Spacecraft Solar Arrays Degradation Forecasting with Evolutionary Designed ANN-based Predictors

Maria Semenkina, Shakhnaz Akhmedova, Eugene Semenkin and Ivan Ryzhikov
Institute of Computer Sciences and Telecommunication, Siberian State Aerospace University, Krasnoyarskiy Rabochiy ave., 31, Krasnoyarsk, 660014, Russia

Keywords: Spacecraft Solar Array, Degradation Forecasting, ANN-based Predictors, Automated Design, Self-configuring Evolutionary Algorithms, Bio-Inspired Algorithms Co-operation.

Abstract: The problem of forecasting the degradation of spacecraft solar arrays is considered. The application of ANN-based predictors is proposed and their automated design with self-adaptive evolutionary and bio-inspired algorithms is suggested. The adaptation of evolutionary algorithms is implemented on the base of the algorithms' self-configuration. The island model for the bio-inspired algorithms cooperation is used. The performance of four developed algorithms for automated design of ANN-based predictors is estimated on real-world data and the most perspective approach is determined.

1 INTRODUCTION

The future lies in the development of alternative energy sources such as solar arrays (SA). However, the efficiency of their application in systems of outer space assignment strongly depends on the environmental influence that results in their degradation. The testing of ground-based solar panels is an expensive and time-consuming procedure, therefore the use of forecast tools for the degradation of solar panels can significantly improve the process of SA implementation.

Artificial neural networks (ANN) are often used for prediction problems solving because of their generalization ability (Zhang, 1998). However, the efficiency of ANN-based predictors essentially depends on the choice of an effective ANN structure and the successful tuning of weight coefficients. Different types of evolutionary algorithms (EA) as well as so called bio-inspired approaches are often used for both tasks. In this paper we use proven Self-Configuring Evolutionary Algorithms (SelfCEA) (Semenkin, 2012) and Co-Operation of Biology Related Algorithms (COBRA) (Akhmedova, 2013) for the automated design of ANN-based predictors.

The rest of the paper is organized as follows: in Section 2 the problem description is given, in Section 3 we give some information on optimization techniques, in Section 4 approaches for the automated design of neural networks are described, in Section 5 we consider the outcomes of numerical experiments, and in the last section some conclusions and directions of further investigations are presented.

2 PROBLEM DESCRIPTION

It is necessary to develop a mathematical model for predicting solar array degradation according to available SA parameter changes measured together with the corresponding parameters of solar activity during the fulfilment of the real spacecraft's mission.

The model is adjusted to determine the electrical characteristics of solar panels based on the following environment factors:

- integral fluence of protons with energies less than 1 MeV;
- integral fluence of protons with energies less than 10 MeV;
- integral fluence of protons with energies less than 100 MeV;
- integral fluence of electrons with energies less than 0,6 MeV;
- integral fluence of electrons with energies less than 2 MeV;

Semenkina M., Akhmedova S., Semenkin E. and Ryzhikov I..
Spacecraft Solar Arrays Degradation Forecasting with Evolutionary Designed ANN-based Predictors.
DOI: 10.5220/0005122004210428
In *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2014), pages 421-428
ISBN: 978-989-758-039-0

- the parameter defined as the number of days since the spacecraft separation that characterizes the SA damage by meteoritic bodies and UV radiation;
- the parameter that characterizes the degree of spacecraft solar illumination.

We pose the following output parameters:

- open circuit voltage of the solar battery Ucv (open circuit voltage);
- current intensity of solar panel Isc (amperage short circuit)

for both sections of the spacecraft's SA, i.e. four output parameters are considered.

So it is necessary to design the SA degradation predictor based on the flight data. We will design it automatically in the form of an artificial neural network. For this aim, we will use the specific optimization approach that allows an automated choice of ANN structure and adjustment of weight coefficients.

