Вестник РУДН. Серия: Инженерные исследования
RUDN Journal of Engineering Research

2026;27(1):25–36

ISSN 2312-8143 (Print); ISSN 2312-8151 (Online)
journals.rudn.ru/engineering-researches

DOI: 10.22363/2312-8143-2026-27-1-25-36
EDN: GZMJMP

Research article / Научная статья

Modification of the Structure and Parameters of Genetic Algorithms with Fuzzy Operators Implemented by Hybrid Controllers

Dmitry A. Rogachev[a], **Aleksey F. Rogachev**[b] ✉

[a] Federal Scientific Center for Hydraulic Engineering and Land Reclamation named after A.N. Kostyakov, *Moscow, Russian Federation*
[b] Volgograd State Agrarian University, *Volgograd, Russian Federation*
✉ rafr@mail.ru

Article history

Received: August 14, 2025.
Revised: November 1, 2025
Accepted: November 12, 2025

Conflicts of interest

The authors declare that there is no conflict of interest.

Abstract. The use of evolutionary genetic AI methods and, in particular, genetic algorithms (GA) ensures the construction of sufficiently universal optimization systems with various architectures and macroparameters. A systematic investigation of GA structures, parameters, and performance has made it possible to identify and synthesise key trends in their modification. The effects of GA structure and parameters on the timing and accuracy of fitness function optimization were performed on the OneMax binary chromosome coding test problem. Numerical studies of the solution of the problem of evolutionary genetic optimization have shown the applicability of a fuzzy controller to increase the efficiency of GA. Numerical experiments have demonstrated that the average fitness value increases rapidly during the initial stage of the optimization process, which can be attributed to the high initial crossover probability (*Pc*). After 20–50 epochs, the optimization process reaches a stable regime. The fuzzy adaptation of the parameters of the genetic operators makes the algorithm more robust compared to fixed parameters. Recommendations have been formulated for the selection of macroparameters and for substantiating the choice of algorithm modification strategies in specific application domains, including the allocation of limited water resources.

Keywords: optimization, modification of algorithms, adaptive parameters, membership function, fuzzy inference, water resources distribution

Authors' contribution

Rogachev D.A. — research concept and design, computer program development, text drafting; *Rogachev A.F.* — material collection, algorithm and program testing; results analysis, text drafting. All authors read and approved the final version of the article.

For citation

Rogachev DA, Rogachev AF. Modification of the structure and parameters of genetic algorithms with fuzzy operators implemented by hybrid controllers. *RUDN Journal of Engineering Research.* 2026;27(1):25–36. http://doi.org/10.22363/2312-8143-2026-27-1-25-36 EDN: GZMJMP

Модификация структуры и параметров генетических алгоритмов с нечеткими операторами, реализуемых гибридными контроллерами

Д.А. Рогачев[a], **А.Ф. Рогачев**[b]✉

[a] Федеральный научный центр гидротехники и мелиорации им. А.Н. Костякова, *Москва, Российская Федерация*
[b] Волгоградский государственный аграрный университет, *Волгоград, Российская Федерация*
✉ rafr@mail.ru

История статьи

Поступила в редакцию: 14 августа 2025 г.
Доработана: 1 ноября 2025 г.
Принята к публикации: 12 ноября 2025 г.

Заявление о конфликте интересов

Авторы заявляют об отсутствии конфликта интересов.

Аннотация. Применение эволюционно-генетических методов AI и, в частности генетических алгоритмов — genetic algorithms (GA), обеспечивает построение достаточно универсальных систем оптимизации с различными архитектурами и макропараметрами. Исследования структур, параметров и результатов функционирования GA, проведенные на основе системного подхода, позволили обобщить тенденции модификации GA, влияния структуры и параметров GA на время и точность оптимизации функции приспособленности проведены на тестовой задаче OneMax с бинарным кодированием хромосомы. Проведенные численные исследования решения задачи эволюционно-генетической оптимизация показали применимость нечеткого контроллера для повышения эффективности GA. Численными экспериментами показано, что среднее значение фитнес-функции быстро растёт в начале процесса оптимизации благодаря высокому значению начальной вероятности *Pc*, принимаемой для генетического оператора скрещивания. Через 20…50 эпох процесс стабилизируется. Нечеткая адаптация параметров генетических операторов делает алгоритм более робастным по сравнению с фиксированными параметрами. Сформулированы рекомендации по выбору макропараметров и обоснованию выбора варианты модификации алгоритмов для конкретных предметных областей, включая распределение ограниченных водных ресурсов.

Ключевые слова: оптимизация, модификация алгоритмов, адаптивные параметры, функция принадлежности, нечеткий вывод, нечеткий контроллер, распределение водных ресурсов

Вклад авторов

Рогачев Д.А. — концепция и дизайн исследования, разработка программы для ЭВМ, написание текста; *Рогачев А.Ф.* — сбор материалов, тестирование алгоритма и программы; анализ результатов, написание текста. Все авторы ознакомлены с окончательной версией статьи и одобрили ее.

Для цитирования

Рогачев Д.А., Рогачев А.Ф. Модификация структуры и параметров генетических алгоритмов с нечеткими операторами, реализуемых гибридными контроллерами // Вестник Российского университета дружбы народов. Серия: Инженерные исследования. 2026. Т. 27. № 1. С. 25–36. http://doi.org/10.22363/2312-8143-2026-27-1-25-36 EDN: GZMJMP

Introduction

The application of evolutionary and genetic artificial intelligence methods, particularly genetic algorithms (GAs), makes it possible to develop versatile optimization systems with diverse architectures.

