И. С. Рыжиков

## АВТОМАТИЧЕСКАЯ ИДЕНТИФИКАЦИЯ ЛИНЕЙНЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В АНАЛИТИЧЕСКОМ ВИДЕ

*Рассматривается сведение задачи идентификация линейных динамических систем к задаче поиска глобального оптимума. Рассматривается подход, который позволяет автоматически определять структуру и параметры линейного дифференциального уравнения через решение оптимизационной задачи с помощью модифицированного гибридного метода эволюционных стратегий. Располагая априорной информацией такой, как вектор начальных состояний системы или его оценка, выборка измерений выхода динамической системы и входная характеристика.*

*Ключевые слова: эволюционные стратегии, идентификация, структура и параметры, дифференциальное уравнение.*



UDK 519.8

E. S. Semenkin, M. E. Semenkina

## INTEGRATION OF INTELLIGENT INFORMATION TECHNOLOGIES ENSEMBLES WITH SELF-CONFIGURING GENETIC PROGRAMMING ALGORITHM*

*Self-configuring genetic programming algorithm with the modified uniform crossover operator, that realizes a selective pressure on the recombination stage, is used for the automated integration of the computational intelligence technique ensembles. Ensemble members are the symbolic regression formulas, the artificial neural networks or their combination. They are also designed automatically with the self-configuring genetic programming algorithm. The comparative analysis of the approach performance is given on the benchmark and real world problems.*

*Keywords: Genetic programming, self-configuration, neural networks, symbolic regression, ensembles, automated design, classification problems.*

For many real world problems we can observe the following situation. There is a big data base of the results of the complex system behavior observations but appropriate model of this system is not yet clear. Here we can use intelligent information technologies (IIT) to obtain the first stage model within short time in order to simulate the system and learn its properties that gives us a possibility to develop a full profile model of the system. However, the design of IIT can also be a problem.

Currently, intelligent systems have got wide propagation in various fields of human activity connected with complex systems modeling and optimization. Artificial neural networks [25], fuzzy logic [31], symbolic regression [18], evolutionary algorithms [8] and other techniques and technologies are the popular tools for the system investigation due to their capability to solve complex intelligent problems that are difficult for the classic techniques [17].

The highly increasing computing power and technology made possible the use of more complex intelligent architectures, taking advantage of more than one intelligent technique in a collaborative way. This is an effective combination of intelligent techniques that outperform or compete to simple standard intelligent techniques.

One of the hybridization forms, the ensemble technique, has been applied in many real world problems. It has been observed that the diversity of members, making up a committee, plays an important role in the ensemble approach [5].

Different techniques have been proposed for maintaining the diversity among members by running on the different feature sets [14] or training sets (e. g. bagging [1] and boosting [11]).

Some techniques, such as neural networks, can be run on the same feature and training sets producing the diversity by different structures [20]. Simple averaging, weighted averaging, majority voting, and ranking are common methods usually applied to calculate the ensemble output.

Johansson et al. [16] used genetic programming (GP) for building an ensemble from the predefined number of the artificial neural networks (ANN) where functional set consisted of the averaging and multiplying and the terminal set included the models and constants. In [2], a similar approach was proposed where first a specified number of the neural networks is generated and then a genetic programming algorithms applied to build an ensemble making up symbolic regression from partial decisions of the specific members.

*The study was supported by The Ministry of education and science of Russian Federation, project № 16.740.11.0742, 14.740.12.1341 and 11.519.11.4002.

In this paper we apply the self-configuring genetic programming technique to construct formula that shows how to compute an ensemble decision using the component IIT decisions. The algorithm involves different operations and math functions and uses the models of different kinds providing the diversity among the ensemble members. Namely, we use symbolic expressions and neural networks, automatically designed with our GP algorithm, as the ensemble members. The algorithm automatically chooses component IIT which are important for obtaining an efficient solution and doesn't use the others.

The rest of the paper is organized as follows. Section 2 explains the idea of selective pressure during the stage of crossover in GP and briefly describes the result of the algorithms performance investigation. Section 3 describes the method for the GP self-configuring and its testing results that confirm the method usefulness. Section 4 describes the method of the ANN automated design and its performance evaluation. In Section 5 we describe the GP-based approach to the IIT ensembles automated integration and the results of the performance comparative analysis. In Section 6 we apply developed approach to one problem from the speech recognition area. In Conclusion section we discuss the results and directions of the further research.

