Self-configuring genetic programming algorithm for solving symbolic regression problems

T S Karaseva\(^1,2\) and S A Mitrofanov\(^1\)

\(^1\)Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russia
\(^2\)Siberian Federal University, 79, Svobodny Pr., Krasnoyarsk, 660041, Russia

E-mail: tatyanakarasewa@yandex.ru

Abstract. In most cases, the only way to study and solve practical problems is to investigate them with the help of models. However, this method is also to be a complicated problem where a significant part of the effort is aimed at finding the functional dependencies between input and output variables. But a great number of methods of numerical identification solve problems by developing a model that, in fact, is a black box. The reduction different types of problems to the problem of symbolic regression allow us to overcome the lack of these methods. The algorithm of genetic programming is applied for solving the problems of symbolic regression. The given paper considers an algorithmic complex that includes a genetic algorithm and an algorithm for genetic programming for solving symbolic regression problems. The uniform crossover operator is applied for these methods; it ensures the flexibility of the algorithm due to the greater diversity of structures resulting from crossover. The opportunity for selection two or more parents for recombination is realized. To automate the selection of the algorithms parameters a self-configuring procedure at the population level is realized and the efficiency of its application is proved for test problems of symbolic regression. The practical implementation of algorithms for solving classification problems and differential equations is carried out.

1. Introduction

It is supposed to formalize any studying process by developing a mathematical model \([1\text{–}2]\). Often it is the only possible way to study and solve practical problems. But it is also a complicated problem where a significant part of the effort is aimed at finding the functional dependencies between input and output variables.

However, methods of numerical identification solve problems by developing a model that is «essentially a black box». The reduction of different types of problems to the problem of symbolic regression allows us to overcome the lack of these methods \([3]\).

The problem of symbolic regression is considered to find a mathematical expression in a symbolic form that approximates the dependency between the initial and final set of variables.

The problem can be formalized as follows: let there exists a sample \((\tilde{x}_i, y_i)\), \(i = 1, 2, \ldots, n\), where \(n\) is a sample size and vector \(\tilde{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})^T\) contains a set of values of \(m\) independent variables, and \(y_i = f(\tilde{x}_i)\) is a corresponding \(\tilde{x}_i\) dependent variable. It is necessary to find the expression \(\hat{f}(\tilde{x})\) in the symbolic form that approximates the dependency between \(\tilde{x}_i\) and \(y_i\).
The algorithm of genetic programming (GP) can be applied for solving problems of symbolic regression [4]. The GP algorithm is a method when the natural evolution processes are modeled, and its peculiar feature is the representation of the individual in the form of a tree.

The problems of symbolic regression, induction of sequences, game strategies, symbolic integration and differentiation, classification, development of decision trees, etc., are solved with the help of the algorithm of genetic programming.

The problem that restricts the spread of the genetic programming application means the necessity to have the knowledge of the evolutionary algorithms theory. It is possible to apply a self-configuring algorithm of genetic programming that selects independently effective settings for the problem being solved to eliminate this difficulty.

2. Algorithmic complex

The algorithmic complex was developed during the given investigation process; a classical algorithm of genetic programming is a basis of this complex. This implementation contains the main groups of operators: selection, crossover, mutation. It is necessary to determine the performance criteria for further consideration of the algorithmic complex. This paper examines the following criteria: reliability (a percentage of successful runs) and a generation number where the algorithm found the solution.

This paper studies the appropriateness of using operators, procedures, etc. according to the scheme presented below. To test the algorithm GP 100 individuals were allocated and a number of generations was 300. A number of runs for each function was 30. Let's note that the algorithm found a solution if an error was less than 1%. According to the testing results the settings that provide the best, average and worst fitness are determined. We will call them the best, average and worst accordingly.

2.1. Operator of the uniform crossover

It was concluded from the literature analysis that only the standard and single-point crossover operators are the mostly applied in the algorithm GP. However, for the GP algorithm it is also possible to realize the uniform crossover operator that is an analog of the same crossover in genetic algorithms (GA). Its complexity is in the tree-based representation of individuals. The uniform crossover provides the flexibility of the algorithm and a greater variety of tree types [6].

