УДК 519.6
Doi: 10.31772/2712-8970-2025-26-1-60-70

Для цитирования: Шерстнев П. А., Семенкин Е. С. Самоконфигурируемые алгоритмы генетического программирования с адаптацией на основе истории успеха // Сибирский аэрокосмический журнал. 2025. Т. 26, № 1. С. 60–70. Doi: 10.31772/2712-8970-2025-26-1-60-70.

For citation: Sherstnev P. A., Semenkin E. S. [Self-Configuring Genetic Programming Algorithms with Success History-Based Adaptation]. *Siberian Aerospace Journal.* 2025, Vol. 26, No. 1, P. 60–70. Doi: 10.31772/2712-8970-2025-26-1-60-70.

Самоконфигурируемые алгоритмы генетического программирования с адаптацией на основе истории успеха

П. А. Шерстнев[1*], Е. С. Семенкин[2]

[1]Сибирский федеральный университет
Российская Федерация, 660041, г. Красноярск, просп. Свободный, 79
[2]Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева
Российская Федерация, 660037, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
[*]E-mail: sherstpasha99@gmail.com

Аннотация. В данной работе представлен новый метод самонастройки алгоритмов генетического программирования (ГП), который базируется на идеях метода Success History based Parameter Adaptation (SHA), изначально разработанного для алгоритма дифференциальной эволюции (ДЭ). Основная идея метода заключается в динамическом анализе истории успешных решений для адаптации параметров алгоритма в процессе поиска решения. Для реализации этой концепции схема работы классического ГП была модифицирована таким образом, чтобы имитировать схему ДЭ, что позволило интегрировать механизм SHA в ГП. Полученный алгоритм, обозначенный как SHAGP (Success-History based Adaptive Genetic Programming), демонстрирует новые возможности для адаптации параметров, таких как вероятность скрещивания и мутации. В работе также проведён обзор существующих методов самонастройки алгоритмов ГП, что позволило выявить их ключевые преимущества и ограничения и использовать эти знания при разработке SHAGP. Дополнительно предложены новые операторы скрещивания, позволяющие динамически настраивать вероятность скрещивания, учитывать селективное давление на данном этапе, а также реализующие многородительское скрещивание. Такая модификация позволяет более гибко управлять процессом рекомбинации генотипов, улучшая адаптивность алгоритма к решаемой задаче. Для настройки вероятностей применения различных операторов (селекции, скрещивания, мутации) используются методы самоконфигурирования эволюционных алгоритмов, в частности, Self-Configuring Evolutionary Algorithm и Population-Level Dynamic Probabilities Evolutionary Algorithm. В рамках работы было реализовано два варианта алгоритма – SelfCSHAGP и PDPSHAGP. Эффективность предложенных алгоритмов была проверена на наборах задач из Feynman Symbolic Regression Database. Каждый алгоритм запускался многократно на каждой задаче для получения достоверной статистической выборки, а результаты сравнивались с использованием статистического критерия Манна – Уитни. Экспериментальные данные показали, что предложенные алгоритмы достигают более высокого показателя надёжности по сравнению с существующими методами самонастройки ГП, причём метод PDPSHAGP демонстрирует наилучшую эффективность более чем в 90 % случаев. Такой универсальный механизм самонастройки может найти применение в широком наборе областей, таких как автоматизация машинного обучения, обработка больших данных, инженерный дизайн, медицина, а также в космических приложениях, например, при проектировании навигационных систем для космических аппаратов и разработке систем управления летательными аппаратами. В этих сферах критически важны высокая надёжность алгоритмов и их способность находить оптимальные решения в сложных многомерных пространствах.

Ключевые слова: самонастройка, генетическое программирование, адаптация, самоконфигурирование, скрещивание, регрессия.

Self-Configuring Genetic Programming Algorithms with Success History-Based Adaptation

P. A. Sherstnev[1*], E. S. Semenkin[2]

[1]Siberian Federal University
79, Svobodny Av., Krasnoyarsk, 660041, Russian Federation
[2]Reshetnev Siberian State University of Science and Technology
31, Krasnoyarskii rabochii prospekt, Krasnoyarsk, 660037, Russian Federation
[*]E-mail: sherstpasha99@gmail.com