**Uniform crossover operator with selective pressure for genetic programming algorithm.** The uniform crossover operator is known as one of the most effective crossover operators in conventional genetic algorithm [30, 4]. Moreover, nearly from beginning, it was suggested to use a parameterized uniform crossover operator and it was proved that tuning this parameter (the probability for a parental gene to be included in the off-spring chromosome) can essentially improve "The Virtues" of this operator [4]. Nevertheless, in the majority of cases using uniform crossover the mentioned possibility is not adopted and the probability for a parental gene to be included in an off-spring chromosome is given equal to 0.5 [8, 13].

That is why it seemed interesting to us to modify the uniform crossover operator with a purpose of improving its performance. Having a desire to avoid real number parameter tuning, we introduced selective pressure during process of recombination [27] making the probability of a parental gene being passed to an off-spring depended on parent fitness values. Like the usual EA selection operators, fitness proportional, rank-based and tournament-based uniform crossover operators were added to the conventional operator which we now call the equiprobable uniform crossover.

As about genetic programming, we use a tree representation in our GP. To apply the crossover operator with selective pressure in GP, that was useful in GA [27], we first have to specify the uniform crossover operator for GP. Such an operator has been theoretically substantiated and introduced by Poli and Langdon in 1998 [22, 23]. See, also, the recent description in [24].

According to [23], the GP uniform crossover process starts at the tree's root node and works its way down each tree along some path until finding function nodes of differing arity at the similar location. Interior nodes are selected for crossover with some probability which is generally set to 0.5. Crossover involves exchanging the selected nodes between the trees, while those nodes not selected for crossover remain unaffected.

We organize the uniform crossover in slightly different way. After parents selection, the uniform crossover operator produces one off-spring choosing nodes from parental trees with the given probability. Crossing over also has a place in the situation when parental nodes contain functions with different arity because all arguments compete with each other. The described modification brings more flexibility to the crossover process and allows the potential for a change in the algorithms behavior.

Having the appropriate uniform crossover operator we can introduce now a selective pressure in GP in the way described above. The off-spring can inherit every of its nodes from one of parents not only equiprobably but also with different probabilities determined by parent fitness values in one of the ways mentioned above: fitness proportionally, according to ranks or through tournament.

We have implemented the described approach and conducted numerical experiments to evaluate the usefulness of the developed operator. As a commonly accepted benchmark for GP algorithms is still an "open issue" [21], we used the symbolic regression problem with test functions usually used for the evaluation of evolutionary optimization algorithms. Having here no place to go into the detail, we just summarize results given in [28]. On this benchmark, the GP algorithm with our uniform crossover operator outperforms the conventional GP algorithm in the average reliability, i.e. proportion of 100 runs when approximation with sufficient precision was found, and its variance (0.53 vs. 0.43 and [0.31, 0.88] vs. [0.13, 0.77]) as well as in the solution quality, i.e. the percentage of the symbolically identical formulas found (78 % vs. 66 %). Testing with real world problems also confirmed the approach perceptiveness.

**Operator Rates based Self-Configuration of Algorithms.** Although EAs have been successful in solving many optimization and modeling problems, the performance of this technique depends on the selection of the EA settings and parameters tuning.Moreover, the process of settings determination and tuning parameters is known to be a time-consuming and a complicated task. Much research has attempted to deal with this problem. Some approaches attempted to determine appropriate settings by experimenting over a set of well-defined functions or through theoretical analysis. Other approaches, usually applying a term such as "self-adaptation", try to eliminate the setting determination process by adapting settings through the algorithm execution.

There exists much research devoted to "self-adapted" EA based on a range of ideas, but all of them aimed at reducing the role of human expert in algorithm designing. Let us follow definitions given by Gabriela Ochoa and Marc Schoenauer, organizers of the workshop "Self-tuning, self-configuring and self-generating evolutionary algorithms" (Self* EAs) within PPSN XI [26]. According to this definition, " The following 3 general paths toward

automated heuristic design will be distinguished: 1) tuning: the process of adjusting the control parameters of the algorithm; 2) configuring: the process of selecting and using existing algorithmic components (such as variation operators); 3) generating: the process of creating new heuristics from existing sub-components borrowed from other search methods". As the main ideas of the algorithms discussed here rely on automated "selecting and using existing algorithmic components", these algorithms might be called self-configuring. Within this, the probabilities with which will be the genetic operators used are subject to the tuning. This allows us to say that the algorithms are partially self-tuning.