3 SELF-ADAPTING OPTIMIZATION TECHNIQUES

Evolutionary and bio-inspired algorithms are well known and often used techniques for complicated optimization problem solving. However, their performance essentially depends on the choice of the algorithm settings and adjustment of parameters. It prevents the wide use of algorithms by end users who are not experts in stochastic optimization, e.g. aerospace engineers. Below we consider two approaches to the automated choice of the appropriate algorithmic scheme: co-evolution and self-configuration.

3.1 Co-Operation of Biology Related Algorithms (COBRA)

Five well-known optimization methods such as the Particle Swarm Optimization Algorithm (PSO) (Kennedy, 1995), the Wolf Pack Search Algorithm (WPS) (Yang, 2007), the Firefly Algorithm (FFA) (Yang, 2009), the Cuckoo Search Algorithm (CSA) (Yang, 2009) and the Bat Algorithm (BA) (Yang, 2010) are combined in one meta-heuristic called Co-Operation of Biology Related Algorithms (COBRA) (Akhmedova, 2013). These biology related optimization approaches work with continuous variables. It is impossible to say in advance which of the above-listed algorithms is the best one or which algorithm should be used for solving the given optimization problem (Akhmedova, 2013). This was the main reason for the development of a new meta-heuristic. At the same time these algorithms are very similar. The idea is the use of the cooperation of these algorithms instead of any attempts to decide which one is the best for the current problem in hand.

The following proposed approach is that five populations are generated (one population for each algorithm) which are then executed in parallel cooperating with each other. It is not required to choose the population size for each algorithm because the proposed algorithm is a self-tuning meta-heuristic. The number of individuals in each algorithm's population can increase or decrease depending on the increasing or decreasing of the fitness value. If the fitness value has not improved during a given number of generations, then the size of all populations increases. And vice versa, if the fitness value has constantly improved, then the size of all populations decreases. Besides, each population can "grow" by accepting individuals removed from other populations. A population "grows" only if its average fitness is better than the average fitness of all other populations. The result of this kind of competition allows us to provide the biggest resource (population size) to the most appropriate (in the current generation) algorithm. This property can be very useful in the case of a hard optimization problem when, as it is known, there is no single best algorithm at all stages of the optimization process execution (Eiben, 2003).

One of the most important driving forces of this meta-heuristic is the migration operator that creates a cooperation environment for component algorithms. All populations exchange individuals in such a way that a part of the worst individuals of each population is replaced by the best individuals of other populations. It brings up-to-date information on the best achievements to all component algorithms and prevents their preliminary convergence to its own local optimum that improves the group performance of all algorithms.

The performance of the proposed algorithm was evaluated on the set of benchmark problems from the CEC'2013 competition (Akhmedova, 2013). This set of benchmark functions (namely there were 28 unconstrained real-parameter optimization problems) was given in (Liang, 2012); there are also explanations about the conducted experiments. The validation of COBRA was carried out for functions with 10, 30 and 50 variables.

Experiments showed that COBRA works successfully and is reliable on this benchmark.

Results also showed that COBRA outperforms its component algorithms when the dimension grows and more complicated problems are solved (Akhmedova, 2013).

3.2 Binary Modification of COBRA

As was mentioned, all the algorithms listed above (PSO, WPS, FFA, CSA and BA) were originally developed for continuous valued spaces. However many applied problems are defined in discrete valued spaces where the domain of variables is finite. For this purpose the binary modification of COBRA (COBRA-b) was developed.

COBRA was adapted to search in binary spaces by applying a sigmoid transformation to the velocity component (PSO, BA) and coordinates (FFA, CSA, WPS) to squash them into a range [0, 1] and force the component values of the positions of the particles to be 0's or 1's.

The basic idea of this adaptation was taken from (Kennedy, 1997); firstly it was used for the PSO algorithm. It is known that in PSO each particle has a velocity (Kennedy, 1995), so the binarization of individuals is conducted by the use of the calculation value of the sigmoid function which is also given in (Kennedy, 1997):

$$s(v) = 1/(1+exp(–v)).$$

After that a random number from the range [0, 1] is generated and the corresponding component value of the particle's position is 1 if this random number is smaller than $s(v)$ and 0 otherwise.