Abstract. In this work, a novel method for self-tuning genetic programming (GP) algorithms is presented, based on the ideas of the Success History based Parameter Adaptation (SHA) method, originally developed for the Differential Evolution (DE) algorithm. The main idea of the method is to perform a dynamic analysis of the history of successful solutions to adapt the algorithm's parameters during the search process. To implement this concept, the operation scheme of classical GP was modified to mimic the DE scheme, allowing the integration of the success history mechanism into GP. The resulting algorithm, denoted as SHAGP (Success-History based Adaptive Genetic Programming), demonstrates new capabilities for parameter adaptation, such as the adjustment of crossover and mutation probabilities. The work also includes a detailed review of existing self-tuning methods for GP algorithms, which allowed for the identification of their key advantages and limitations and the application of this knowledge in the development of SHAGP. Additionally, new crossover operators are proposed that enable dynamic adjustment of the crossover probability, account for the selective pressure at the current stage, and implement a multi-parent approach. This modification allows for more flexible control over the process of genotype recombination, thereby enhancing the algorithm's adaptability to the problem at hand. To adjust the probabilities of applying various operators (selection, crossover, mutation), self-configuring evolutionary algorithm methods are employed, in particular, the Self-Configuring Evolutionary Algorithm and the Population-Level Dynamic Probabilities Evolutionary Algorithm. Within the framework of this work, two variants of the algorithm were implemented – SelfCSHAGP and PDPSHAGP. The efficiency of the proposed algorithms was tested on problem sets from the Feynman Symbolic Regression Database. Each algorithm was run multiple times on each problem to obtain a reliable statistical sample, and the results were compared using the Mann–Whitney statistical test. The experimental data showed that the proposed algorithms achieve a higher reliability metric compared to existing GP self-tuning methods, with the PDPSHAGP method demonstrating the best efficiency in more than 90 % of the cases. Such a universal self-tuning mechanism can find applications in a wide range of fields, such as automated machine learning, big data processing, engineering design, and medicine, as well as in space applications – for example, in the design of navigation systems for spacecraft and the development of control systems for aerial vehicles. In these areas, the high reliability of algorithms and their ability to find optimal solutions in complex multidimensional spaces are critically important.

Keywords: self-tuning, genetic programming, adaptation, self-configuration, crossover, regression.

Introduction

The field of research related to self-tuning is one of the most relevant areas in the development of evolutionary algorithms (EA). Self-tuning methods for EA have become an integral part of the algorithms presented at the IEEE Congress on Evolutionary Computation, one of the leading international forums on evolutionary computing and computational intelligence [1]. This is due to the fact that the efficiency of optimization using EA directly depends on the choice of configuration and numerical parameters, while it is impossible to determine their optimal values for a specific problem in advance. Self-tuning methods are usually divided into two classes: self-configuring, which adjust the configuration of the algorithm (options for selection, crossover, and mutation operators), and adaptive, regulating the numerical parameters of the algorithm (crossover and mutation probabilities, population

size). As optimization problems become more complex, the need for more flexible and adaptive EA increases. This is especially true for genetic programming (GP), which finds application in areas such as automated machine learning, big data processing, engineering design, and medicine, where high reliability of algorithms and their ability to find optimal solutions in complex multidimensional spaces are critical. Similar requirements for adaptability and control accuracy are observed in a number of technical applications, which also concerns some aspects of rocket and space research.

Genetic programming algorithm

The GP algorithm is a family of optimization algorithms, evolving programs represented as tree structures, each internal node of which is an operation, and the end node is an operand [2; 3]. Due to the flexibility of this coding method, GP can be used to solve problems where the structure of the solution is unknown in advance or is difficult to describe analytically. GP is most often used to solve problems of symbolic regression [4; 5], classification [6–8], formation of machine learning models and optimization algorithms [9–11], program synthesis and optimization of complex systems [12; 13]. The stages of GP operation are usually similar to the stages of most EA and include the following steps: the initial population initialization (full method, growing method, combined method); evaluation of individuals (calculation of the values of the fitness function (FF) of each individual in the population); selection of individuals that will form a new generation using a genetic selection operator (proportional, rank, or tournament selection); recombination of the selected individuals to create offspring by applying a genetic crossover operator (single-point, standard, or even crossover); mutation of the offspring individuals by applying a genetic mutation operator (point, growth, exchange, or compression); replacement of the previous generation with offspring. Then the transition to the stage of evaluation of individuals occurs and the cycle is repeated [3].

Overview of self-tuning methods for genetic programming algorithms

Many different self-tuning methods have been developed and studied for GP. Thus, in [14], one of the first methods of self-configuration of GP was proposed, referred to as Population-Level Dynamic Probabilities (PDP), in which, when creating an individual, genetic operators are selected randomly from a given set of options. In this case, the probability of selecting an operator is dynamically adjusted in the process of finding a solution so that successful individuals have a greater chance of being selected in the future. The success of an operator is defined as the achievement of a better FF value by the offspring it created than by the parent. One of the problems of this method is the uncertainty in the choice of a parent for comparison with an offspring. As a rule, the choice is made randomly. Despite this, PDP is successfully used for self-tuning not only GPs, but also other EAs [15; 16]. Another effective self-tuning method was proposed in [17]. The method is called SelfCEA (Self-Configuring Evolutionary Algorithm) and is similar to PDP in many ways – it also dynamically changes the probabilities of applying genetic operators, but based on the average FF value achieved by the operator. The probability of applying an operator that, on average, produces individuals with a higher FF value increases [5; 18]. In another method proposed in [19], each individual is assigned its own probabilities of applying each type of operator, and then, based on feedback from the quality of the solutions created, the probabilities are increased or decreased by a predetermined value. The method showed a significant increase in the reliability of the GP compared to fixed probabilities for geometrically semantic GP, but its efficiency for a standard tree-based GP (Tree-based GP) has not been proven. In addition to the methods of self-configuring GP, methods for adapting the numerical parameters of the EA were also proposed. Thus, in [20] the SAGP algorithm is described, which configures the probabilities of crossover and mutation based on the average values of the tree sizes in the previous and current generations. This allows avoiding tree growth and obtain interpretable dependencies, but can lead to an excessive reduction in the complexity of the functions being created. The authors of the article [21] proposed the CF-GP (Adaptive Crossover + Adaptive Function List) algorithm, which combines adaptive control of crossover probabilities and dynamic removal of ineffective functions from the functional set. How-

