EMPIRICAL STUDY OF SELF-CONFIGURING GENETIC PROGRAMMING ALGORITHM PERFORMANCE AND BEHAVIOUR*

E. Semenkin, M. Semenkina
Siberian State Aerospace University named after Academician M.F. Reshetnev
31 “KrasnoyarskiyRabochiy” prospect, Krasnoyarsk, 660014, Russia.

E-mail: eugenesemenkin@yandex.ru

Abstract. The behaviour of the self-configuring genetic programming algorithm with a modified uniform crossover operator that implements a selective pressure on the recombination stage, is studied over symbolic programming problems. The operator’s probabilistic rates interplay is studied and the role of operator variants on algorithm performance is investigated. Algorithm modifications based on the results of investigations are suggested. The performance improvement of the algorithm is demonstrated by the comparative analysis of suggested algorithms on the benchmark and real world problems.

Introduction

Genetic programming (GP) [1] algorithms are information processing techniques based on the principles of natural evolution. Although GPs have been successfully used in solving many real world problems, the performance of this technique essentially depends on the selection of its settings and tuning parameters. The process of settings determination and parameters tuning is known to be a time-consuming and complicated task. Much research is devoted to this problem. Some of approaches are oriented to the elimination of the setting determination process by adapting settings through the algorithm execution. This research devoted to "self-adapted" GP is based on a range of ideas aimed at reducing the human expert role in algorithm designing.

Following [2] we call algorithms investigated in this study self-configuring as the main idea of their automated design is "selecting and using existing algorithmic components" (see the definition given by G. Ochoa and M. Schoenauer, organizers of the workshop "Self-tuning, self-configuring and self-generating evolutionary algorithms" (Self* EAs) within PPSN XI [3]).

This self-configuration is based on GP operator’s probabilistic rates that are adapting themselves through the algorithm run. The operator’s probability to be chosen for generating off-spring is adapted according to the relative success of this operator during the last generation. All operators are included in the operators’ pool before algorithm’s run and all of them can be used during one generation for producing offspring one by one. Operator’s rates are not included in individual chromosomes and they are not subject to the evolutionary process.

* Results were obtained in the framework of the state task № 140 of the Ministry of Education and Science of the Russian Federation.
Numerical experiments conducted in [2] have demonstrated that self-configuration of GP positively impacts on the algorithm performance. However, in that study algorithm’s self-configuring wasn’t only novelty in GP design. There were also new types of uniform crossover operators that introduced a selective pressure on the stage of recombination. All our algorithms, GP with modified uniform crossover (MGP) and self-configuring GP (SelfCGP), are used for symbolic regression problem being hybridized with self-configuring genetic algorithm (SelfCGA, [4]) as the tool for numerical coefficients adjustment. Although the usefulness of both novelties (self-configuration and new uniform crossover) was demonstrated nevertheless the role of each of them in the GP performance improvement is not yet clear.

In this paper, we study the operator’s probabilistic rates interplay through the algorithm execution and suggest the algorithm modification that improves its performance.

The rest of the paper is organized in following way. Section 2 describes the proposed method for GP self-configuring as well as testing results over the benchmark symbolic regression problems. In Section 3 we investigate the algorithm behaviour, in Section 4 we suggest the algorithm modifications and describe results of numerical experiments, Section 5 contains the performance comparison of suggested algorithms and alternative approaches on two real world classification problems. In the Conclusion we discuss obtained results and directions of the future research.

Operators’ Rates Based Self-Configuration of GP Algorithms

In our algorithm [2] operators’ probabilistic rates dynamic adaptation on the level of population with centralized control techniques was applied (see Fig.1). Instead of the real parameters adjusting, 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 [2]), 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) that is changed as algorithm executes (see Fig. 3).

![Fig.1. Main cycle of SelfCGP.](image-url)
Fig. 2. Flowchart illustrating starting point of SelfCEA.

Check operator type

- Check crossover type
  - Crossover
    - "idle" crossover
  - Others

\[ P_i = \frac{1}{k}, \quad \bar{P}_i = \frac{3}{10k} \]

where \( k \) is operators number that are used

\[ P_i = \frac{\left( \frac{1}{2} \right)^k}{k}, \quad \bar{P}_i = \frac{3}{10k} \]

where \( k \) is crossover operators number without "idle" crossover

Fig. 3. Flowchart illustrating the way to update operator variant probabilities.

