UDC 519.6

Vestnik SibGAU
Vol. 16, No. 1, P. 113–118

HYBRIDIZATION OF LOCAL SEARCH WITH SELF-CONFIGURING GENETIC PROGRAMMING ALGORITHM FOR AUTOMATED FUZZY CLASSIFIER DESIGN

M. E. Semenkina

Siberian State Aerospace University named after academician M. F. Reshetnev
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660014, Russian Federation
E-mail: semenkina88@mail.ru

A fuzzy classifier is one of the intelligent information technologies allowing the generation of a fuzzy rule base suitable for interpretation by human experts. For a fuzzy classifier automated design the hybrid self-configuring evolutionary algorithm is proposed. The self-configuring genetic programming algorithm is suggested for the choice of effective fuzzy rule bases. For the tuning of linguistic variables the self-configuring genetic algorithm is used. A hybridization of self-configuring genetic programming algorithms (SelfCGPs) with a local search in the space of trees is fulfilled to improve their performance for fuzzy rule bases automated design. The local search is implemented with two neighborhood systems (1-level and 2-level neighborhoods), three strategies of a tree scanning ("full", "incomplete" and "truncated") and two ways of a movement between adjacent trees (transition by the first improvement and the steepest descent). The Lamarckian local search is applied on each generation to ten percent of best individuals. The performance of all developed memetic algorithms is estimated on a representative set of test problems of the functions approximation as well as on real-world classification problems. It is shown that developed memetic algorithm requires comparable amount of computational efforts but outperforms the original SelfCGP for the fuzzy rule bases automated design. The best variant of the local search always uses the steepest descent and full scanning for fuzzy classifier design. Additional advantage of the approach proposed is a possibility of the automated features selection. The numerical experiment results show the competitiveness of the approach proposed.

Keywords: genetic programming algorithm, self-configuration, fuzzy classifier, local search on discrete structures.

Вестник СибГАУ
Т. 16, № 1. С. 113–118

ГИБРИДИЗАЦИЯ ЛОКАЛЬНОГО СПУСКА С САМОКОНФИГУРИРУЕМЫМ АЛГОРИТМОМ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ АВТОМАТИЧЕСКОГО ГЕНЕРИРОВАНИЯ НЕЧЕТКИХ КЛАССИФИКАТОРОВ

М. Е. Семенкина

Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660014, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
E-mail: semenkina88@mail.ru

Нечеткие классификаторы являются одним из видов интеллектуальных информационных технологий, использующих базы нечетких правил, которые могут быть легко интерпретированы человеком-экспертом. При автоматическом создании нечетких классификаторов в данной работе используются самоконфигурируемый алгоритм генетического программирования для генерирования баз правил и самоконфигурируемый генетический алгоритм для настройки лингвистических переменных. Рассматриваются самоконфигурируемый алгоритм генетического программирования для автоматического генерирования баз правил для нечетких классификаторов и локальный спуск по деревьям, представляющим собой базы правил. Для локального спуска по дереву представлено два способа перехода между деревьями (переход по первому улучшению и наискорейший спуск), две системы окрестностей (1-соседняя и 2-соседняя окрестности) и три стратегии просмотра этих систем окрестностей («полный», «неполный» и «усеченный» просмотры). В ходе работы было выполнено сравнение эффективности всех вариантов выполнения локального спуска по дереву. Эффективность всех предложенных алгоритмов оценивалась на репрезентативном множестве тестовых задач и на двух реальных практических задачах классификации. По результатам тестирования можно сделать вывод, что локальный спуск, производящий полной просмотр 2-сосденей системы окрестностей, продемонстрировал лучшую эффективность и существенно повысил эффективность самоконфигурируемого алгоритма генетического программирования для автоматического генерирования нечетких классификаторов. Данный гибридный алгоритм почти всегда превосходит лучший для конкретной задачи вариант алгоритма генетического программирования, что позволяет полностью отказаться от выбора его наиболее эффективного варианта настроек. При *решении реальных задач анализа данных гибридный алгоритм продемонстрировал лучший результат среди всех рассмотренных альтернатив.*

Ключевые слова: алгоритм генетического программирования, самоконфигурация, нечеткий классификатор, локальный поиск на дискретных структурах.