ever, the work lacks a detailed statistical analysis of the results, which makes it difficult to assess its effectiveness.

Proposed approach

The scheme of adaptation based on the history of successful applications (SHA) has proven itself as a highly effective method for adjusting the probabilities of crossover and mutation, which is confirmed by successful experimental results [22; 23] and its regular application in various algorithms, including genetic algorithms (GA). For example, in [24], the application of SHA to GA allowed creating SHAGA, which demonstrates higher reliability compared to SelfCGA on real and pseudo-boolean optimization problems. The achieved results give reason to believe that the application of SHA in GP will lead to similar improvements. This will require changing the GP operation scheme. At each generation, genetic operators are sequentially applied to each *i*-th individual from the population. First, parents are selected by selection – since the *i*-th individual already serves as the first parent, one less is selected than in the standard scheme. Then the *i*-th individual is crossed with the selected parents, after which the mutation operator is applied to the resulting offspring. This change in the algorithm design is introduced to integrate the SHA method, which adapts the mutation and crossover probabilities based on the criterion: if the offspring FF value is higher than that of the *i*-th solution, the current parameter setting is considered successful.

In addition, it is necessary to change the operation of the crossover operator. In the standard GP scheme, the crossover operator determines whether an offspring will be created, and if not, then no crossover occurs. In the SHA method, for each bit with probability *CR*, it is chosen whether to transmit it from the parent or the mutant, which is similar to the uniform crossover operator, but with a dynamically changing probability different from the fixed one equal to 0.5. Additionally, when modifying the crossover operator in GP, it is necessary to provide the possibility of selective pressure at this stage and the choice of more than two parents [18]. The crossover process is organized in two stages: first, for each gene with probability *CR*, it is determined whether it will be inherited from the first parent (the current solution) or other parents. If a gene is selected from the first parent or there are only two parents, the algorithm moves on to the next gene. Otherwise, at the second stage, a selection is made among the remaining parents taking into account their FF values, which corresponds to the approach described in [18].

The proposed modification of the crossover operator in SHAGP allows implementing multi-parent crossover with the ability to regulate its intensity using the *CR* parameter and taking into account the selective pressure at the crossover stage. In addition, it is possible to use classical operators (single-point and standard), where the procedure is performed without the previously described changes, but is initiated with the *CR* probability; if the crossover does not occur, the operator returns the first parent (the current solution). According to [18], when using multi-parent crossover, the optimal number of parents for most operators is 2 and 7, and for the tournament one – 3 and 7. However, in this algorithm, additional selective pressure is applied at the crossover stage using the selection operator at the second stage, so the total number of parents increases by 1 compared to the original implementation.

This modification allows using different variants of crossover operators: single-point, standard, uniform equiprobable with two parents, uniform equiprobable with three parents, uniform equiprobable with eight parents, uniform proportional with three parents, uniform proportional with eight parents, uniform rank with three parents, uniform rank with eight parents, uniform tournament with three parents, uniform tournament with eight parents. The selection operator can be any. In this study, the following are used: proportional, rank, tournament with a tournament size of 3, 5 and 7 individuals. The following mutation operators are used: point, growing, exchange, compression.

Since the algorithm has 160 possible configurations, the problem of determining the optimal setting for each problem to be solved arises. In this case, it is advisable to use self-configuring GP methods that dynamically adjust the parameters during operation, providing greater reliability than with random selection.

By combining all the described modifications (change in the GP operation scheme, modified crossover operator, adaptation based on the history of successful applications and self-configuring methods), we obtain a single algorithm – a self-configuring GP algorithm with adaptation based on the history of successful applications (Self-Configuring SHAGP). The pseudocode of the proposed algorithm is presented below:

1. Initialization.
- *1.1. Generate the initial population of binary trees randomly.*
- *1.2. Calculate the FF value for each individual.*
- *1.3. Initialize the parameter history:*
  - *1.3.1. Fill the H_MR array (for the mutation probability) with values 0.1.*
  - *1.3.2. Fill the H_CR array (for the crossover probability) with values 0.9.*
  - *1.3.3. Set the history index k = 0.*
- *1.4. Initialize the probabilities of applying the operators for each type:*
  - *1.4.1. P_sel (selection operators) – equally likely across all variants.*
  - *1.4.2. P_cross (crossover operators) – equally likely across all variants.*
  - *1.4.3. P_mut (mutation operators) – equally likely across all variants.*