In order to specify our algorithms more precisely, one can say that according to the classification [9], we use dynamic adaptation on the population level [19]. The probabilities of genetic operators to be applied are modified "on the fly" as the algorithm executes. According to the classification given in [12] we use centralized control techniques (central learning rule) for parameters settings with some differences from the usual approaches. Operator rates (the probability to be chosen for generating off-spring) are adapted according to the relative success of the operator during the last generation independently of the previous results. That is why our algorithm avoids problem of high memory consumption typical for centralized control techniques [12]. In some cases it results in situation where no operators have success (off-spring fitness improvement) but nevertheless some of them (worked badly but better than others) will be rewarded. Operators' rates are not included in individual chromosomes and they are not subject to the evolutionary process. All operators can be used during one generation for producing off-springs one by one.

As mentioned above, we apply operators probabilistic rates dynamic adaptation on the level of population with centralized control techniques. To avoid the issues of precision caused while using real parameters, we used setting variants, namely types of selection, crossover, population control and level of mutation (medium, low, high). Each of these has its own probability distribution. E.g., there are 5 settings of selection fitness proportional, rank-based, and tournament-based with three tournament sizes. During initialization all probabilities are equal to 0.2 and they will be changed according to a special rule through the algorithms execution in such a way that a sum of probabilities should be equal to 1 and no probability could be less than predetermined minimum balance. The "idle crossover" is included in the list of crossover operators to make crossover probabilities less than 1 as is used in conventional algorithms to model a "childless couple".

When the algorithm creates the next off-spring from the current population it first has to configure settings, i.e. to form the list of operators with using the probability operator distributions. The algorithm then selects parents with the chosen selection operator, produces an off-spring with the chosen crossover operator, mutates offspring with the chosen mutation probability and puts off-spring into the intermediate population. When the intermediate population is filled, the fitness evaluation is computed and the operator rates (probabilities to be chosen) are updated according to operator productivities. Then the next parent population is formed with the chosen survival selection operator. The algorithm stops after a given number of generations or if the termination criterion (e.g., given error minimum) is met.

The productivity of an operator is the ratio of the average off-spring fitness obtained with this operator and the average fitness of the overall off-spring population. Successful operators having productivity more than 1 increase their rates obtaining portions from unsuccessful operators.

The described approach can be used both in GA and GP as well as in other EA techniques. In this paper, we used self-configuring GA (SelfCGA) for tuning parameters of the symbolic regression formulas and training ANN weights.

We have solved the same test symbolic regression problems with the proposed self-configuring GP (SelfCGP) and demonstrated competitive results with conventional GP (average reliability is 0.69, variance is [0.42, 1], and 74 % of the symbolically identical solutions). Then we have solved two hard classification problems (German Credit and Australian-1 Credit from [10]) and compare our results with alternative approaches found in scientific literature.

We concluded that the SelfCGP can produce competitive results; it has the usual drawbacks of any general-purpose technique losing to the problem specific algorithms on corresponding problems, but has the advantage requiring no algorithmic details adjustment.

**ANN automated design with self-configuring genetic programming algorithm.** Usually, the GP algorithm works with tree representation, defined by functional and terminal sets, and exploit the specific solution transformation operators (selection, crossover, mutation, etc.) until termination condition will be met [24].

For the ANN automated design, the terminal set of our GP includes 16 activation functions such as bipolar sigmoid, unipolar sigmoid, Gaussian, threshold function, linear function, etc. The functional set includes specific operation for neuron placement and connections. The first function is the placing a neuron or a group of neurons in one layer. There will no additional connections appeared in this case. The second function is the placing 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 tree preceded by the neuron (group of neurons) from the right branch of tree. In this case, new connections will be added that connect the neurons from the left tree's branch with the neurons from the right tree's branch. Input neurons cannot receive any signal but have to send a signal to at least one hidden neuron.

The GP algorithm forms the tree from which the ANN structure is derived. The ANN training is executed to evaluate its fitness that depends on its performance in solving problem in hand, e.g., approximation precision or number of misclassified instances. For training this ANN,

connection weights are optimized with selfconfiguring genetic algorithm (SelfCGA) that does not need any end user efforts to be the problem adjusted doing it automatically. When GP finishes giving the best found ANN structure as the result, this ANN is additionally trained with again Self-CGA hybridized with local search.

We compared the performance of the ANNs designed with our SelfCGP algorithm with the alternative methods on the set of problems from [10]. Materials for the comparison we have taken from [32] where together with results of authors' algorithm (CROANN) the results of 15 other approaches are presented on three classification problems (Iris,Wisconsin Breast Cancer, Pima Indian Diabetes) from [10].