Introduction. A fuzzy classifier [1] is one of the intelligent information technologies allowing the generation of a fuzzy rule base suitable for interpretation by human experts. It is an appropriate method for solving classification problems, which are a well-known application of natural computing algorithms.

However, the process of fuzzy classifier design and adjustment is rather complex even for experts in fuzzy systems. For automated implementation of the fuzzy classifier it is necessary to consider its design as an optimization problem. This problem is very complicated for standard optimization tools, which makes evolutionary algorithms rather popular in this field [2; 3]. A genetic programming algorithm (GP) can be used for the automated design of the fuzzy classifier rule base because of the ability of the GP to work with variable length chromosomes. The use of a GP can simplify the implementation of the Pittsburg [3] method and, in this case, there is no necessity to implement the Michigan [2] method for reducing the dimension of the search space.

It is well known that a GP requires a lot of effort in its adoption for any problem in hand. That is why before suggesting GP usage to end users, e. g., medicine or finance specialists for application in the development of classification tools, we must take care to avoid those main issues which are problems even for evolutionary computation experts. We have to suggest a way to avoid issues in the adjustment of the algorithm and the self-configuration for GP (Self-CGP) [4] could be a solution here.

It would be a good idea to use a local search for trees representing a fussy classifier rule base to improve the Self-CGP effectiveness. However, although the local search is often used for real valued and discrete optimization problems nevertheless it is not commonplace to use it for such a data structure as a tree. In this paper, we consider a new way for the local search on trees representing fuzzy rule bases.

Having conducted numerical experiments, we have found that the proposed approach positively impacts on the performance of the algorithm and deserves special attention and further investigation.

The rest of the paper is organized as follows. Section 1 explains the idea of self-configuring evolutionary algorithms. Section 2 describes the proposed method of fuzzy classifier automated design. Section 3 describes the idea of a local search for trees. Section 4 shows test results for the method proposed. Section 5 presents the results of the numerical experiments comparing the performance of the proposed approach and alternatives in solving real-world problems. In the Conclusion we discuss the results.

Self-configuring evolutionary algorithm. The self-configuring evolutionary algorithms (SelfCEA), which do not require for their adjustment any efforts of the end user as the method is adjusted automatically, use a dynamic adaptation on the population level [5] and centralized control techniques [6] for parameter settings with some differences from the usual approaches. Instead of tuning real parameters, variants of settings are used, namely types of selection (fitness proportional, rank-based, and tournament-based with three tournament sizes), crossover (one-point, two-point, as well as equiprobable, fitness proportional, rank-based, and tournament-based uniform crossovers [7]), population control and level of mutation (medium, low, high for all mutation types). Each of these has its own initial probability distribution, which is changed as the algorithm is executed [4].

This self-configuring technique can be used both for genetic algorithms (SelfCGA) and genetic programming (SelfCGP). In [8] the SelfCGA the performance was estimated on 14 test problems from [9]. As a commonly accepted benchmark for GP algorithms is still an "open issue" [10], the symbolic regression problem with 17 test functions borrowed from [9] was used in [7] for testing the self-configuring genetic programming algorithm. The statistical significance was estimated with ANOVA.

Analysing the results of SelfCGA [8] and SelfCGP [7] performance evaluation, we observed that the self-configuring evolutionary algorithms demonstrate better reliability than the average reliability of the corresponding single best algorithm. They can be used instead of conventional EA in complex problem solving.

Self-Configuring Evolutionary Algorithm for Automated Fuzzy Classifier Design. We have to describe our method of modelling and optimizing a rule base for a fuzzy logic system with GP and linguistic variables adjusted with a GA.