2. Main loop (for each generation):
- *2.1. For each individual i:*
  - *2.1.1. Randomly choose an index r from the range [0, H_size].*
  - *2.1.2. Set MR_i using the Cauchy distribution with center H_MR[r] and scale 0.1.*
  - *2.1.3. Set CR_i using the Normal distribution with center H_CR[r] and standard deviation 0.1.*
  - *2.1.4. Choose a selection operator variant using the probability distribution P_sel.*
  - *2.1.5. Choose a crossover operator variant using the probability distribution P_cross.*
  - *2.1.6. Choose a mutation operator variant using the probability distribution P_mut.*
  - *2.1.7. Apply the chosen selection operator to select parents.*
  - *2.1.8. Apply the selected crossover operator to the ith individual (the first parent) and the other parents, producing an offspring with probability CR_i.*
  - *2.1.9. Apply the selected mutation operator to the resulting offspring with probability MR_i.*
  - *2.1.10. Calculate the FF value of the offspring.*
- *2.2. Replacement:*
  - *2.2.1. For each individual i: if the FF value of the offspring is better than that of the i-th individual, replace the i-th individual with the offspring.*
- *2.3. Updating the parameter history:*
  - *2.3.1. For all individuals for which the replacement occurred, collect the MR and CR values used, as well as the FF value improvements.*
  - *2.3.2. Update H_MR[k] and H_CR[k] using the weighted average of the successful values.*
  - *2.3.3. Set k = k+1 or 0 if k > H_size.*
- *2.4. Update operator application probabilities:*
  - *2.4.1. Update the application probabilities of the genetic operators P_sel, P_cross and P_mut, using the self-configuration method.*
- *2.5. Update the globally best individual.*
- *2.6. If the stopping criterion is not met, then go to 2.1.*

3. Termination:
- *3.1. Return the best individual found and the statistics of the algorithm's operation.*

Let us discuss the workflow of Self-Configuring SHAGP. Firstly, initialization is considered. The algorithm starts with generating a random population of binary trees and calculating their FF values. The initial parameters are fixed: the H_MR array is filled with the value 0.1, the H_CR array is filled with the value 0.9, and the probabilities of applying the operators (P_sel, P_cross, P_mut) are set equal to those in the original implementation of the SelfCGP and PDPGP methods. Then, forming a new

generation is discussed. Before creating a descendant, an index r is randomly selected from the parameter history for each individual. Based on it, the MR and CR values are generated: MR is determined using the Cauchy distribution with the center H_MR[r] (0.1) and the scale 0.1, CR is determined by the normal distribution with the center H_CR[r] (0.9) and the standard deviation 0.1. The selection, crossover, and mutation operators are selected based on the probabilities of their application, and a new descendant is formed with their help. Next, we will discuss adaptation of parameters. When replacing individuals, successful MR and CR values are saved for subsequent adaptation. Parameters are updated based on the weighted average of successful values. Finally, self-configuration of operators is considered. After the formation of a new generation, the probabilities of applying genetic operators are updated using the selected self-configuration method.

Study of the efficiency of self-configuring genetic programming algorithms

To test the proposed method, the Feynman Symbolic Regression Database [25] was used, containing 120 equations of varying complexity with the number of unknowns from 1 to 9. These equations cover a wide range of physical phenomena, including mechanical, electromagnetic, quantum and thermodynamic processes. Each of the tested self-configuring algorithms had the same set of types of genetic operators and a functional set. All algorithms were given the same number of generations (1000) during which they worked, and the population size (100).

The following algorithms participated in the study: SelfCGP is a version of GP based on SelfCEA with an extended set of operators, including selective pressure; PDPGP is an algorithm using the PDP mechanism for tuning operators with selective pressure; PDPSHAGP is a PDP modification of the SHAGP algorithm, implementing dynamic adaptation of crossover and mutation probabilities; SelfCSHAGP is a version of SHAGP based on SelfCEA.

For each of 120 equations, a sample of 1000 points randomly distributed in space was used. The details of the sample formation are described in [25]. After that, the data was divided into a training (750 points) and a test (250 points) sample. To take into account the stochastic nature of evolutionary algorithms, 100 runs were carried out for each problem, while the best value of the R^2 metric was saved at each run [26]. To confirm the statistical significance of the differences in the results of the algorithms, the Mann-Whitney statistical criterion with a significance level of 0.05 was used.

When comparing the results of solving regression problems using a large number of problems, the issue of interpreting the value of the determination coefficient R^2 arises, which can take negative values and thereby shift the average indicators and distort the evaluation of the methods. Often, a reliability indicator is used to solve this issue - the proportion of successfully found solutions, where success is determined by reaching a predetermined error threshold. However, this approach may lead to loss of information, since the result strongly depends on the chosen threshold. More informative is the calculation of reliability at different threshold values. Figure 1 shows a graph of the values of reliability averaged over 120 equations at different threshold values (from 0 to 1 with a step of 0.01) for each of the tested methods.