Analyzing comparison results, we observed that the performance of the approach suggested in this paper is high enough comparing alternative algorithms (1*st* , 3rd and 4*th* positions, correspondingly). However, the main benefit from our SelfCGP algorithm is the possibility to be used by the end user without expert knowledge in ANN modeling and evolutionary algorithm application. Additional dividend is the size of designed ANNs. The ANNs designed with SelfCGP contain few hidden neurons and connections and use not all given inputs although perform well. It can be used for the discrimination of the unimportant inputs that could also give additional information for experts.

Now we can conclude that the self-configuring genetic programming algorithm is the suitable tool for ANN automated design. It can also produce the competitive results based on symbolic regression approach. This algorithm makes the end user free from the ANN structure design and the GP settings determination. We can use it for the design of IIT ensembles.

5 Integration of IIT ensembles with self-configuring genetic programming algorithm

Having the developed appropriate tool for IIT automated design that does not require the effort for its adjustment, we applied our self-configuring genetic programming technique to construct formula that shows how to compute an ensemble decision using the component IIT decisions. The algorithm involves different operations and math functions and uses the models of different kinds providing the diversity among the ensemble members. In our numerical experiments, we use symbolic expressions and neural networks, automatically designed with our SelfCGP algorithm, as the ensemble members. The algorithm automatically chooses the component IIT which are important for obtaining an efficient solution and doesnt use the others. The ensemble component IIT are taken from the preliminary IIT pool that includes 20 ANNs and 20 symbolic regression formulas (SRFs) generated before handwith Self-CGP. For the designing every IIT, corresponding data set was randomly divided into two parts, i.e., training sample (70%) and validation sample (30%).

The first experiment was conducted for comparing the performance of the ensembling method based on the SelfCGP with the others. We used the same three problems from [10] and the real world problem of the simulation of the spacecraft solar arrays degradation (SAD) process [3] as the test bed. In Table 1 below we used following abbreviations: ANNE is the ANN ensemble, SRFE is the symbolic regression formulas ensemble, ANN+SRF is the integration of ANN and SRF, s.a. is the simple averaging, and w.a. is the weighted averaging. The last two rows contain the performance of non-ensembling models based on SelfCGP. Numbers in the first three columns are the error measure calculated as it was given in [32] and the numbers in the last column are deviations from the correct value.

Results in Table 1 demonstrate that the SelfCGP based ensembling methods used ANNs or ANNs and SRFs integration outperform other approaches. Although, GPNS+GPEN approach [2] for SAD problem demonstrates the performance 0.0430 [28]. The statistical robustness of the results obtained was confirmed by ANOVA tests which were used for processing received evaluation of the performance.

Within the second numerical experiment we solved two hard classification problems and compared our results with alternative approaches.

The first data set, called the German Credit Data Set, includes customer credit scoring data with 20 features, such as age, gender, marital status, credit history records, job, account, loan purpose, other personal information, etc. The second data set includes Australian credit scoring data and contains 14 attributes, where six are continuous attributes and eight are categorical attributes.

Both data sets are made public from the UCI Repository of Machine Learning Databases [10], and are often used to compare the accuracy with various classification models.

Results for alternative approaches have been taken from scientific literature. In [15] the performance evaluation results for these two data sets are given for authors' two-stage genetic programming algorithm(2SGP) as well as for the following approaches taken from other papers: conventional genetic programming (GP+SRF), classification and regression tree (CART), C4.5 decision trees, k nearest neighbors (k-NN), linear regression (LR). Additional material for comparison we have taken from [29] where is evaluation data for authors' automatically designed fuzzy rule based classifier as well as for other approaches found in literature: Bayesian approach, boosting, bagging, random subspace method (RSM), cooperative coevolution ensemble learning (CCEL).

The results of the comparison (proportion of the correctly classified objects in the validation set) are given in Table 2.

As one can see, the ensembles automatically designed with the SelfCGP outperform other ensembles (Boosting, Bagging, CCEL) and single classifiers including these specially implemented for bank scoring problems solving (Fuzzy, 2SGP).