In BA each bat also has a velocity (Yang, 2010), which is why we can apply exactly the same procedure for the binarization of this algorithm. But in WPS, FFA and CSA individuals have no velocities. For this reason, the sigmoid transformation is applied to position components of individuals and then a random number is compared with the obtained value.

Experiments with the same 28 test problems from (Liang, 2012) showed that the COBRA-b works successfully and reliably but slower than the original version of COBRA with a smaller success rate obtained (Akhmedova, 2013).

Such a result was expected as the binary modification needs more computing efforts in continuous variables space and should not be used instead of the original COBRA. However, it can be recommended for solving optimization problems with the binary representation of solutions.

3.3 Self-configuring Evolutionary Algorithm

If somebody decides to use evolutionary algorithms for solving real world optimization problems, it will be necessary to choose the effective variant of algorithm parameters such as the kind of selection, recombination and mutation operators. Choosing the right EA setting for each problem is a difficult task even for experts in the field of evolutionary computation. It is the main problem of effectively implementing evolutionary algorithms for end users. We can conclude that it is necessary to find the solution for the main problem of evolutionary algorithms before suggesting for end users any EAs application for the automated design of tools for solving real world problems.

We propose using the self-configuring evolutionary algorithms (SelfCEA) which do not need any end user efforts as the algorithm itself adjusts automatically to the given problem. In these algorithms (Semenkin, 2012), (Semenkin, 2012), the dynamic adaptation of operators' probabilistic rates on the level of population with centralized control techniques is applied (see Fig.1).

Instead of the adjusting real parameters, setting variants were 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 (Semenkin, 2012)), population control and level of mutation (medium, low, high for two mutation types). Each of these has its own initial probability distribution (see Fig. 2) which is changed as the algorithm executes (see Fig. 3).

This self-configuring technique can be used both for the genetic algorithm (SelfCGA) and for the genetic programming algorithm (SelfCGP). In (Semenkin, 2012) SelfCGA performance was estimated on 14 test problems from (Finck, 2009). As a commonly accepted benchmark for GP algorithms is still an "open issue" (O'Neill, 2010), the symbolic regression problem with 17 test functions borrowed from (Finck, 2009) was used in (Semenkin, 2012) for testing the self-configuring genetic programming algorithm. Statistical significance was estimated with ANOVA.

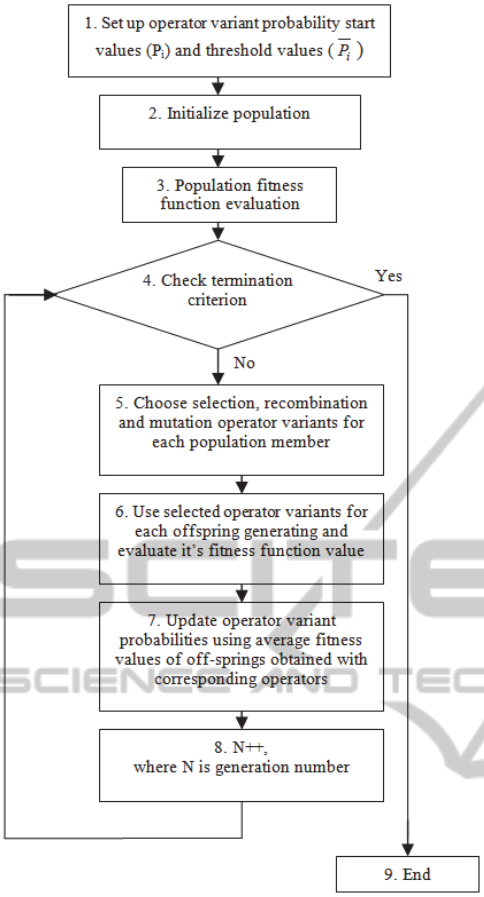

Figure 1: Main part of SelfCEA block diagram.