Usually, a GP algorithm works with a tree representation of solutions, defined by functional and terminal sets, and exploits specific solution transformation operators (such as selection, crossover, or mutation) until the termination condition is met [11]. The terminal set of our GP includes the terms of the output variable, i. e. class markers. The functional set includes a specific operation for dividing an input variables vector into sub-vectors or, in other words, for the separation of the examples set into parts according to input variable values. It might be that our GP algorithm will ignore some input variables and will not include them in the resulting tree, i. e., a high performance rules base that does not use all problem inputs can be designed. This feature of our approach allows the use of our GP for the selection of the most informative combination of problem inputs.

The tuning of linguistic variables is executed to evaluate the fuzzy system fitness that depends on its performance when solving the problem in hand, e. g., the number of misclassified instances. A linguistic variable consists of a set of terms or linguistic variable values representing some fuzzy concepts. Each of the terms is defined by a membership function. The tuning of linguistic variables consists in the optimization of membership function parameters simultaneously for all the terms of linguistic variables involved in problem solving. In this paper, we propose adjusting linguistic variables with the self-configuring genetic algorithm (SelfCGA) combined with the conjugate gradient method. We use here membership functions with a Gaussian shape. For the coding of membership function parameters, the mean value of the Gaussian function and its standard deviation are written consecutively for each term in the chromosome. For automatic control of the number of terms the possibility of ignoring a term is provided: all bits of the ignored term are set as 0 with the probability equal to 1/3.

The efficiency of the proposed approach has been tested on a representative set of known problems. The test results showed that the fuzzy classifiers designed with the suggested approach (SelfCGP-FL) [12] have a small number of rules in comparison with the full rule base. These fuzzy systems usually have a small enough classification error. This is why we can recommend the developed approach for solving real world problems.

Local search for discrete structures. We can formulate some general rules of neighbourhood system construction for trees that can be symbolic expressions, neural network models or fuzzy logic:

1. The tree is neighbouring to the original one if one element of the terminal set was replaced with another element.
2. The tree is neighbouring to the original one if one binary function from the functional set was replaced with another binary function.
3. The tree is neighbouring to the original one if one unary function from the functional set was replaced with another unary function.
4. Changes in the tree associated with functional elements generate larger changes in the phenotype than changes in the elements of the terminal set.

Trees with modified leaves (terminal set elements) will be called 1-level neighbours and trees with a modified functional element will be called 2-level neighbours. In our hybridization of the local search with self-configuring genetic programming for fuzzy logic classifier design, trees with a randomly replaced class type will belong to the 1-level neighbourhood and trees with a randomly replaced feature for vector dividing on sub-vectors will belong to the 2-level neighbourhood.

The search in such neighbourhoods for the locally best-found solution should improve the efficiency of the problem solving without a significant increase in computational efforts. However, the effectiveness of the local search depends not only on the choice of neighbourhood but also on the method of search.

There are several ways of movement between adjacent trees: transition by the first improvement and steepest descent that mean an exhaustive search of neighbouring trees. In this paper, we will use both ways of movement and both systems of neighbourhood. In the first case, the 2-level neighbourhood will be used at the beginning of the algorithm execution and the 1-level neighbourhood will be used on the later stages. We call this method of search a "full" local search. In the second case, only the 1-level neighbourhood will be used, this variant is named an "incomplete" local search. Changes in tree nodes that are closer to the top of the tree have a more significant impact on the result obtained. Therefore, when we use the 2-level neighbourhood, nodes which are closer to the top will be changed before others. This means that the tree will be considered from in the top-down way. Furthermore a "truncated" local search will be considered that means viewing only n randomly chosen nodes in the tree.

The local search procedure on the tree structure can be described as follow:

1. All tree nodes receive their numbers and order in which they will be considered: k_1, k_2, ..., k_n, where n is equal to the number of nodes which must be considered, k_i is the number of the i-th node. The order of considered nodes depends on the chosen strategy of neighbourhood searching ("full", "incomplete" or "truncated" local searches). Set i to be equal to 1.