*Table 1*

**Ensembling methods comparison**

| Classifier | Iris | Cancer | Diabetes | SAD |
|---|---|---|---|---|
| SelfCGP ANNE | 0 | 0 | 17.18 | 0.0418 |
| SelfCGP SRFE | 0.0133 | 0.34 | 18.21 | 0.0548 |
| SelfCGP ANN+SRFE | 0 | 0.06 | 17.43 | 0.0465 |
| ANN s.a. | 0.0267 | 1.09 | 19.75 | 0.0542 |
| ANN w.a. | 0.0267 | 1.03 | 19.03 | 0.0503 |
| SRF s.a. | 0.0533 | 1.27 | 20.23 | 0.0628 |
| SRF w.a. | 0.04 | 1.22 | 19.86 | 0.0605 |
| ANN+SRF s.a. | 0.04 | 1.18 | 19.79 | 0.0617 |
| ANN+SRF w.a. | 0.0267 | 1.09 | 19.34 | 0.0556 |
| SelfCGP SRF | 0.0267 | 1.23 | 20.01 | 0.0621 |
| SelfCGP ANN | 0.0133 | 1.05 | 19.69 | 0.0543 |

*Table 2*

**Classification methods comparison**

| Classifier | Australian credit | German credit | Classifier | Australian credit | German credit |
|---|---|---|---|---|---|
| SelfCGP ANN+SRFE | 0.9094 | 0.8126 | Fuzzy | 0.8910 | 0.7940 |
| SelfCGP ANNE | 0.9046 | 0.8075 | 2SGP | 0.9027 | 0.8015 |
| SelfCGP SRFE | 0.9046 | 0.8050 | GP+SRF | 0.8889 | 0.7834 |
| SelfCGP+SRF | 0.9022 | 0.7950 | LR | 0.8696 | 0.7837 |
| SelfCGP+ANN | 0.9022 | 0.7954 | Bayesian | 0.8470 | 0.6790 |
| GP+ANN | 0.8969 | 0.7863 | RSM | 0.8660 | 0.7460 |
| Boosting | 0.7600 | 0.7000 | k-NN | 0.8744 | 0.7565 |
| Bagging | 0.8470 | 0.6840 | CART | 0.8986 | 0.7618 |
| CCEL | 0.7150 | 0.7151 | C4.5 | 0.8986 | 0.7773 |

Certainly, the last two methods give not only calculation formula but also the production rules that might be more important for end user. It means that we have to focus our further research on this direction.

**SelfCGP applications in speech recognition.** Successful application of our approach in the area of classification brought us to the idea of adapting to the most complex problems such as speech recognition. We have chosen the ISOLET problem ([10]) for the first attempt to check our approach in this field due to its relative simplicity and the availability of the data set and other approaches known results for comparison.

ISOLET problem is the recognition problem of English letters pronounced by 150 different speakers those spoke the name of each letter of the alphabet twice. The features include spectral coefficients; contour features, sonorant features, presonorant features, and post-sonorant features. Exact order of appearance of the features is not known. It gives the data set with 617 attributes (all of them are continuous, realvalued attributes scaled into the range –1.0 to 1.0), 26 classes, and 7797 instances.

Having in mind the necessity to verify the ensembles, we have randomly divided the data set in three parts 4679 instances for single ANNs training, 1559 instances for single ANNs testing and 1559 instances for the ensembles cross-validation. Ensembles training was executed on the first 6238 instances. Both ANN-based classifiers and their ensembles were automatically designed with SelfCGP algorithm. Preliminary pool of classifiers consisted of 10 members although usually ensembles involved from 2–3 till 7 classifiers.

Alternative approaches for performance comparison have been taken from [6, 7, 10]. In table 3 bellow OPT means conjugate-gradient implementation of back propagation, C4.5 means Quinlan's C4.5 system, OPC means one-per-class representation, ECOC means error-correcting output code, raw means unpruned decision trees, pruned means pruned decision trees (CF = 0.25), hard means default trees, soft means trees with softened thresholds, multiclass means one tree to do all 26-way classifications [7, 6], SelfCGP+ANN means single best ANN-based classifier automatically generated with our self-configuring genetic programming algorithm, SelfCGP+ANN+Ens means the ensemble of ANN-based classifiers automatically designed with SelfCGP.

As one can see from Table 3, both our approaches demonstrate competitive results (2*nd* and 8*th* position of 39).

ANN-based classifier automatically generated with SelfCGP can be considered as enough powerful tool for speech recognition problems but our ensembling technique can essentially improve the classifier performance making it to be one of the best.

The SelfCGP based automatic designing the ensembles of heterogeneous IIT allows improving the reliability and effectiveness of data analysis. The obtained results are approved by solving some real-world problems.