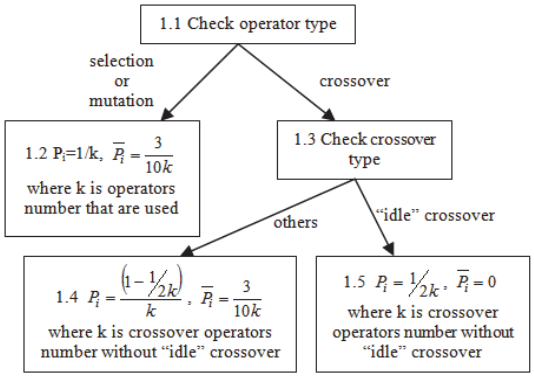

Figure 2: Flowchart illustrating step 1 in SelfCEA block diagram.

Analysing the results, related to SelfCGA (Semenkin, 2012) and SelfCGP (Semenkin, 2012), it can be seen that self-configuring evolutionary algorithms demonstrate higher reliability than the average reliability of the corresponding single best algorithm but sometimes worse than the best reliability of this algorithm.

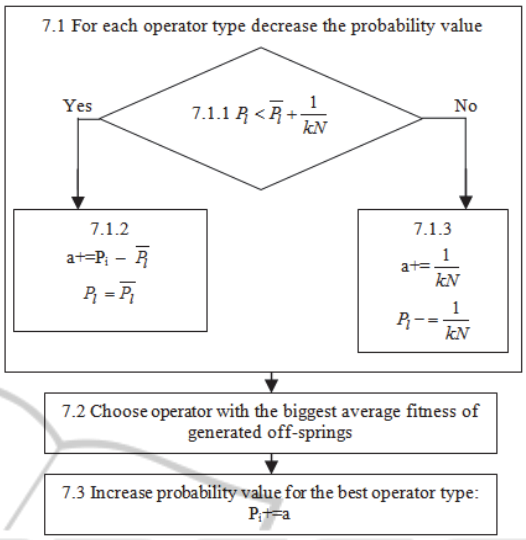

Figure 3: Flowchart illustrating step 7 in SelfCEA block diagram.

Both SelfCGA and SelfCGP can be used for the automated choice of effective structures and weights tuning of ANN-based predictors.

4 ANN AUTOMATED DESIGN

The appropriate structure of ANN must be chosen for the effective solving of the forecasting problem. Below we consider COBRA-b and SelfCGA for the choice of the number of layers, the number of neurons in each layer and the type of the activation function of each neuron for the multi-layered perceptron as well as SelfCGP for the automated design of the feed-forward ANN with an arbitrary structure.

4.1 ANN in Binary String

First of all we choose the perceptron with 5 hidden layers and 5 neurons in each hidden layer as a maximum size of the structure for ANN. Each node is represented by a binary string of length 4. If the string consists of zeros ("0000") then this node does not exist in ANN. So, the whole structure of the neural network is represented by a binary string of length 100 (25x4); each 20 variables represent one hidden layer. The number of input neurons depends on the problem in hand. ANN has one output layer.

We use 15 of the most common activation functions. For determining which activation function will be used on a given node, the integer that corresponds to its binary string is calculated.

Thus we use the optimization methods for problems with binary variables (COBRA-b, SelfCGA) for finding the best structure and the optimization method for problems with real-valued variables (COBRA, SelfCGA hybridized with local search) for the weight coefficients adjustment of each structure.

Although the ANNs structure automated design by self-adapting optimization technics improves their efficiency, it can work unsatisfactorily with big real-world problems. Therefore, the automation of the most important input selection can have a significant impact on the efficiency of neural networks. In this paper, we use additional bits in every string for the choice of relevant variables to put them in model. The number of these bits equals the number of input variables. If this bit is equal to '0' then the corresponding input variable is not used in the model and is removed from the sample. During initialization, the probability for a variable to be significant will be equal to 1/3. This idea can help end users to avoid the significant and complicated procedure of choosing the appropriate set of input variables with essential impact on the model performance.