For each operator type decrease the probability value

- Yes
  - \[ P_i < \bar{P}_i + \frac{1}{kN} \]
    - \( a += P_i - \bar{P}_i \)
    - \( P_i = \bar{P}_i \)
  - No
    - \( a += \frac{1}{kN} \)
    - \( P_i = \frac{1}{kN} \)

Choose operator with the biggest average fitness of generated off-springs

Increase probability value for the best operator type:

\[ P_i += a \]
The uniform crossover operator in evolutionary algorithms is known as one of the most effective crossover operators [5, 6]. For this reason, the uniform crossover operator for GP was modified in [2] with the purpose of improving its performance. Modification gives the possibility to fulfil uniform crossover also in case when nodes have different arity because all the arguments of these functions compete with each other. E.g., if one tree is "COS – X" (cos(x)) and other one is "/ – Y – Z" (y/z) and the off-spring inherits the node "/" then the resulting subtree can be one of "y/z", "x/z" or "y/x". If the offspring inherits the node "COS", then the resulting subtrees can be "cos(x)", "cos(y)", "cos(z)". This modification brings more flexibility to the crossover process and adds the potential for a change in the behaviour of the algorithm.

Selective pressure during the recombination process was introduced making the probability of a parental gene being passed to the offspring depend on parent fitness values [7]. The offspring can inherit each of its nodes from one of its parents not only equiprobably, but also with different probabilities determined by parent fitness values in one of the ways as it is made in the usual selection operators. Fitness proportional, rank-based and tournament-based uniform crossover operators were added to the conventional operator that can be called now the equiprobable uniform crossover.

As a commonly accepted benchmark for GP algorithms is still an "open issue" [8], the symbolic regression problem with 17 test functions borrowed from [9] were used in [2] for testing the modified genetic programming algorithm with new uniform crossover operators (MGP).

The results of self-configuring GP (SelfCGP) performance evaluation over 17 test problems and the performance comparison with conventional GP are shown in Table 1 below ([2]). The experiments settings are 100 individuals, 300 generations and 100 algorithm runs for each test function. The variance is given over 17 test functions. The reliability is the portion of the runs for a given algorithm that give a satisfactorially precise solution (MSE is less than 0.01). The statistical significance was estimated with ANOVA.

From Table 1 we can see that GP with modified uniform crossover operators (MGP) performs better than conventional GP. This demonstrates that the proposed operators may be useful.

SelfCGP reliability averaged over 17 test functions is better than the averaged best reliability of conventional GP and slightly less than best reliability of modified GP. The worse reliability (for the hardest problem) averaged over 100 runs is equal to 0.42. The best reliability is equal to 1.00. Computational efforts are less than alternative algorithms have. It gives us the possibility to recommend SelfCGP for solving symbolic regression problems as better alternative to the conventional GP. The main advantage of SelfCGP is that there is no need of algorithmic details adjustment without any losses in the performance. This makes this algorithm useful for many applications where end users, being no experts in evolutionary modelling, nevertheless intend to apply GP for solving these problems.
Table 1. Self-configuring GP performance over test symbolic regression problems

| Algorithms          | Reliability | Average generations variance | % precise solutions | % conditionally precise solutions | % approx. solutions |
|---------------------|-------------|------------------------------|---------------------|-----------------------------------|--------------------|
| GP                  | 0.43        | [0.00, 0.91]                 | [33, 289]           | 50                                | 16                 |
| MGP                 | 0.53        | [0.11, 1.00]                 | [27, 243]           | 58                                | 20                 |
| SelfCGP             | 0.69        | [0.42, 1.00]                 | [49, 201]           | 58                                | 16                 |
| SelfCGP+Cconv       | 0.46        | [0.19, 0.83]                 | [35, 254]           | 50                                | 17                 |
| SelfCGP+UC          | 0.67        | [0.41, 1.00]                 | [29, 219]           | 66                                | 16                 |

Here SelfCGP uses both conventional GP crossover operators and uniform crossover operators with selective pressure. However, it is interesting to consider separately the performance of new uniform crossover operators (SelfCGP+UC) as well as the performance of conventional GP crossover operators (SelfCGP+Conv) within SelfCGP. It could give us knowledge about the contribution of different operator types involved in SelfCGP to the performance of the algorithm.