Let us describe the operation of the uniform crossover operator for the GP algorithm. As a result of the uniform crossover, only offspring is developed; the sub trees of the nodes compete among themselves in order to transfer their vertex to the offspring.

A certain number of parents (from two or more) for the subsequent crossover is selected. Then nodes are selected in the tree-parents, one of them will be transferred to the offspring.

The node is selected randomly. It is worth noting that it is necessary to check its arity for each node.

About the unary node

Both possible branches of a binary node can be considered as a possible argument of a single function.

About a binary node

If one operates with a binary node, then a left branch of one of the parents is passed as a possible variant of the left branch of the offspring.

The argument of the node with a single function is passed as a possible variant to both branches of the binary node.

To justify the necessity of including this type of crossover, testing of the GP algorithm with the uniform crossover operator was performed; a set of test functions was applied for the function for which the best, average and worst settings from a standard set were determined and then everywhere the crossover type was replaced by a uniform one [7]. For these settings the experiments were performed, and the obtained results (reliability value, average generation) were compared with those already available. Figures 1-2 show the averages for all test functions of the reliability values and the numbers of the generation with the found solution.
Figure 1. Reliability of the GP algorithm depending on the presence of the uniform crossover operator.

Figure 2. Generation number of the GP algorithm where the solution was found depending on the presence of the uniform crossover operator.

It is possible to implement the choice of more than two parents for this operator. And it was done. According to the testing results a number of parents was determined to ensure the greatest effectiveness.

Figure 3. Dependence of the reliability value with a uniform crossover taking into account a number of parents.
A crossover was replaced by a uniform one with $n$ number of parents, where $n = 2, 3, ..., 10$ in settings of the «average» group obtained at the previous stage of testing. The averaged values for the given number of parents are shown in figure 3.

It should be noted that if $n$ is more than 2 the reliability decreases, and the largest value provided by a number of parents $n$ is 7. The presented data are obtained according to the testing scheme presented above.

2.2. Setting constants

The functional dependence always contains some numerical constants. It is practically impossible to determine the required constant randomly; therefore, optimization methods should be applied. There exists a block of optimization of constants realized by means of genetic algorithm (GA) in the developed algorithmic complex [5]. As a result of the access to this block, a window for entering GA parameters will be open, so it is possible to select the algorithm type (standard, self-configuring), initial parameters (population size and a number of individuals) and algorithm configuration in the case of a standard GA, similar to the GP algorithm. Functional dependencies are obtained as a result of the genetic programming algorithm operation. These dependencies almost always contain numerical constants. The genetic algorithm optimizes constants for each solution (functional dependence) obtained by the GP algorithm.

2.3. Self-configuring of evolutionary algorithms

However, when solving problems applying the standard GP and GA algorithms, the problem of choosing parameters arises. Due to the complexity of setting parameters of the genetic programming algorithm, self-configuring is applied for solving problems. Self-configuring was carried out at the level of population [8].

The probability of choosing a particular type of operator from a group depends on the success of its application. At the first stage, the probability of choosing a type of operator will have the same values. At the following stages, the following formula will be used to select an operator:

$$p_i = p_{all} + r_i \frac{(100 - n \cdot p_{all})}{scale},$$

$$error = \sum_{i=1}^{n} \frac{(y_i - y_i^*)^2}{\max(y^*) - \min(y^*)},$$

where $p_{all} = \frac{20}{n}$, $n$ is a number of operators, $r_i = \frac{\text{success}_i}{\text{used}_i}$, $scale = \sum_{i=1}^{n} r_i$, and $\text{used}_i$ is a number indicating how many times the operator $i$ was used, $\text{success}_i$ is a number of times when the $i$ operator succeeded, that is, the offspring’s fitness turned out to be better than the average fitness of the parental individuals.

The application of this procedure will significantly reduce the time resources and will simplify the interaction with algorithms of researchers who are not specialists in the field of evolutionary algorithms. A full search of the GP parameters for this implementation supposes testing of 450 combinations. Obviously, the use of self-configuring will reduce the resources for finding optimal parameters significantly.

