4.56	7.12	7.33

Table 3

Offline error for dynamic Rastrigin function

Algorithm	$D = 2$	$D = 5$	$D = 10$
Self-adaptive	0	3.56	14.53
Restarting optimization	0	4.02	15.01
Local adaptation	0	6.74	21.23
Diversity increasing	0.7	6.87	21.8
The explicit memory	1.3	8.15	34.36
Average	0.4	5.868	21.386
SelfDOPGA	0	2.83	11.56

As we can see, the SelfDOPGA shows better performance that the average performance of individual techniques. The most significant difference is observed in complex problems with higher dimensionality. In the case of low dimensionality, the SelfDOPGA yields to the best algorithm. The memory approach shows low performance, because changes are not cyclic, but random.

Although there is a single technique that is better or equal to the performance of the SelfDOPGA, the choice of the appropriate algorithm requires the problem analysis. The SelfDOPGA does not use any a priori information about a problem and controls the search strategy automatically during the run.

Conclusions. Optimization in a changing environment is a complex problem. In many real-world problems, the changes are random and their types are unknown beforehand. There exists a great variety of search strategies that are efficient with a specific type of changes.

In this work, the self-configuring DOP genetic algorithm with a multiple strategy ensemble is proposed. The combination of competitive and cooperative coevolution schemes is used to control search strategies interaction. The SelfDOPGA has better performance than the average performance of individual algorithms and performs automatically in an adaptive way.

In further works, the SelfDOPGA should be extended with other DOP techniques. More non-stationary problems with different types of changes should be tested.

Acknowledgements. This research is supported by the Ministry of Education and Science of the Russian Federation within the State assignment № 2.1889.2014/K. The author expresses his gratitude to Mr. Ashley Whitfield for his efforts to improve the text of this article.

Благодарности. Исследования поддерживаются Министерством образования и науки Российской Федерации в рамках государственного проектного задания № 2.1889.2014/K. Автор выражает признательность Эшли Уитфильду за помощь в совершенствовании текста статьи.

References

1. Yang S., Jin Y. Evolutionary Computation in Dynamic and Uncertain Environments. Springer-Verlag, Berlin, Heidelberg, 2007, 605 p.

2. Nguyena T.T., Yang S., Branke, J. Evolutionary dynamic optimization: A survey of the state of the art. Swarm and Evolutionary Computation 6, 2012, p. 1–24.

3. Cruz C., González J. R., Pelta D. Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing*, Springer-Verlag, 2011, no. 15 (7), p. 1427–1448.

4. Weicker K. Evolutionary Algorithms and Dynamic Optimization Problems. Ph.D. dissertation thesis, Der Andere Verlag, 2003, 211 p.

5. Yang, S., Evolutionary Computation for Dynamic Optimization Problems. Tutorial, GECCO'13, 2013, p. 667–682.

6. Goh C.-K., Tan K. C. A competitive–cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 2009, no. 13(1), p. 109–127.

7. Ivanov I., Sopov E. [Self-configured genetic algorithm for multi-objective decision making support]. *Vestnik SibGAU*. 2013, no. 1(47), p. 30–35 (In Russ.).

8. Ivanov I., Sopov E. [Investigation of the self-configured coevolutionary algorithm for complex multi-objective optimization problem solving]. *Sistemy Upravleniya i Informatsionnye Tekhnologii*. 2013, no. 1.1(51), p. 141–146 (In Russ.).

9. Ivanov I., Sopov E. Design Efficient Technologies for Context Image Analysis in Dialog HCI Using Self-Configuring Novelty Search Genetic Algorithm. In the 11th International Conference on Informatics in Control, Automation and Robotics, ICINCO'14. Vienna, Austria, 2014, p. 832–839.

10. Sopov E. A., Sopov S. A. The convergence prediction method for genetic and PBIL-like algorithms with binary representation. *Proceeding of IEEE International Siberian Conference on Control and Communications (SIBCON'11)*, Tomsk, 2011, p. 203–206.