For the choice of more flexible models more sophisticated tools must be used.

4.2 ANN Design with SelfCGP

We have to describe our way to model and optimize an ANN structure with genetic programming (GP) techniques before the employment of our SelfCGP algorithm.

Usually, the GP algorithm works with tree representation, defined by functional and terminal sets, and exploits specific solution transformation operators (selection, crossover, mutation, etc.) until the termination condition will be met (Poli, 2008).

The terminal set of our GP includes input neurons and 15 different activation functions. The functional set includes specific operations for neuron placement and connections. The first operation is the placing of a neuron or a group of neurons in one layer. There will be no appearance of additional connections in this case. The second operation is the placing of a neuron or a group of neurons in sequential layers in such a way that the neuron (group of neurons) from the left branch of the tree precedes the neuron (group of neurons) from the right branch of the tree. In this case, new connections will be added which connect the neurons from the left branch of the tree with the neurons from the right branch of the tree. Input neurons cannot receive any signal but have to send a signal to at least one hidden neuron. It might be so that our GP algorithm does not include some of the input neurons in the resulting tree, i.e., a high performance ANN structure that does not use all problem inputs can be found. This feature of the approach allows the use of GP for the selection of the most informative combination of problem inputs. The tree and corresponding neural network example are presented in Figure 4.

The GP algorithm forms the tree from which the ANN structure is derived. The ANN training is executed to evaluate its fitness which depends on its performance in solving the problem in hand, e.g., approximation precision or number of misclassified instances. For training this ANN, connection weights are optimized with the self-configuring genetic algorithm (SelfCGA) which does not need any end user efforts as the algorithm itself adjusts automatically to the given problem". When GP finishes giving the best found ANN structure as the result, this ANN is additionally trained with again the SelfCGA hybridized with a local search. The same approach is used for the application of SelfCGP.

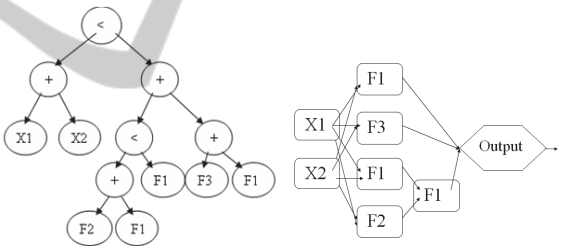

Figure 4: Tree and corresponding neural network example.

The efficiency of the proposed approach was tested on the representative set of test problems (approximation, time series prediction) (Panfilov, 2012). The test results showed that the neural networks created by SelfCGP have a small number of neurons in comparison with neural networks obtained by means of neuro-simulator and are not fully connected (few connections between neurons). Besides, the automatically designed ANNs did not include all inputs in model, i.e. reduced the input space.

5 EXPERIMENTAL RESULTS

We used evolutionary designed ANN-based predictors for the forecasting of the solar array

Table 1: Results for solar arrays degradation prediction.

Algorithm	1	2	3	4	Mean
SelfCGP+ANN	4,3196 (5,0442)	4,1441 (4,9392)	4,65 (5,53)	5,4863 (6,2066)	4,65 (5,43)
SelfCGP	4,6726 (5,7688)	4,4827 (5,5344)	5,03 (6,21)	5,9346 (7,3268)	5,03 (6,21)
SelfCGA+ANN	4,8584 (5,2672)	4,661 (5,0531)	5,232 (5,671)	6,17056 (6,6897)	5,23 (5,67)
COBRA+ANN+S	5,1127	2,8841	4,5755	8,6907	5,3158
COBRA+ANN	5,04907	4,77149	4,95868	6,62939	5,3522

degradation. The list of used tools is as follows:

1. Self-configuring genetic programming algorithm for the automated ANN design (SelfCGP+ANN) (Panfilov, 2012);
2. Self-configuring genetic algorithm for the automated ANN design and the significant variables choice (SelfCGA+ANN);
3. Co-Operation of Biology Related Algorithms for automated ANN structure search (COBRA+ANN+S) (Akhmedova, 2014);
4. Co-Operation of Biology Related Algorithms for adjusting weight coefficients in the single layer perceptron with 3 neurons in the hidden layer (COBRA+ANN).