The graph (Fig. 1) shows how the reliability of different methods changes with increasing threshold. PDPSHAGP (orange dotted line) shows the best results, remaining higher than the others throughout the range. SelfCSHAGP (red dotted line) is also higher than the others, but the curve falls off faster. PDPGP (blue solid line) and SelfCGP (green dash-dotted line) are noticeably inferior, especially at high threshold values. The average reliability values calculated for all thresholds and problems are: SelfCGP – 0.742; PDPGP – 0.773; SelfCSHAGP – 0.797; PDPSHAGP – 0.848.

Fig. 2 shows pie charts constructed based on the results of a statistical test performed to compare the SelfCSHAGP and PDPSHAGP algorithms with other self-tuning algorithms. The diagrams are divided into three categories: “superior” (green sector) – the number of functions where the first algorithm showed the best results; “no difference” (gray sector) – statistically insignificant differences; “inferior” (red sector) – cases where the second algorithm demonstrated the best results.

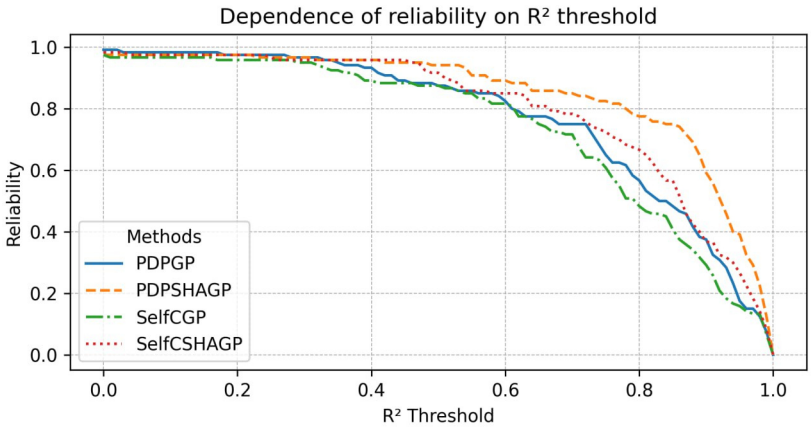

Рис. 1. Зависимость надежности от порогового значения коэффициента детерминации

Fig. 1. Dependence of reliability on the threshold value of the coefficient of determination

The diagrams show that both algorithms using SHA (SelfCSHAGP and PDPSHAGP) outperform their competitors in most test functions. SelfCSHAGP confidently outperforms SelfCGP (80 vs. 2) and significantly wins over PDPGP (41 vs. 3), although the share of problems where the differences were statistically insignificant is quite large (38 and 76, respectively). PDPSHAGP demonstrates even higher results: the algorithm outperforms SelfCGP (109 vs. 4) and PDPGP (92 vs. 2) with an insignificant number of draws, which indicates its leadership among the compared algorithms.

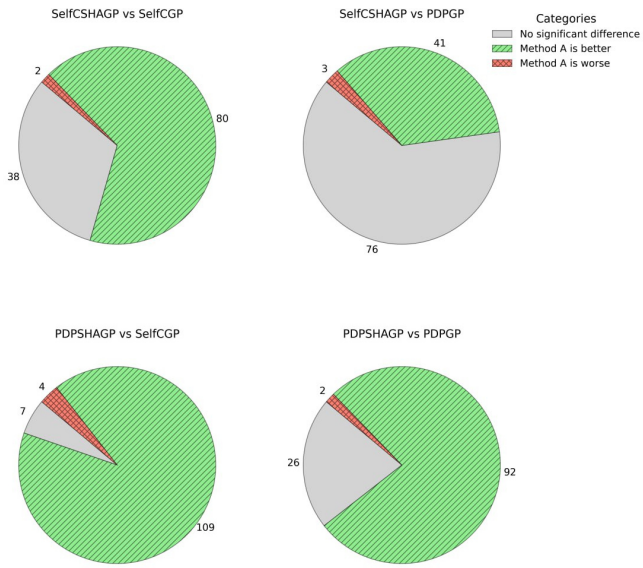

Рис. 2. Результаты сравнения методов самонастройки
с использованием статистического теста

Fig. 2. Results of comparing self-tuning methods using a statistical test

Conclusion

This paper presents and studies a self-configuring GP algorithm with parameter adaptation based on the history of successful applications. The algorithm allows configuring both the parameters of the crossover and mutation probabilities, and the variants of genetic operators. Particular attention is paid

to the modified crossover operator, which is distinguished by the ability to adapt the intensity of crossover by adjusting the probability of its application, applying selective pressure at this stage, and using multi-parent crossover. As part of the study, the algorithm is implemented in two versions that differ in the self-configuration method: SelfCSHAGP and PDPSHAGP. The results of comparative experiments on regression problems showed that the proposed algorithms outperform earlier approaches on most test problems, and in the remaining cases demonstrate comparable performance. The most effective implementation was the one using the PDPEA method for operator tuning.