2. The new value for the k_i-th node will be randomly chosen, which has to belong to the corresponding neighbourhood (the 1-level neighbourhood for leaves and the 2-level neighbourhood in other cases).

3. If i is not equal to n then go to the step 3.1, otherwise go to step 3.2.

3.1. If the fitness value for the modified tree is better than for the original tree then, in the case of transition by the first improvement, we save the modified tree, set up $i = i + 1$ and go to step 2, and, in the case of steepest descent, we save the fitness value and the modified tree and continue searching the original tree with $i = i + 1$ (go to step 2). Otherwise, if there is no fitness improvement, we continue searching the original tree with $i = i + 1$ (go to step 2).

3.2. In the case of transition by the first improvement, the local search procedure is finished. In the case of steepest descent, we substitute the original tree by the best one found and with this new tree go to step 1. If the new tree is equal to the original one than the local search procedure is finished.

During implementation and testing of the considered local search procedures the number of additional fitness function estimations must be taken into account. This number significantly depends on the way of the movement and the strategy for searching the neighbourhood. In addition, the speed of the hybrid algorithm depends on the selection of individuals to be improved by the local search (only the best individual or p % best in each generation, or once every t generations).

Experiment results for hybridization of local search with self-configuring genetic programming algorithm for automated fuzzy classifier design. For the test of the proposed hybrid algorithm, the same test function set was used as for the self-configuring genetic programming algorithm for automated fuzzy classifier generation. Since local search algorithms precisely identify the optimum position, the comparison of efficiency should be done with the criterion of reliability. The reliability of the algorithm is the ratio of the number of successful algorithm runs to the total number of algorithm runs. The algorithm run is considered as successful if the desired accuracy is achieved. Each algorithm received the same computing resources to find a solution and was launched 100 times for each test problem. The statistical significance was estimated with ANOVA. During testing the performance evaluation was performed for the self-configuring genetic programming algorithm hybridized with three types of local search (“full”, “incomplete” and “truncated”) and two strategies for movement (“first improvement” and “steepest descent”) that can be designated as “LS-1imp.+SelfCGP-FL-f”, “LS-1imp.+SelfCGP-FL-inc”, “LS-1imp.+SelfCGP-FL-t”, “LS-SD+SelfCGP-FL-f”, “LS-SD+SelfCGP-FL-inc” and “LS-SD+SelfCGP-FL-t” respectively.

The results obtained are presented in tab. 1.

Table 1

Algorithm reliability on test problems

Algorithm	Reliability	Average number of fitness function evaluation
SelfCGP+FL	0.64 [0.33, 0.96]	[4600, 21100]
LS-1imp+SelfCGP-FL-f	0.68 [0.37, 0.97]	[4340, 20500]
LS-1imp+SelfCGP-FL-inc	0.64 [0.34, 0.96]	[4150, 19800]
LS-1imp+SelfCGP-FL-t	0.65 [0.35, 0.97]	[4210, 20050]
LS-SD+SelfCGP-FL-f	0.72 [0.43, 0.99]	[4540, 21000]
LS-SD+SelfCGP-FL-inc	0.66 [0.38, 0.96]	[4380, 20650]
LS-SD+SelfCGP-FL-t	0.68 [0.39, 0.96]	[4500, 20800]

The follow criteria for evaluating the algorithms were selected:

1. Reliabilities that were averaged over all test problems and the spread of their values in brackets (“Reliability”).
2. Information on the number of resources required to find the first suitable solutions in terms of accuracy that were averaged over all tasks and in brackets the spread on all tasks (“Average number of fitness function evaluation”).

It is easy to see that the local search variant with a greater neighbourhood size and more detailed search though it has better reliability and a worse number of fitness function evaluations. The proposed local search algorithms increased the efficiency of the previously considered self-configuring genetic programming algorithms. With the joint application of self-configuring genetic programming and local search algorithms the performance is more often greater than of the best setting variant of genetic programming algorithms.