*Table 3*

**Performance comparison for ISOLET problem**

| Algorithms and their configurations | % errors | % correct |
|---|---|---|
| OPT 30-bit ECOC | 3.27 | 96.73 |
| SelfCGP+ANN+Ens | 3.40 | 96.60 |
| OPT 62-bit ECOC | 4.04 | 95.96 |
| OPT OPC | 4.17 | 95.83 |
| C4.5 107-bit ECOC soft pruned | 6.61 | 93.39 |
| C4.5 92-bit ECOC soft pruned | 6.86 | 93.14 |
| C4.5 45-bit ECOC soft pruned | 6.99 | 93.01 |
| SelfCGP+ANN | 7.21 | 92.79 |
| C4.5 107-bit ECOC soft raw | 7.44 | 92.56 |
| C4.5 92-bit ECOC soft raw | 7.57 | 92.43 |
| C4.5 107-bit ECOC hard pruned | 8.08 | 91.91 |
| C4.5 92-bit ECOC hard pruned | 8.15 | 91.85 |
| C4.5 62-bit ECOC soft pruned | 8.40 | 91.60 |
| C4.5 30-bit ECOC soft pruned | 8.60 | 91.40 |
| C4.5 62-bit ECOC soft raw | 8.60 | 91.40 |
| C4.5 77-bit ECOC hard pruned | 8.85 | 91.15 |
| C4.5 45-bit ECOC soft raw | 9.30 | 90.70 |
| C4.5 62-bit ECOC hard pruned | 9.88 | 90.12 |
| C4.5 45-bit ECOC hard pruned | 9.94 | 90.06 |
| C4.5 30-bit ECOC soft raw | 11.23 | 88.77 |
| C4.5 30-bit ECOC hard pruned | 11.87 | 88.13 |
| C4.5 multiclass soft pruned | 15.33 | 84.67 |
| C4.5 multiclass soft raw | 15.91 | 84.09 |
| C4.5 multiclass hard pruned | 16.29 | 83.71 |
| C4.5 15-bit ECOC soft pruned | 16.61 | 83.39 |
| C4.5 multiclass hard raw | 16.93 | 83.07 |
| C4.5 OPC soft pruned | 18.99 | 81.01 |
| C4.5 15-bit ECOC soft raw | 20.59 | 79.41 |
| C4.5 107-bit ECOC hard raw | 21.42 | 78.58 |
| C4.5 92-bit ECOC hard raw | 22.39 | 77.61 |
| C4.5 OPC soft raw | 24.31 | 75.69 |
| C4.5 15-bit ECOC hard pruned | 24.57 | 75.43 |
| C4.5 77-bit ECOC hard raw | 27.20 | 72.80 |
| C4.5 OPC hard pruned | 28.03 | 71.97 |
| C4.5 62-bit ECOC hard raw | 29.70 | 70.30 |
| C4.5 OPC hard raw | 33.29 | 66.71 |
| C4.5 45-bit ECOC hard raw | 36.43 | 63.57 |
| C4.5 30-bit ECOC hard raw | 43.04 | 56.96 |
| C4.5 15-bit ECOC hard raw | 63.57 | 36.43 |

Certainly, the computational efforts for the implementation of the described approach and the model complexity are severely increasing comparing to each single learning model. However it is usual drawback of any ensembling when one has to implement many members of ensemble. There are no additional problems with our approach here. Advantages of the ensembling are better performance and reliability that essentially compensates extra computational efforts. In fact, really additional computational effort for our approach is the necessity to run the genetic programming algorithm that combines single models outputs into an output of the ensemble. Our experiments showed that it is less than the efforts for evolutionary generating one single model, i.e., could be considered as acceptable disadvantage.

As about the model complexity, again our approach does not bring much extra drawback comparing with any other ensemble technique. Of course, a computational model given by the genetic programming algorithm might be much more complicated compared to the usual ensembling methods such as weighted sum of outputs or voting. However, our experiments show that the genetic programming algorithm never includes all possible single models into an ensemble taking usually a few of them. As the greater part of the ensemble computational complexity is given by the computational efforts needed for calculating the output for each model, our approach has the advantage upon usual ensembling methods that include in the ensemble all available single models.

The further development of the system is aimed to the expansion of its functionality by including the other types of IITs (fuzzy logic systems, decision trees, neuro-fuzzy systems, other kinds of ANNs, multiobjective selection, etc.).

## References

1. Breiman, L. Bagging predictors, Machine Learning. 1996. Vol. 24 (2). P. 123–140.

2. Bukhtoyarov V., Semenkina O. Comprehensive evolutionary approach for neural network ensemble automatic design // Proceedings of the IEEE World Congress on Computational Intelligence, Barcelona, Spain, 2010. P. 1640–1645.