The obtained results confirm the prospects of the approach and allow outlining further directions for its development: analysis of the algorithm's efficiency in solving problems of other classes (for example, in the formation of machine learning models) and the integration of additional self-tuning methods, including population size adjustment.

Благодарности. Работа выполнена при поддержке Минобрнауки России в рамках Государственного задания в сфере науки (проект № FEFE-2023-0004).

Acknowledgment. This research was funded by the State Assignment project № FEFE-2023-0004.

Библиографические ссылки

1. IEEE Congress on Evolutionary Computation [Электронный ресурс]. 2025. URL: https://www.cec2025.org/ (дата обращения: 08.01.2025).

2. Koza J. R. Genetic programming: on the programming of computers by means of natural selection. Sixth printing, 1998, Massachusetts Institute of Technology, 609 p.

3. Kuranga C., Pillay N. A Comparative Study of Genetic Programming Variants // Artificial Intelligence and Soft Computing. ICAISC 2022. Lecture Notes in Computer Science. 2023. Vol. 13588, P. 377–386. DOI: 10.1007/978-3-031-23492-7_32.

4. Genetic Programming-based Feature Selection for Symbolic Regression on Incomplete Data. Evolutionary Computation / B. Al-Helali et al. 2024. P. 1–27.

5. Karaseva T. S., Mitrofanov S. A. Self-configuring genetic programming algorithm for solving symbolic regression problems // IOP Conference Series: Materials Science and Engineering. Krasnoyarsk, 16–18 April 2020. 2020. Vol. 862. P. 52–69. DOI: 10.1088/1757-899X/862/5/052069.

6. Traffic Classification in Software-Defined Networking Using Genetic Programming Tools / S. Margariti, I. Tsoulos, E. Kiousi, E. Stergiou // Future Internet. 2024. Vol. 16. P. 338. DOI: 10.3390/fi16090338.

7. Maurya P., Kushwaha A., Prakash O. Medical Data Classification Using Genetic Programming: A Systematic Literature Review // Expert Systems. 2025. Vol. 42, No. 3. DOI: 10.1111/exsy.70007.

8. A Genetic Programming Approach to Binary Classification Problem / L. Santoso, B. Singh, S. Rajest et al. EAI Endorsed Transactions on Energy Web, 2020. DOI: 10.4108/eai.13-7-2018.165523.

9. A hyper-heuristic approach to automated generation of mutation operators for evolutionary programming / L. Hong, J. Drake, J. Woodward, E. Özcan // Applied Soft Computing. 2018. Vol. 62. DOI: 10.1016/j.asoc.2017.10.002.

10. Scott E. 'Siggy, Bassett J. Learning Genetic Representations for Classes of Real-Valued Optimization Problems. 2015. DOI: 10.1145/2739482.2768460.

11. Hyper-heuristic approach: automatically designing adaptive mutation operators for evolutionary programming / L. Hong, J. R. Woodward, E. Özcan et al. // Complex Intell. Syst. 2021. Vol. 7. P. 3135–3163. DOI: 10.1007/s40747-021-00507-6.

12. Trajectory optimization method for spacecraft orbit transfer with finite thrust. Xinan Jiaotong Daxue Xuebao / C. Wang, Y. Qu, Z. Lu et al. // Journal of Southwest Jiaotong University. 2013. Vol. 48. P. 390–394. DOI: 10.3969/j.issn.0258-2724.2013.02.030.

13. Semenkin E., Semenkina M. Spacecrafts' control systems effective variants choice with self-configuring genetic algorithm // ICINCO 2012 – Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics. Rome, 28–31 July 2012. 2012. Vol. 1. P. 84–93.

14. Niehaus J., Banzhaf W. Adaption of Operator Probabilities in Genetic Programming // Genetic Programming. EuroGP 2001. Lecture Notes in Computer Science. 2001. Vol. 2038. P. 325–336. DOI: 10.1007/3-540-45355-5_26.

15. Липинский Л. В., Кушнарева Т. В. Исследование моделей и процедур самоконфигурации генетического программирования для формирования деревьев принятия решений в задачах интеллектуального анализа данных // Вестник СибГАУ. 2016. Т. 17, № 3. С. 579–586.

16. Митрофанов С. А., Семенкин Е. С. Дифференциальная эволюция в алгоритме обучения деревьев принятия решений // Сибирский журнал науки и технологий. 2019. Т. 20, № 3. С. 312–319. DOI: 10.31772/2587-6066-2019-20-3-312-319.

17. Semenkin E. S., Semenkina M. E. Self-configuring Genetic Algorithm with Modified Uniform Crossover Operator // LNCS. 2012. Vol. 7331. P. 414–421.