Additionally, we used for the comparison some other approaches:

1. Symbolic regression based predictor designed with SelfCGP (Semenkin, 2012);
2. Fuzzy logic (FL) and neural-fuzzy logic (NFL) based predictors designed with GA (Shabalov, 2012);
3. ANN based predictor designed with Genetic Programming based Ensembling (GPEN) technique (Bukhtoyarov, 2012);
4. ANN based predictor designed with Neural Network Ensemble (GASEN) technique (Bukhtoyarov, 2012).

For all approaches, the prediction error was estimated in the same way:

$$error = \frac{100}{s}\frac{1}{m}\sum_{i}^{m}\frac{1}{y^i_{\max}-y^i_{\min}}\sum_{j}^{s}\left|y^i_j - o^i_j\right|,$$

where s is the test sample size, m is the outputs number, y is the true output value, and o is the model output.

The problem sample for the prediction of solar array degradation contains information about 295 days and has 7 inputs and 4 outputs. We used first 169 examples for training and the last part of the examples for testing our models. Results for all techniques presented in this paper are given in Table 1. The first number in each cell is the best found result among all algorithm runs. The mean error averaged over all runs is presented in brackets.

It is easy to see that SelfCGP for the ANN automated design exhibits the smallest among the best results; its mean result is better than the best results of other techniques that do not use the self-configuration. In particular, the relative error, that is equal to 4.319, is equivalent to the first output error that is equal to 0.1597V from a possible 3.6971V. Typically, SelfCGP and SelfCGA used only six inputs from seven and ignored the integral fluence of protons with the smallest energy.

The results of alternative approaches are presented in Table 2. Comparing this Table with Table 1 it can be seen that the approaches described in this paper essentially outperform alternatives that use fuzzy logic (NFL, FL) and are outperformed by ensembling methods, although self-configuring algorithms outperform one ensembling method as well (GASEN). The current best method (GPEN) to a large extent does not outperform self-configuring algorithms although it uses the ensembling technique. One can reasonably conclude that an ensembling technique based on the EA self-configuration could further improve the quality of the prediction.

Table 2: Best results for alternative methods.

Algorithm	GPEN	GASEN	NFL	FL
Mean	4,29	5,23	5,87	7,66

An additional observation is the differences in required computational efforts for different methods. The ensembling methods mentioned above have to configure more than 10 individual intellectual information technologies and generate formulae for their interaction. This is at least 10 times more time consuming than using one neural network. Certainly, the methods presented in this paper do not use any ensembling techniques and need much less computing time. At the same time, the best

algorithm among those presented in the paper (SelfCGP+ANN) requires 1,5 times more computational efforts although it uses the same number of fitness function evaluations. Nevertheless, SelfCGP+ANN should be used for the real-world SA degradation prediction as it has a much smaller error while the extra time spent is just some hours for computing that cannot be considered as a serious drawback in a process requiring many months of expensive experimentations.

6 CONCLUSIONS

In the paper four approaches to the automated design of ANN-based predictors for the degradation of spacecraft solar arrays were described and their performance estimation on real-world data was fulfilled. All these approaches differ from alternatives in the way they are adapted to the problem in hand. Namely, all these approaches are self-adapted and do not require from end users any expertise in computational intelligence (evolutionary computations, neural networks, etc.). The most perspective approach was determined, i.e. SelfCGP, although others also deserve further development. The evident way of approach improvement is the use of an ensembling technique although other directions should also be used, e.g. better implementation of self-adaptation techniques.