As we can see from Table 1, the self-configuration with conventional crossover operators is just slightly better than the conventional GP itself but worse than MGP (without self-configuration at all) and essentially worse than self-configuring GPs contained uniform crossover operators with selective pressure. We can conclude that self-configuration is a useful idea but its main advantage is not the performance improvement but the automated choice of settings.

It is interesting to observe that the SelfCGP+UC performance is slightly inferior than the performance of complete SelfCGP, i.e. the presence of conventional crossover operators helps new uniform crossover operators.

As far as the solving of symbolic regression problems is concerned, it is interesting to have not only a sufficiently precise computational procedure but also a symbolically correct answer, i.e. exactly the same analytical expression that was used to generate the database. The last three columns in Table 1 contain information on the quality of the obtained approximations. The first column shows the percentage of precise solutions symbolically identical to the test function. The second one shows the percentage of conditionally precise solutions that required some elementary transformations and rounding of numbers to be symbolically identical to the test function. The third column shows the percentage of the obtained solutions which cannot be transformed into a symbolically identical form (but give a sufficiently good approximation).

We can see that the self-configuration itself (without new uniform crossover operators) does not improve this ability of the conventional GP. Such a role of uniform crossover operators in GP can be explained through the observation that these operators prevent GP
tree’s blow up which usually complicates the symbolic expression and provides no possibility to find a precise solution.

**Self-Configuring Genetic Programming Behaviour Analysis**

Let us look inside the workings of the algorithm and try to illustrate the interaction of operators over the execution of the algorithm. For this we used the graph showing the distribution of operator probabilities. Studying this graph we can try to understand which is the most efficient operator, whether it is possible and useful to reject some “unnecessary” kinds of operators, etc.

In all the figures below, the abscissa axes represent a number of generations (from 0 to 130) and the ordinate axes depict the probabilities of operator’s employment.

In Figure 4 below we can see the typical interplay of the mutation operators (1-point and tree-based mutations each with 3 levels) averaged over 20 SelfCGP runs. This interplay is very sophisticated; operators interact actively over the algorithm execution, changing each other. It is clear that there are no “unnecessary” kinds of mutation operators as each of them plays an important role in a particular stage of the algorithm’s run. We cannot reject any kind of operator.

![Fig.4. Mutation’s probabilistic rates interplay.](image)

Another form of the operators’ interaction is shown in Figure 5 which depicts the typical interplay of the selection operators. Usually, they actively compete with each other on the first 25-35 generations but then they split into two groups. Two or three operators reduce their rates fast and will never recover, but two or three others reach high rates and do not decrease until the end of the run. The problem here is that we cannot say in advance which operator will win. One of tournament selection operators is usually among the winners; the rank based selection is often among the winners; the fitness proportional selection wins rarely, but all of them can be important for future problem solving.
Crossover operators demonstrate completely different interplay (see Figure 6). There is no hard competition among them. From the early beginnings of the run, it becomes clear which operator is the most important; this operator reaches a higher and higher rate and usually no other operator can change this situation. Winners are usually rank based and equiprobable uniform crossovers. Two operators always lose; they are the standard crossover and proportional uniform ones.

It is also interesting to consider separately the interplay of new uniform crossover operators within SelfCGP (SelfCGP+UC) as well as the interaction of conventional GP crossover operators (SelfCGP+Conv.).

SelfCGP+UC behaviour is depicted in Figure 7 where we can see the active competition without an evident loser.
The typical behaviour of SelfCGP+Conv. is shown in Figure 8 where we see no competition at all. The 1-point crossover operator outperforms the standard crossover operator during all benchmark problems.

The analysis of operators interplay within SelfCGP based on the observations of changes in operators’ probabilistic rates shows that there are no “unnecessary operators” among mutations and selections. Each of them can be useful in different examples of the given problem. But this is not the case for crossover operators. On the other hand, our observations allow us to infer the limitation of conventional standard crossover and fitness proportional uniform crossover that never won over all the test problems of symbolic regression used in this study. These two kinds of crossover seem to be “unnecessary” and this brings us to an idea to modify SelfCGP, rejecting two probably useless operator variants making the algorithm less complicated and, hopefully, more effective.
Performance Comparison of Algorithms Over Test Symbolic Regression Problems

The above observations have brought us to the idea to reject “unnecessary” operators and to design the SelfCGP algorithm with a combination of remaining operators. We call this special case of algorithm SelfCGP+SC. One of the variants of different ways of algorithm’s behaviour on different problems is depicted in Figure 9 below.