Numerical Experiments with Real World Problems. The developed approach was applied to two credit scoring problems from the UCI repository [13] often used to compare the accuracy with various classification models:

Credit (Australia-1) (14 attributes, 2 classes, 307 examples of creditworthy customers and 383 examples of non-creditworthy customers);

Credit (Germany) (20 attributes, 2 classes, 700 records of creditworthy customers and 300 records of non-creditworthy customers).

Both classification problems were solved with fuzzy classifiers designed by hybrid SelfCGP (SelfCGP-FL) hybridized with a different variant of the local search (LS). This technique was trained on 70 % of the instances from the data base and validated on the remaining 30 % of examples. The results of the validations (the portion of correctly classified instances from the test set) averaged for 40 independent runs are given in tab. 2 below. The statistical significance of all our experiments was estimated with ANOVA.

We first compared the fuzzy classifier performance with ANN-based [14; 15] and symbolic regression based [7] classifiers automatically designed by SelfCGP (SelfCGP+ANN and SelfCGP+SRF). As we have observed, the algorithm proposed in this paper demonstrates high performance on both classification tasks.

We then conducted the comparison of the proposed algorithms with alternative classification techniques. The results for the alternative approaches have been taken from scientific literature. In [16] the performance evaluation results for these two data sets are given for the authors' two-stage genetic programming algorithm (2SGP) specially designed for bank scoring as well as for the following approaches taken from other papers: conventional genetic programming (GP), multilayered perceptron (MLP), classification and regression tree (CART), C4.5 decision trees, *k* nearest neighbors (*k*-NN), and linear regression (LR). We have taken additional material for comparison from [17] which includes evaluation data for the authors' automatically designed fuzzy rule based classifier as well as for other approaches found in the literature: the Bayesian approach, boosting, bagging, the random subspace method (RSM), and cooperative coevolution ensemble learning (CCEL). The results obtained are given in tab. 2. As can be seen from tab. 2, the proposed algorithm demonstrates the best performance for both problems.

It is necessary to stress that fuzzy classifiers designed by SelfCGP hybridized with local search give additionally human interpreted linguistic rules which is not the case for the majority of other algorithms in tab. 2. Designed rule bases usually contain 10–15 rules which do not include all given inputs, i. e. are much easier to be interpreted by humans.

Table 2

The comparison of classification algorithms

Classifier	Australian credit	German credit
LS+SelfCGP-FL	0.9041	0.8021
2SGP	0.9027	0.8015
SelfCGP-FL	0.9022	0.7974
SelfCGP+ANN	0.9022	0.7954
SelfCGP+SRF	0.9022	0.7950
Fuzzy	0.8910	0.7940
C4.5	0.8986	0.7773
CART	0.8986	0.7618
k-NN	0.8744	0.7565
LR	0.8696	0.7837
RSM	0.8660	0.7460
Bagging	0.8470	0.6840
Bayesian	0.8470	0.6790
Boosting	0.7600	0.7000
CCEL	0.7150	0.7151

Analysis of the data sets shows that input variables can be divided into some groups so that the inputs of one group are highly correlated to each other but the correlation between inputs of different groups is weak. There are also inputs weakly correlated with the output. A fuzzy classifier designed with the suggested hybrid SelfCGP doesn't usually include inputs of the last kind. Moreover, it usually includes members of every group of inputs but only one input from each, i. e. it does not include highly correlated inputs in the rule base. This allows the algorithm to create relatively small rule bases with rather simple rules.

Conclusion. The self-configuring genetic programming algorithm and the local search were hybridized to design fuzzy classifiers with high efficiency. A special way of representing the solution provides the opportunity to create relatively small rule bases with rather simple rules. The quality of classification is high as well, which was demonstrated through the solving of two real world classification problems from the area of bank scoring.