18. Semenkin E., Semenkina M. Self-configuring genetic programming algorithm with modified uniform crossover // Evolutionary Computation (CEC), 2012 IEEE Congress on Evolutionary Computation, June 2012. P. 1–6. DOI: 10.1109/CEC.2012.6256587.

19. Self-tuning geometric semantic Genetic Programming / M. Castelli, L. Manzoni, L. Vanneschi et al. // Genetic Programming and Evolvable Machines. 2016. Vol. 17, No. 1. DOI: 10.1007/s10710-015-9251-7.

20. Oh S., Suh W.-H., Ahn C.-W. Self-Adaptive Genetic Programming for Manufacturing Big Data Analysis // Symmetry. 2021. Vol. 13, No. 4. P. 709. DOI: 10.3390/sym13040709.

21. Al-Madi N., Ludwig S. Adaptive Genetic Programming applied to Classification in Data Mining. Fourth World Congress on Nature and Biologically Inspired Computing (IEEE NaBIC'12), Mexico City, Mexico, November 2012. DOI: 10.1109/NaBIC.2012.6402243.

22. Tanabe R., Fukunaga A. Success-history based parameter adaptation for Differential Evolution // 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 2013, P. 71–78. DOI: 10.1109/CEC.2013.6557555.

23. Renkavieski C., Parpinelli R. L-SHADE with Alternative Population Size Reduction for Unconstrained Continuous Optimization. Computer on the Beach, September 2020, P. 351–358. DOI: 10.14210/cotb.v11n1.p351-358.

24. Stanovov V., Akhmedova S., Semenkin E. Genetic algorithm with success history based parameter adaptation // IJCCI 2019 – Proceedings of the 11th International Joint Conference on Computational Intelligence: 11, Vienna, 17–19 September 2019. Vienna : 2019. P. 180–187. DOI: 10.5220/0008071201800187.

25. Marian S., Tegmark M. E. AI Feynman: A physics-inspired method for symbolic regression // Science Advances. 2020, Vol. 6, No. 16. DOI: 10.1126/sciadv.aay2631.

26. Rights M. D., Sterba S. K. A framework for effect size measures in multilevel models: A review and recommendations // Psychological Methods. 2019. Vol. 24, No. 3. P. 289–315.

References

1. IEEE Congress on Evolutionary Computation (CEC) [Electronic resource]. 2025. Available at: https://www.cec2025.org/ (accessed: 08.01.2025).

2. Koza J. R. Genetic programming: on the programming of computers by means of natural selection. Sixth printing, 1998, Massachusetts Institute of Technology, 609 p.

3. Kuranga C., Pillay N. A Comparative Study of Genetic Programming Variants. *Artificial Intelligence and Soft Computing*. ICAISC 2022. Lecture Notes in Computer Science. 2023. Vol. 13588, P. 377–386. DOI: 10.1007/978-3-031-23492-7_32.

4. Al-Helali B. et al. Genetic Programming-based Feature Selection for Symbolic Regression on Incomplete Data. Evolutionary Computation, 2024, P. 1–27.

5. Karaseva T. S., Mitrofanov S. A. Self-configuring genetic programming algorithm for solving symbolic regression problems. I*OP Conference Series: Materials Science and Engineering*. Krasnoyarsk, 16–18 April 2020. 2020, Vol. 862, P. 52069. DOI: 10.1088/1757-899X/862/5/052069.

6. Margariti S., Tsoulos I., Kiousi E., Stergiou E. Traffic Classification in Software-Defined Networking Using Genetic Programming Tools. *Future Internet.* 2024, Vol. 16, P. 338. DOI: 10.3390/fi16090338.

7. Maurya P., Kushwaha A., Prakash O. Medical Data Classification Using Genetic Programming: A Systematic Literature Review. *Expert Systems*. 2025, Vol. 42, No. 3. DOI: 10.1111/exsy.70007.

8. Santoso L., Singh B., Rajest S., Rajan R., Kadhim K. A Genetic Programming Approach to Binary Classification Problem. EAI Endorsed Transactions on Energy Web, 2020. DOI: 10.4108/eai.13-7-2018.165523.

9. Hong L., Drake J., Woodward J., Özcan E. A hyper-heuristic approach to automated generation of mutation operators for evolutionary programming. Applied Soft Computing. 2018, Vol. 62. DOI: 10.1016/j.asoc.2017.10.002.

10. Scott E. 'Siggy, Bassett J. Learning Genetic Representations for Classes of Real-Valued Optimization Problems. 2015. DOI: 10.1145/2739482.2768460.

11. Hong L., Woodward J. R., Özcan E. et al. Hyper-heuristic approach: automatically designing adaptive mutation operators for evolutionary programming. *Complex Intell. Syst.* 2021, Vol. 7, P. 3135–3163. DOI: 10.1007/s40747-021-00507-6.