11. Semenkin E. S., Semenkina M. E. Self-configuring Genetic Algorithm with Modified Uniform

Crossover Operator. Advances in Swarm Intelligence. Lecture Notes in Computer Science 7331. Springer-Verlag, Berlin Heidelberg, 2012, p. 414–421.

12. Branke J., Schmeck H. Designing evolutionary algorithms for dynamic optimization problems. Theory and Application of Evolutionary Computation: Recent Trends, Springer-Verlag, 2002, p. 239–262.

13. Morrison R. W., De Jong K. A. A test problem generator for non-stationary environments. Proc. the 1999 Congr. on Evol. Comput., 1999, p. 2047–2053.

14. Li C., Yang S., Nguyen T.T. et al. Benchmark Generator for CEC2009 Competition on Dynamic Optimization, Technical Report 2008, Department of Computer Science, University of Leicester, U.K., 2008, 14 p.

15. Branke J., Memory enhanced evolutionary algorithms for changing optimization problems. In Proc. the Congr. on Evol. Comput, 1999, p. 1875–1882.

Библиографические ссылки

1. Yang S., Jin Y. Evolutionary Computation in Dynamic and Uncertain Environments. Berlin, Heidelberg : Springer-Verlag, 2007, 605 p.

2. Nguyena T. T., Yang S., Branke J. Evolutionary dynamic optimization: A survey of the state of the art // Swarm and Evolutionary Computation. 2012. № 6. Pp. 1–24.

3. Cruz C., González J. R., Pelta D. Optimization in dynamic environments: a survey on problems, methods and measures // *Soft Computing*. 2011. № 15 (7). Pp. 1427–1448.

4. Weicker K. Evolutionary Algorithms and Dynamic Optimization Problems : Ph. D. dissertation thesis. Der Andere Verlag, 2003. 211 p.

5. Yang S. Evolutionary Computation for Dynamic Optimization Problems // Tutorial, GECCO'13. 2013. Pp. 667–682.

6. Goh C.-K., Tan K. C. A competitive–cooperative coevolutionary paradigm for dynamic multiobjective optimization // *IEEE Transactions on Evolutionary Computation*. 2009. № 13(1). Pp. 109–127.

7. Иванов И., Сопов Е. Самоконфигурируемый генетический алгоритм решения задач поддержки многокритериального выбора // Вестник СибГАУ. 2013. № 1 (47). С. 30–35.

8. Иванов И., Сопов Е. Исследование эффективности самоконфигурируемого коэволюционного алгоритма решения сложных задач многокритериальной оптимизации // Системы управления и информационные технологии. 2013. № 1.1 (51). С. 141–146.

9. Ivanov I., Sopov E. Design Efficient Technologies for Context Image Analysis in Dialog HCI Using Self-Configuring Novelty Search Genetic Algorithm // In the 11th Intern. Conf. on Informatics in Control, Automation and Robotics, ICINCO'14. Vienna, Austria, 2014. Pp. 832–839.

10. Sopov E. A., Sopov S. A. The convergence prediction method for genetic and PBIL-like algorithms with binary representation // *Proceeding of IEEE International Siberian Conference on Control and Communications (SIBCON'11)*. Tomsk, 2011. Pp. 203–206.

11. Semenkin E. S., Semenkina M. E. Self-configuring Genetic Algorithm with Modified Uniform Crossover Operator. Advances in Swarm Intelligence // Lecture Notes in Computer Science 7331. Berlin Heidelberg : Springer-Verlag, 2012. Pp. 414–421.

12. Branke J., Schmeck H. Designing evolutionary algorithms for dynamic optimization problems. Theory and Application of Evolutionary Computation: Recent Trends. Springer-Verlag, 2002. Pp. 239–262.

13. Morrison R. W., De Jong K. A. A test problem generator for non-stationary environments // Proc. the 1999 Congr. on Evol. Comput. 1999. Pp. 2047–2053.

14. Benchmark Generator for CEC2009 Competition on Dynamic Optimization / C. Li [et al.] // Technical Report 2008. Department of Computer Science, University of Leicester, 2008. 14 p.

15. Branke J. Memory enhanced evolutionary algorithms for changing optimization problems // In Proc. the Congr. on Evol. Comput. 1999. Pp. 1875–1882.