The results obtained allow us to conclude that the developed approach is workable and useful and should be further investigated and expanded.

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. Ishibuchi H., Nakashima T., Murata T. Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems. *IEEE Trans. on Systems, Man, and Cybernetics,* 1999, vol. 29, p. 601–618.

2. Cordón O., Herrera F., Hoffmann F. and Magdalena L. Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases. *Singapore: World Scientific.* 2001.

3. Herrera F. Genetic Fuzzy Systems: Taxonomy, Current Research Trends and Prospects. *Evol. Intel.* 2008, vol. 1, no. 1, p. 27–46.

4. Semenkina M. E. [Effectiveness investigation of self-adaptive evolutionary algorithms for data mining information technology design]. *Iskusstvennyy intellekt i prinyatiye resheniy.* 2013, no. 1, p. 13–23 (In Russ.).

5. Meyer-Nieberg S., Beyer H.-G. Self-Adaptation in Evolutionary Algorithms. Lobo F. G., Lima C. F., Michalewicz Z. (eds.) *Parameter Setting in Evolutionary Algorithm,* 2007, vol. 54, p. 47–75.

6. Gomez J. Self Adaptation of Operator Rates in Evolutionary Algorithms. *Deb, K. et al. (eds.) GECCO 2004. LNCS,* 2004, vol. 3102, p. 1162–1173.

7. Semenkin E., Semenkina M. Self-Configuring Genetic Programming Algorithm with Modified Uniform Crossover Operator. *Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC),* 2012, p. 1918–1923.

8. Semenkin E. S., Semenkina M. E. Self-Configuring Genetic Algorithm with Modified Uniform Crossover Operator. *Advances in Swarm Intelligence, Lecture Notes in Computer Science,* vol. 7331. Springer-Verlag, Berlin Heidelberg, 2012, p. 414–421.

9. Finck S. et al. Real-Parameter Black-Box Optimization Benchmarking. *Presentation of the noiseless functions. Technical Report Researh Center PPE.* 2009.

10. O'Neill M., Vanneschi L., Gustafson S., Banzhaf W. Open Issues in Genetic Programming. *Genetic Programming and Evolvable Machines 11,* 2010, p. 339–363.

11. Poli R., Langdon W.B., McPhee N.F. A Field Guide to Genetic Programming. *Published via http://lulu.com.* 2008. Available at: http://www.gp-field-guide.org.uk.

12. Semenkina M., Semenkin E. Hybrid Self-configuring Evolutionary Algorithm for Automated Design of Fuzzy Classifier. *Advances in Swarm Intelligence. Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Hedelberg, 2014, vol. 8791, part 1, p. 310–317.

13. 13. Frank A., Asuncion A. UCI Machine Learning Repository (2010) *Irvine, CA: University of California, School of Information and Computer Science.* Available at: http://archive.ics.uci.edu/ml.

14. Semenkin E., Semenkina M. Artificial Neural Networks Design with Self-Configuring Genetic Programming Algorithm. Filipic B., Silc J. (Eds.) *Bio-inspired Optimization Methods and their Applications: Proceedings of the Fifth International conference BIOMA 2012,* 2012, p. 291–300.

15. Semenkin E. S., Semenkina M. E., Panfilov I. A. Neural Network Ensembles Design with Self-Configuring Genetic Programming Algorithm for Solving Computer Security Problems. *Computational Intelligence in Security for Information Systems, Advances in Intelligent Systems and Computing.* Springer-Verlag, Berlin Heidelberg, 2012, vol. 189, p. 25–32.

16. Huang J.-J., Tzeng G.-H., Ong Ch.-Sh. Two-Stage Genetic Programming (2SGP) for the Credit Scoring Model. *Applied Mathematics and Computation,* 2006, vol. 174, p. 1039–1053.