ACKNOWLEDGEMENTS

This research is supported by the Ministry of Education and Science of Russian Federation within State Assignment № 2.1889.2014/K.

REFERENCES

Akhmedova, Sh., Semenkin, E., 2013. New optimization metaheuristic based on co-operation of biology related algorithms. *Vestnik. Bulletine of Siberian State Aerospace University*, *Vol. 4 (50)*, 2013, pp. 92-99.

Akhmedova, Sh., Semenkin, E., 2013. Co-Operation of Biology Related Algorithms. *In: Proc. of the IEEE Congress on Evolutionary Computation (CEC 2013), Cancún*, *Mexico,* 2013, pp. 2207-2214.

Akhmedova, Sh., Semenkin, E., 2014. Co-Operation of Biology Related Algorithms Meta-Heuristic in ANN-Based Classifiers Design. *In: Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC)*, 2014. –accepted to publication.

Bukhtoyarov, V., Semenkin, E., Shabalov, A., 2012. Neural Networks Ensembles Approach for Simulation of Solar Arrays Degradation Process. *In Proc. of Hybrid artificial intelligent systems 7th International Conference, HAIS 2012, Salamanca, Spain,* 2012, pp. 186-195.

Eiben, A. E., Smith, J. E., 2003. *Introduction to Evolutionary Computing.* Springer Verlag, 2003, 299p.

Finck, S. et al., 2009. Real-parameter black-box optimization benchmarking 2009. *In: Presentation of the noiseless functions. Technical Report Researh Center PPE.*

Kennedy, J., Eberhart, R., 1995. Particle Swarm Optimization. *In Proc. of IEEE International Conference on Neural networks, IV, 1995,* pp. 1942–1948.

Kennedy, J., Eberhart, R.,1997. A discrete binary version of the particle swarm algorithm. *In Proc. of the World Multiconference on Systemics, Cybernetics and Informatics, Piscataway, NJ, 1997*, pp. 4104-4109.

Liang, J. J., Qu, B. Y., Suganthan, P. N., Hernandez-Diaz, A. G, 2013. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization*. In Technical Report 2012, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China, and Technical Report, Nanyang Technological University, Singapore.

O'Neill, M., Vanneschi, L., Gustafson, S., Banzhaf, W., 2010. Open issues in genetic programming. *In: Genetic Programming and Evolvable Machines 11*, 2010, pp. 339–363.

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

Poli, R., Langdon, W. B., McPhee, N. F., 2008. *A Field Guide to Genetic Programming.* Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. (With contributions by J. R. Koza).

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

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

Shabalov, A., Semenkin, E., Galushin, P., 2012. Integration of Intelligent Information Technologies Ensembles for Modeling and Classification. *In Proc. of Hybrid artificial intelligent systems 7th*

International Conference, HAIS 2012, Salamanca, Spain, 2012, pp. 365-374.

Yang, C., Tu, X., Chen, J., 2007. Algorithm of Marriage in Honey Bees Optimization Based on the Wolf Pack Search. *In Proc. of the International Conference on Intelligent Pervasive Computing, 2007*, pp. 462-467.

Yang, X. S., 2009. Firefly algorithms for multimodal optimization. *In Proc. of the 5th Symposium on Stochastic Algorithms, Foundations and Applications, 2009*, pp. 169–178.

Yang, X. S., Deb, S., 2009. Cuckoo Search via Levy flights. *In Proc. of the World Congress on Nature & Biologically Inspired Computing, IEEE Publications, 2009,* pp. 210-214.

Yang, X. S., 2010. A new metaheuristic bat-inspired algorithm. *In Nature Inspired Cooperative Strategies for Optimization, Studies in Computational Intelligence,* Vol. 284, 2010, pp. 65-74.

Zhang G., Patuwo B. E., Hu M. Y., 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting 14*, 1998, pp. 35–62.