We can see that there is strong competition among different operators and the superiority of one of them is not entirely unconditional. Additionally, every operator can be a winner on corresponding test problems.

Fig.9. Probabilistic rates interplay for special combination of crossover operators

For an investigation of this new variant of SelfCGP, we use the same 17 benchmark symbolic regression problems that we used above. The results of the algorithm performance and solution quality evaluations are presented in Table 2.

Table 2. Modified algorithm performance and solution quality over test symbolic regression problems

| Algorithm        | SelfCGP+SC |
|------------------|------------|
| Reliability      | 0.71       |
|                  | [0.44, 1.00] |
| Average generations variance | 31, 197 |
| % precise solutions | 59 |
| % cond. precise solutions | 18 |
| % approx. solutions | 23 |

As we can see, the SelfCGP+SC that includes the best crossover operators and rejects the two worst ones demonstrates the best performance and the second best decision quality (Tables 1, 2). It means that our investigation of the operators’ interaction within SelfCGP yielded a useful result.

One more interesting observation here is that the SelfCGP+UC that has a worse performance than SelfCGP+SC nevertheless has a much higher ability in finding the
symbolically correct answer (Tables 1, 2). We suppose that the reason for this is the conventional 1-point crossover that is included in SelfCGP+SC. This operator can produce offspring that is much more complicated than its parents. This can result in the GP tree blow up or, at least, can complicate the tree more than is necessary to find a symbolically precise solution and gives no possibility of reducing the expression on the next generation. At the same time, a more complicated tree can give an expression with less error.

We can recommend using SelfCGP+UC in applied symbolic regression problem solving if the symbolically precise or simple solution is more important than some decrease of numerical error. In the opposite situation it would be better to use SelfCGP+SC.

**Performance Comparison of Algorithms on Classification Problems**

We have studied the behaviour of different SelfCGP variants on some test symbolic regression problems and observed competitive results with conventional GP. Now we will solve two hard classification problems and compare our results with alternative approaches. Classification problems will be solved by the construction of a separating surface by means of symbolic regression formulation.

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 and other personal information. There are 700 records judged to be creditworthy and 300 records judged to be non-creditworthy. The second data set includes Australian credit scoring data with 307 examples of creditworthy customers and 383 examples of non-creditworthy customers. It contains 14 attributes, where six are continuous attributes and eight are categorical attributes. Both data sets have been made public by the UCI Repository of Machine Learning Databases [10], and are often used to compare the accuracy with various classification models. There are many reports on results achieved on these problems that make them useful for comparison.

For our experiments we used 100 individuals and a maximum of 500 generations over 20 runs. For each run the data sets were randomly divided into training and validation subsets with proportion 70% - 30%. The results were averaged. The SelfCGP fitness function is here a proportion of correctly classified instances from the validation subset.

The results for alternative approaches have been taken from scientific literature. In [11] 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), classification and regression tree (CART), C4.5 decision trees, linear regression (LR). We have taken additional material for comparison from [12] where there is evaluation data for authors' automatically designed fuzzy rule based classifier as well as for other approaches found in literature: random subspace method (RSM), cooperative coevolution ensemble learning (CCEL).

The results of the performance comparison (proportion of correctly classified instances from the validation subset) of all the above mentioned classifiers with the four variants of SelfCGP are presented in Table 3 below. SelfCGPs results are averaged over 20 runs.

We can see that SelfCGP with the special combination of crossover operators is the best here. The second best approach is 2SGP from [11]. SelfCGP with only uniform crossover operators takes third place and SelfCGP with all crossover operators is fourth.
| Classifier | Austr | Germ |
|------------|-------|------|
| SelfCGP+S | 0.904 | 0.800 |
| C          | 0.902 | 0.801 |
| 2SGP       | 0.902 | 0.797 |
| SelfCGP+U | 0.902 | 0.795 |
| C          | 1.000 |      |
| SelfCGP+U | 0.898 | 0.793 |
| onv.       | 0.898 | 0.777 |
| C4.5       | 0.891 | 0.794 |
| Fuzzy      | 0.888 | 0.783 |
| GP         | 0.874 | 0.756 |
| CART       | 0.869 | 0.783 |
| LR         | 0.866 | 0.746 |
| CCEL       | 0.852 | 0.677 |
| RSM        | 0.885 | 0.677 |

Our intention was to clarify whether the suggested approach could give results competitive with those from alternative techniques without attempting to implement the best tool for banking credit scoring. This is why we did not adapt our algorithms for these problems, making no modifications for category data and giving computational resources to 500 generations (rather than 1000 as in [11]), etc.