17. Sergienko R., Semenkin E., Bukhtoyarov V. Michigan and Pittsburgh Methods Combining for Fuzzy Classifier Generating with Coevolutionary Algorithm for Strategy Adaptation. *IEEE Congress on Evolutionary Computation*, IEEE Press, New Orleans, LA, 2011.

Библиографические ссылки

1. Ishibuchi H., Nakashima T., Murata T. Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems // Proc. of IEEE Trans. on Systems, Man, and Cybernetics. 1999. T. 29. P. 601–618.

2. Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases / O. Cordón [et al.]. Singapore : World Scientific. 2001.

3. Herrera F. Genetic Fuzzy Systems: Taxonomy, Current Research Trends and Prospects // Evol. Intel. 2008. Vol. 1, no. 1. P. 27–46.

4. Семенкина М. Е. Самоадаптивные эволюционные алгоритмы проектирования информационных технологий интеллектуального анализа данных // Искусственный интеллект и принятие решений. 2013. № 1. С. 13–23.

5. Meyer-Nieberg S., Beyer H.-G. Self-Adaptation in Evolutionary Algorithms // Parameter Setting in Evolutionary Algorithm / F. G. Lobo, C. F. Lima, Z. Michalewicz (eds.). 2007. Vol. 54. Pp. 47–75.

6. Gomez J. Self Adaptation of Operator Rates in Evolutionary Algorithms // Proc. of GECCO 2004. LNCS. 2004. Vol. 3102. Pp. 1162–1173.

7. Semenkin E., Semenkina M. Self-Configuring Genetic Programming Algorithm with Modified Uniform Crossover Operator // Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC). 2012. Pp. 1918–1923.

8. 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.

9. Real-Parameter Black-Box Optimization Benchmarking 2009. Presentation of the noiseless functions / S. Finck [et al.] // Technical Report Researh Center PPE. 2009.

10. Open Issues in Genetic Programming / M. O'Neill [et al.] // Genetic Programming and Evolvable Machines. 2010. № 11. Pp. 339–363.

11. Poli R., Langdon W. B., McPhee N. F. A Field Guide to Genetic Programming [Электронный ресурс] // Published via http://lulu.com. 2008. (With contributions by J. R. Koza). URL: http://www.gp-field-guide.org.uk.

12. Semenkina M., Semenkin E. Hybrid Self-configuring Evolutionary Algorithm for Automated Design of Fuzzy Classifier // Advances in Swarm Intelligence. Lecture Notes in Computer Science. 2014. Vol. 8791, Part 1. Pp. 310–317.

13. Frank A., Asuncion A. UCI Machine Learning Repository [Электронный ресурс]. Irvine, CA: University of California, School of Information and Computer Science, 2010. URL: http://archive.ics.uci.edu/ml.

14. Semenkin E., Semenkina M. Artificial Neural Networks Design with Self-Configuring Genetic Programming Algorithm // Bio-inspired Optimization Methods and their Applications : Proceedings of the Fifth Intern. Conf. BIOMA 2012 / B. Filipic, J. Silc (Eds.). 2012. Pp. 291–300.

15. Semenkin E. S., Semenkina M. E., Panfilov I. A. Neural Network Ensembles Design with Self-Configuring Genetic Programming Algorithm for Solving Computer Security Problems // Computational Intelligence in Security for Information Systems, Advances in Intelligent Systems and Computing 189. Berlin Heidelberg : Springer-Verlag, 2012. Pp. 25–32.

16. Huang J.-J., Tzeng G.-H., Ong Ch.-Sh. Two-Stage Genetic Programming (2SGP) for the Credit Scoring Model // Applied Mathematics and Computation. 2006. № 174. Pp. 1039–1053.

17. Sergienko R., Semenkin E., Bukhtoyarov V. Michigan and Pittsburgh Methods Combining for Fuzzy Classifier Generating with Coevolutionary Algorithm for Strategy Adaptation // Proc. of IEEE Congress on Evolutionary Computation. New Orleans, LA : IEEE Press, 2011.