In this algorithmic complex a self-configuring procedure is implemented both for the GP algorithm and for the GA. A block of constant optimization assumes the possibility of selection both a standard and self-adjusting type for GA.

The expediency of using self-configuring is proved by the results of testing the GP and self-configuring genetic programming (SGP) algorithms. For this test, a representative set of symbolic regression problems was applied. It is worth noting that a choice of these problems for the GP algorithm is considered as an open question [7].
The resources were selected according to the described testing scheme to search for the solution by means of GP and SGP algorithms. The results of the experiments for GP and SGP averaged over 19 problems are presented in table 1.

| Algorithm  | Reliability | Average generation |
|------------|-------------|--------------------|
| Best settings | 0.73        | 107                |
| GP         | Average settings | 0.57  | 120                |
|            | Worst settings  | 0.34  | 138                |
| SGP        | 0.68         | 113                |

It can be seen from the data presented in the table that the reliability value of the SGP is on average better than the corresponding value of the GP with average settings.

The Student's t-test was used with a p-level 0.05 to study the hypotheses of the significance of the differences. The examination of the hypothesis on the equality of mathematical expectations by the Student's t-test proves the statistically significant superiority of the SGP over GP with average settings for both reliability and a number of the average generation. The reliability of the SGP differs from the GP algorithm with the best settings insignificantly, i.e. this difference is not statistically significant, but taking into account the data of the Student's t-test we can prove a number of the average generation received with a help of SGP is worse than with a help of GP with the best settings; this difference is statistically significant and substantial.

3. Practical implementation

3.1. Problems solution of the medical diagnostics
Further, the proposed algorithmic complex will be tested in real practical problems of classification. The classification problem will be solved by developing the separating surfaces represented by the symbolic expressions. At all 10 runs were made for each problem. And for each run data set was randomly divided into a training and testing sets in proportions of 70 and 30%, respectively. The solving results will be compared with the alternative approaches.

The initial data presented in the form of a database on the diagnosis of cardiac pathology was selected for solving the first problem; the second problem concerned the diagnosis of Parkinson's disease [8].

Let's consider the results of the alternative approaches taken from [10]. Table 2 shows the percentages of the correctly classified sample objects while solving the problems described above. Let's compare the results obtained with the help of the SGP with the results of other approaches. Table 2 shows the results of the following classification methods: the k-nearest neighbor method (k-NN); artificial neural network (ANN); random forest (RF); decision trees (DT); naive Bayes classifier (NB) [11].

| Problem  | k-NN  | ANN  | RF   | DT   | NB   | Self-configuring GP |
|----------|-------|------|------|------|------|--------------------|
| Problem 1| 58%   | 80%  | 77%  | 72%  | 84%  | 85%                |
| Problem 2| 75%   | 75%  | 85%  | 90%  | 69%  | 91%                |

Taking into account the obtained data, one can say that our own algorithmic complex coped with the task is no worse than the other considered classification methods.

3.2. Cauchy problem solving in symbolic form for ordinary differential equations
The solution of ordinary differential equations (ODE) with the help of SGP was the following stage of practical implementation [12]. Solving problems was carried out with the help of the hybrid method. At the first stage a numerical solution of the ODE is performed using the Runge-Kutta method of the fourth
order. As a result, a table of numbers representing the values of the desired function at the given points will be obtained. This table is the input data for the GP algorithm. The expression in symbolic form will be found taking into account the data.

### Table 3. Cauchy problems for ODE.

| № | Equation | Initial conditions | Interval | Exact solution |
|---|----------|--------------------|----------|----------------|
| 1 | $xy' - 2y - 2x^4 = 0$ | $y(-1) = 2$ | $[-1; 1]$ | $x^2 + x^4$ |
| 2 | $xy + (x + 1)y' = 0$ | $y(0) = 1$ | $[0; 3]$ | $(x + 1)e^{-x}$ |
| 3 | $2x(x^2 + y) - y' = 0$ | $y(-1.4) = 4.139327$ | $[-1.4; 1.4]$ | $e^{x^2} - x^2 - 1$ |
| 4 | $(x - xy') + x^2 \cos(x) = 0$ | $y(0.1) = 0.05998$ | $[0.1; 6]$ | $x(0.5 + \sin(x))$ |
| 5 | $y'' + 2\sin(x) = 0$ | $y(0) = 0$ | $[0; 6]$ | $2\sin(x)$ |

The problems solved by means of the described method are presented in table 3. For each problem 10 runs were performed; according to the results the average number of the generation was determined where the solution was found and its form was identified: symbolically accurate, symbolically conditionally accurate or approximate.