We can remark here that it was not so important to generate a symbolically correct formulation for solving the problems above. That is why we suggest using SelfCGP+SC in such situations.

**Conclusion**

In this paper, we study the behaviour of the self-configuring genetic programming algorithm based on crossover operators with the selective pressure and operator’s probabilistic rates self-adaptation. Unlike our previous papers, where we investigated SelfCGP as a “black box” by means of just statistics, this study gives us an opportunity to glance inside the self-configuration process and better understand how operators interplay influences the algorithm’s behaviour.

Our investigations demonstrate that both the self-configuration and new uniform crossover operators bring positive changes in GP algorithm performance. It is demonstrated over test problems of the symbolic regression. However, the main advantage of the self-configuration is not a performance improvement but the possibility to avoid the time and resource consuming stage of rational algorithm settings determination. The essential improvement of the GP algorithm performance can be achieved using both modifications together.
After that we studied the probabilistic operator’s rates interplay. Based on observations we concluded that it is not necessary to include all available variants of crossover operator in our algorithm and found the effective combination of these operators. As far as mutation and selection operators are concerned, all their variants are useful and necessary and not one of them can be rejected from the algorithm. All conclusions were derived from the statistically proved results of numerical experiments with benchmark problems from the area of symbolic regression.

Finally, we checked our modifications on two hard classification problems that demonstrated the usefulness of the proposed modifications.

We developed the self-configuring evolutionary algorithm (SelfCEA) that hybridizes SelfCGP and SelfCGA and can be used for solving different problems by means of different tools. It would be an interesting task to adopt our SelfCEA for automated design and tuning of artificial neural networks, fuzzy logic based systems, decision trees and other computational intelligence tools including problem-specific ones.

In the future we intend to check whether it is possible to find the same features of SelfCEA in solving the mentioned problems as well as to use some other large-scale data sets to test our approach and enhance its performance.

References
[1] Koza, J.R. 1992 Genetic programming: on the programming of computers by means of natural selection. MIT Press, MA, Cambridge
[2] Semenkin, E., Semenkina, M. 2012 Self-configuring genetic programming algorithm with modified uniform crossover operator Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC), pp. 1918-1923.
[3] Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (Eds.) 2010 Parallel problem solving from nature PPSN XI 11th international conference Kraków, Poland.
[4] Semenkin, E., Semenkina, M. 2012 Self-configuring genetic algorithm with modified uniform crossover operator. In: Tan, Y., Shi, Y., and Ji, Z. (Eds.): Advances in swarm intelligence, ICSI 2012. LNCS 7331, Part I, pp. 414–421. Springer, Heidelberg
[5] Gomez, J. 2004 Self adaptation of operator rates in evolutionary algorithms. In: Deb, K. et al. (Eds.) GECCO 2004. LNCS 3102, pp. 1162–1173. Springer, Heidelberg
[6] De Jong, K.A., Spears, W. 1991 On the virtues of parameterized uniform crossover Proceedings of the 4th international conference on genetic algorithms. Morgan Kaufmann
[7] Semenkin E.S., Semenkina M.E. 2007 Application of genetic algorithm with modified uniform recombination operator for automated implementation of intelligent information technologies Vestnik SibSAU 3(16) pp.27-32
[8] O’Neill M., Vanneschi, L., Gustafson, S., Banzhaf, W. 2010 Open issues in genetic programming Genetic Programming and Evolvable Machines 11 pp. 339–363.
[9] Finck, S. et al. 2009 Real-parameter black-box optimization benchmarking 2009 Presentation of the noiseless functions Technical report research center PPE
[10] Frank A., Asuncion, A. 2010 UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. University of California, School of Information and Computer Science, Irvine, CA
[11] Huang, J.-J., Tzeng, G.-H., Ong, Ch.-Sh. 2006 Two-stage genetic programming (2SGP) for the credit scoring model Applied Mathematics and Computation 174 pp. 1039–1053.
[12] Sergienko, R., Semenkin, E., Bukhtoyarov, V. 2011 Michigan and pittsburgh methods combining for fuzzy classifier generating with coevolutionary algorithm for strategy adaptation 2011 IEEE Congress on Evolutionary Computation (CEC’2011). New Orleans, LA