3. Bukhtoyarov V., Semenkin E., Shabalov A. Neural Networks Ensembles Approach for Simulation of Solar Arrays Degradation Process // Hybrid Artificial Intelligent Systems. Lecture Notes in Computer Science, 2012. Vol. 7208. P. 186–195.

4. De Jong K. A., Spears W. On the Virtues of Parameterized Uniform Crossover // Proceedings of the 4th Intern. Conf. on Genetic Algorithms, Morgan Kaufmann. July, 1991.

5. Dietterich T. G. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization // Machine Learning. 2000. Vol. 40, № 2. P. 139–158.

6. Dietterich T. G., Bakiri G. Error-correctingoutput codes: A general method for improving multiclass inductive learning programs // Proceedings of AAAI-91. 1991.

7. Dietterich T. G., Bakiri G. Solving multiclass learning problems via error-correcting output codes // J. of Artificial Intelligence Research 2. 1995. P. 263–286.

8. Eiben A. E., Smith, J. E. Introduction to evolutionary computing. Springer, Berlin, 2003 // IEEE Congress on Evolutionary Computation. 2011. New Orleans, LA.

9. Eiben A. E., Hinterding R., Michalewicz Z. Parameter control in evolutionary algorithms // IEEE Transactions on Evolutionary Computation. 1999. Vol. 3(2). P. 124–141.

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

11. Friedman J. H., Hastie T., Tibshirani, R. Additive logistic regression: a statistical view of boosting, Annals of Statistics. 2000. Vol. 28, № 2, P. 337–374.

12. Gomez J. Self Adaptation of Operator Rates in Evolutionary Algorithms // Deb K. et al. (Eds.) GECCO 2004. LNCS 3102. 2004. P. 1162–1173.

13. Haupt R. L., Haupt S. E. Practical genetic algorithms. John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.

14. Ho T. K., Hull J. J., Srihari S. N. Decision combination in multiple classifier systems // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1994. Vol. 16, № 1. P. 66–75.

15. 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. P. 1039–1053.

16. Johansson U., Lofstrom T., Konig R., Niklasson L. Building Neural Network Ensembles using Genetic Programming // Intern. Joint Conf. on Neural Networks, 2006.

17. Konar A. Computational Intelligence: Principles, techniques and applications. Springer, Berlin, 2005.

18. Kronberger G. Symbolic Regression for Knowledge Discovery: Bloat, Overfitting, and Variable Interaction Networks. Trauner Verlag, 2011.

19. Meyer-Nieberg S., Beyer H.-G.: Self-Adaptation in Evolutionary Algorithms. In F. Lobo, C. Lima, Z. Michalewicz (Eds.) Parameter Setting in Evolutionary Algorithm. 2007. P. 47–75.

20. Navone H. D., Granitto P. M., Verdes P. F., Ceccatto H. A. A learning algorithm for neural network ensembles // Inteligencia Artificial, Revista Iberoame-ricana de Inteligencia Artificial. № 12. 2001. P. 70–74.

21. O'Neill M., Vanneschi L., Gustafson S., Banzhaf W. Open issues in genetic programming // Genetic Programming and Evolvable Machines. 2010. P. 339–363.

22. Poli R., Langdon W. B. On the Ability to Search the Space of Programs of Standard, One-Point and Uniform Crossover in Genetic Programming. Technical Report CSRP-98-7. The University of Birmingham (UK), 1998.

23. Poli R., Langdon W. B. On the search properties of different crossover operators in genetic programming // Genetic Programming 1998. Proc. of the 3rd Annual Conf. Wisconsin, USA, 22–25 July. 1998. P. 293–301.

24. Poli R., Langdon W. B., McPhee N. F. A Field Guide to Genetic Programming. – Published via. URL: http://lulu.comandfreelyavailableat; http://www.gp-field-guide.org.uk. 2008.

25. Rojas, R. Neural networks: a systematic introduction. Springer, Berlin, 1996.

26. Schaefer R., Cotta C., Kolodziej J., Rudolph G. Parallel Problem Solving from Nature // PPSN XI 11th Intern. Conf. (Eds.). Krakow, Poland, September 11–15, 2010.

27. Semenkin E. S., Semenkina M. E. Application of GA with modified uniform recombination operator for automated implementation of intellectual information technologies // Vestnik. Scientific Journal of Siberian State Aerospace University named after academician M. F. Reshetnev. 2007. Iss. 3 (16). P. 27–32.