So, for the first problem the average number of the generation was 21; the algorithm found 4 symbolically accurate and symbolically conditionally accurate and 2 approximate solutions out of 10 runs. A generation number for the second problem is 30, and as a result of the experiment, 4 accurate and 6 conditionally accurate solutions were obtained. While solving the third problem, 6 accurate, 2 conditionally accurate and 2 approximate solutions were obtained from 10 received symbolic expressions, and a generation number was 37. While the algorithm’s operating, 10 conditionally accurate solutions were obtained for the fourth problem. As a result of the experiment carried out according to the data of Problem 5, 6 accurate, 4 conditionally accurate solutions were obtained, and the algorithm found a solution on the 23rd generation on the average.

While solving of the Cauchy problem for ODE, the self-configuring algorithm of GP was marked with the reliability of the obtained solutions. However, it should be noted that a hybrid method has the drawback of the error accumulating due to the application of the numerical method of solving the ODE.

### 4. Conclusion

In the course of the work, an algorithmic complex included a standard and self-configuring algorithm for genetic programming to solve symbolic regression problems was implemented. Also, the genetic algorithm was applied for configuring the constants from the dependency, for which a choice of both standard and self-configuring types is possible.

The operator of the uniform crossover for the GP algorithm is presented. The effectiveness of its application has been proved experimentally. In addition, it is established that the most effective number of parents is equal to 7. But due to the large number of combinations of settings (that is still growing when the uniform operator is included) it becomes necessary to automate the selection of operators. A possible variant of self-configuring for evolutionary algorithms is described.

SCA and CGP are preferable to a standard algorithm since it allows to avoid the majority of problems for non-specialists in the field of evolutionary algorithms. After all, the best algorithm settings for a particular task are almost never known. Within the framework of this investigation, practical implementation of algorithms was carried out. The problems of diagnostics of cardialgia and Parkinson’s disease were considered. When solving real problems of medical diagnostics, the implemented algorithm coped with the problems that are no worse than methods of data mining mentioned above.

The application of the self-configuring algorithm of genetic programming in solving the Cauchy problem for ODE is considered.

### References

[1] Chaushevski A, Bosevski T and Nikolova-Poceva S 2016 *International Journal on Information*
Technologies and Security 1(8) 41-54

[2] Nikolova-Poeva S and Iliev A 2016 International Journal on Information Technologies and Security 1(8) 55–68

[3] Bukhtoyarov V V and Semenkin E S 2010 Software products and systems 3 34-38

[4] Koza J R 1992 Genetic Programming: On the Programming of Computers by Means of Natural Selection (MIT Press, Cambridge, MA)

[5] Karaseva T S and Mitrofanov S A 2017 Investigation of genetic algorithm setting by self-configuring method Youth. Society. Modern science, technologies & innovations (Krasnoyarsk: SibGU) pp 49-52

[6] Semenkin E S and Semenkina M E 2012 Lecture Notes in Computer Science, Springer-Verlag 414-21

[7] Sergienko A B 2015 Test functions for global optimization V.1.32 (SibSAU) pp 29-35, 91-95

[8] Semenkin E S and Semenkina M E 2014 Self-configuring evolutionary algorithms for modeling and optimization (TIR, Magnitogorsk) p 310

[9] Machine Learning Repository Retrieved from: http://archive.ics.uci.edu/

[10] RapidMiner Retrieved from: https://rapidminer.com/

[11] Karaseva T S 2015 Solving problems of medical diagnostics by methods of intellectual data analysis Materials of the XIX International Scientific and Practical Conference, dedicated to the 55th anniversary of the Reshetnev Siberian State Aerospace University (Krasnoyarsk: SibGU) pp 46–47

[12] Tsoulos I G and Lagaris I E 2006 Genet Program Evolvable 33-54