12. Wang C., Qu Y., Lu Z., An H., Xia H., Ma G. Trajectory optimization method for spacecraft orbit transfer with finite thrust. Xinan Jiaotong Daxue Xuebao. *Journal of Southwest Jiaotong University*. 2013, Vol. 48, P. 390–394. DOI: 10.3969/j.issn.0258-2724.2013.02.030.

13. Semenkin E., Semenkina M. Spacecrafts' control systems effective variants choice with self-configuring genetic algorithm. *ICINCO 2012 – Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics*. 2012, Vol. 1, P. 84–93.

14. Niehaus J., Banzhaf W. Adaption of Operator Probabilities in Genetic Programming. *Genetic Programming*. EuroGP 2001. Lecture Notes in Computer Science. 2001, Vol. 2038, P. 325–336. DOI: 10.1007/3-540-45355-5_26.

15. Lipinskiy L. V., Kushnareva T. V. [A study of models and procedures of self-configuring genetic programming for forming decision trees in data mining tasks]. *Vestnik SibSAU.* 2016, Vol. 17, No. 3, P. 579–586 (In Russ.).

16. Mitrofanov S. A., Semenkin E. S. [Differential evolution in the decision tree learning algorithm]. *Siberian Journal of Science and Technology*. 2019, Vol. 20, No. 3, P. 312–319. DOI: 10.31772/2587-6066-2019-20-3-312-319 (In Russ.).

17. Semenkin E. S., Semenkina M. E. Self-configuring Genetic Algorithm with Modified Uniform Crossover Operator. *LNCS*. 2012, Vol. 7331, P. 414–421.

18. Semenkin E., Semenkina M. Self-configuring genetic programming algorithm with modified uniform crossover. *Evolutionary Computation (CEC), 2012 IEEE Congress on Evolutionary Computation*, June 2012, P. 1–6. DOI: 10.1109/CEC.2012.6256587.

19. Castelli M., Manzoni L., Vanneschi L., Popovič A. et al. Self-tuning geometric semantic Genetic Programming. *Genetic Programming and Evolvable Machines*. 2016, Vol. 17, No. 1. DOI: 10.1007/s10710-015-9251-7.

20. Oh S., Suh W.-H., Ahn C.-W. Self-Adaptive Genetic Programming for Manufacturing Big Data Analysis. *Symmetry*. 2021, Vol. 13, No. 4, P. 709. Available at: DOI: 10.3390/sym13040709.

21. Al-Madi N., Ludwig S. Adaptive Genetic Programming applied to Classification in Data Mining. *Fourth World Congress on Nature and Biologically Inspired Computing (IEEE NaBIC'12)*. Mexico City, Mexico, November 2012. Available at: https://doi.org/10.1109/NaBIC.2012.6402243.

22. Tanabe R., Fukunaga A. Success-history based parameter adaptation for Differential Evolution. *2013 IEEE Congress on Evolutionary Computation*, Cancun, Mexico, 2013, P. 71–78. DOI: 10.1109/CEC.2013.6557555.

23. Renkavieski C., Parpinelli R. L-SHADE with Alternative Population Size Reduction for Unconstrained Continuous Optimization. *Computer on the Beach*, September 2020, P. 351–358. DOI: 10.14210/cotb.v11n1.p351-358.

24. Stanovov V., Akhmedova S., Semenkin E. Genetic algorithm with success history based parameter adaptation. *IJCCI 2019 – Proceedings of the 11th International Joint Conference on Computational Intelligence*: 11, Vienna, 17–19 September 2019. Vienna, 2019, P. 180–187. DOI: 10.5220/0008071201800187.

25. Marian S., Tegmark M. E. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*. 2020, Vol. 6, No. 16, P. 2631. DOI: 10.1126/sciadv.aay2631.

26. Rights M. D., Sterba S. K. A framework for effect size measures in multilevel models: A review and recommendations. *Psychological Methods*. 2019, Vol. 24, No. 3, P. 289–315.

Шерстнев Павел Александрович – аспирант кафедры программной инженерии, инженер-исследователь Центра искусственного интеллекта; Сибирский федеральный университет. E-mail: sherstpasha99@gmail.com.

Семенкин Евгений Станиславович – доктор технических наук, профессор; кафедра системного анализа и исследования операций, Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. E-mail: eugenesemenkin@yandex.ru. https://orcid.org/0000-0002-3776-5707

Sherstnev Pavel Aleksandrovich – graduate student, Research Engineer; Artificial Intelligence Center, Siberian Federal University. E-mail: sherstpasha99@gmail.com.

Semenkin Evgeniy Stanislavovich – Dr. Sc., Professor, Department of Systems Analysis and Operations Research; Siberian State University of Science and Technology. E-mail: eugenesemenkin@yandex.ru. https://orcid.org/0000-0002-3776-5707

Статья поступила в редакцию 24.02.2025; принята к публикации 04.03.2025; опубликована 11.04.2025
The article was submitted 24.02.2025; accepted for publication 04.03.2025; published 11.04.2025