28. Semenkin E., Semenkina M.: Self-Configuring Genetic Programming Algorithm with Modified Uniform Crossover Operator // Proceedings of the IEEE Congress on Evolutionary Computation. June 10–15, 2012.

29. 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. New Orleans, LA., 2011.

30. Syswerda G. Uniform crossover in genetic algorithms // Proceedings of the 3rd Intern. Conf. on Genetic Algorithms. Morgan Kaufmann, 1989.

31. Yager R. R., Filev D. P. Essentials of fuzzy modelling and control. Wiley, New York, 1994.

32. Yu J. J. Q., Lam A. Y. S., Li V. O. K. Evolutionary Artificial Neural Network Based on Chemical Reaction Optimization // IEEE Congress on Evolutionary Computation. 2011. New Orleans, LA., 2011.

Е. С. Семенкин, М. Е. Семенкина

## ПРОЕКТИРОВАНИЕ АНСАМБЛЕЙ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ САМОКОНФИГУРИРУЕМЫМ АЛГОРИТМОМ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

*Самоконфигурируемый алгоритм генетического программирования с модифицированным оператором равномерного скрещивания, использующим селективное давление на этапе рекомбинации, применяется для автоматического формирования ансамблей интеллектуальных информационных технологий. Символьные выражения, нейронные сети или их комбинации являются членами ансамблей, которые генерируются также автоматически при помощи самоконфигурируемого алгоритма генетического программирования. Сравнительный анализ эффективности подхода был проведен на тестовых и реальных практических задачах.*

*Ключевые слова: генетическое программирование, само-конфигурация, нейронные сети, символьная регрессия, ансамбли, автоматическое проектирование, задачи классификации.*



UDK 519.68

O. E. Semenkina

## EFFECTIVENESS COMPARISON OF ANT COLONY AND GENETIC ALGORITHMS FOR SOLVING COMBINATORIAL OPTIMIZATION PROBLEMS*

*This paper considers ant colony optimization, genetic algorithm and parallel versions of these methods for solving traveling salesman problem.*

*Keywords: Genetic algorithm, Ant colony optimization, Traveling Salesman Problem, Parallelization.*

One of the most interesting and topical methods for optimization problems solving are stochastic algorithms working with many current solutions at the same time. This paper considers ant colony optimization (ACO) [1] and genetic algorithm (GA) [2] and also parallel versions of these methods. These algorithms can be very useful for finding approximate solutions of complex optimization problems. Both algorithms are nature-inspired optimization metaheuristics, where ACO is based on the behavior and organization of ant colonies in their search for food whereas GA is based on some principals of evolution.

Investigation of effectiveness of ACO and GA was conducted by solving well-known problem of combinatorial optimization namely Traveling Salesman Problem (TSP). Software implementation of these algorithms was programmed in C++ and the library MPICH2 was used for parallelization.

In the genetic algorithm for the TSP a chromosome is represented as permutation of the n numbers (numbers of cities). That is why some standard operations have a few changes, but many adjustable parameters in GA remains such as mutation probability, the type of selection – tournament selection (parameter is the size of the tournament), proportional and rank selection (linear or exponential ranking), population size and number of generations. Solutions in ACO are also represented as permutation of n cities and ants chose next city using taboo-list and pheromone matrix at every stage. The same to GA, ACO have rather adjustable parameters: ant colony size ($m$), number of iteration, evaporation rate ($\rho$), relatively importance of previous search experience ($\alpha$) and relatively importance of the distance between cities ($\beta$).

At first efficiency of each algorithm on a test task was investigated (a grid 5 on 5 cities) and regularities in dependence of algorithms efficiency on its settings were revealed. ACO algorithm shows the best results at the following settings: $\alpha = 1$, $\beta = 10$, $\rho \in [0.5, 0.7]$ and quantity of ants approximately equals to number of the cities. The best settings for GA are tournament selection with small tournament size and low or very low mutation rate. This test task solution ACO showed better results than the GA because ACO rely on distance between the cities at initialization and begins to work with rather a quite good route while GA begins with an absolutely random population. Therefore, ACO requires considerably less computational resources on the problem with a lot of decisions. Therefore reliability of algorithms was investigated on more complex test task with a number of the cities equals to 30 and called Oliver30. After the completion of algorithms work local search was applied to the best solutions. For both algorithms identical number of calculations, namely 30 individuals (ants) and 10000 generations (iterations) was given.

*The study was supported by The Ministry of education and science of Russian Federation, project № 16.740.11.0742, 14.740.12.1341 and 11.519.11.4002.