One example is hydromelioration under conditions of limited irrigation water supply. Classical analytical methods of mathematical optimization, when applied to water resource allocation problems with a nonlinear objective function [1], do not always yield optimal solutions [2; 3]. Owing to numerous technological and environmental constraints, the optimization of water distribution requires nonlinear approaches, including artificial intelligence techniques and evolutionary genetic programming.

Genetic algorithms (GAs) are stochastic methods of global heuristic optimization inspired by the principles of natural evolution [4–7]. In conventional GAs, key parameters such as crossover probability (Pc) and mutation probability (Pm) are typically fixed throughout the search process. In contrast, adaptive GAs adjust these parameters dynamically in response to the state of the population (for example, its diversity or the fitness of individuals) in order to prevent premature convergence or stagnation. In this context, adaptive strategies that enable both structural and parametric modification of the GA during its operation have proven effective in solving optimization problems [9–13].

Numerous studies have examined methods for modifying genetic algorithms, focusing on several principal approaches: problem-oriented adaptation, procedure-oriented adaptation [14; 15], and the use of fuzzy controllers, including hardware implementations.

The problem-oriented approach involves adapting the fundamental components of the GA, such as designing the chromosome representation and selecting appropriate selection mechanisms, while accounting for the specific characteristics of the optimization problem.

Procedure-oriented adaptation involves modifying the GA architecture and its parameters during the optimization process, including their initial settings. This approach encompasses the use of adaptive fitness functions [18], dynamic adjustment of crossover probability (Pc) and mutation probability (Pm) [19; 20], as well as expansion of the solution search space [7]. In particular, [7] proposes an approach in which the values of Pc and Pm are varied according to the fitness of the current individuals. Noteworthy results have also been reported for hybrid parallel GAs [22] and AI-based GAs applied to time series forecasting [23].

Adaptive strategies employ feedback during the evolutionary modeling process to regulate parameter adjustments. An example from evolutionary strategies is the well-known Rechenberg rule, which is used to control the mutation rate. According to this rule, if the proportion of successful mutations exceeds a specified threshold (e.g., 0.2), the mutation rate should be increased; otherwise, it should be decreased.

One of the classical approaches to parameter adaptation, proposed in [6], is based on analyzing the dynamics of the fitness function. Other methods rely on measures such as population entropy or fuzzy logic; however, these approaches are generally more complex to implement.

Several algorithms for the adaptive modification of crossover and mutation parameters in genetic algorithms have been implemented in hardware, as reflected in, for example, US Patent US-4593367-A[1] and US Patent 6553357-B2[2].

A key challenge lies in selecting parameter values for problem-oriented GAs that ensure effective optimization with respect to both accuracy and computational efficiency. Because GAs are inherently dynamic and adaptive, the use of fixed parameter values contradicts the principle of evolutionary variability. This consideration motivates a natural evolutionary approach based on the adaptation of GA components during execution. Such an approach may incorporate feedback on the current state of the search process, predefined rules for dynamic parameter adjustment, and mechanisms for algorithm self-adaptation.

References [14; 16] outline the principal approaches to adaptation aimed at systematically improving GA performance, including deterministic adaptation, adaptive control, self-adaptation, and the use of fuzzy controllers.

Nevertheless, the selection of appropriate methods for modifying GA architecture and adjusting its parameters remains an open issue and requires further investigation, which constitutes the focus of this study.

[1] Guha A., Harp S.A., Samad T. Genetic algorithm synthesis of neural networks. Patent US-4593367-A, United States, 1989. URL: https://patents.google.com/patent/US5140530A/en (дата обращения: 20.11.2025).

[2] *Mathias K.E., Eshelman L.J., David J.D.* Method for improving neural network architectures using evolutionary algorithms. Patent US-6553357-B2, United States, 2003. Available from: https://patents.google.com/patent/US5140530A/en (accessed: 20.11.2025).

1. Methods and Materials

The *objective of this study* is to substantiate a fuzzy-based approach to modifying GA parameters as a direction of combinatorial artificial intelligence, to implement it in software, and to evaluate its effectiveness using a test function.

The influence of the main GA parameters on accuracy and computational performance was analyzed using the well-known OneMax test problem, for which the fitness function is defined as [5], and whose global maximum is known a priori:

$$f = \sum_{i=0}^{N-1} \text{indiv}[i], \quad (1)$$

where i denotes the gene index in the chromosome and N represents the total number of genes.

The genetic selection operator was implemented using the tournament selection method with a parameter of $n = 3$, within the recommended tournament size range of 2–4. To ensure comparability of results, the number of epochs was fixed at $n = 50$ across all experiments.

The crossover probability was adjusted according to individual fitness: it decreased for highly fit individuals in order to preserve high-quality solutions and increased for less fit individuals to promote greater search diversity.

The crossover probability parameter is adapted according to the fitness values of the parent individuals:

$$P_C = \begin{cases} \dfrac{k_1 (f_{\max} - f')}{(f_{\max} - \bar{f})} & \text{if } f' > \bar{f}, \\ k_2, \text{ otherwise} \end{cases} \quad (2)$$

where $\bar{f}$ denotes the average fitness value of the population.

A similar dependence was used to modify the mutation probability P_m, but for a single individual:

$$P_m = \begin{cases} k_3 \dfrac{f_{\max} - f}{f_{\max} - f_{\backslash \text{avg}}} & \text{if } f \geq f_{\backslash \text{avg}}, \\ k_4 & \text{if } f < f_{\backslash \text{avg}}. \end{cases} \quad (3)$$

Mutation intensity is increased for individuals whose fitness falls below the population average in order to introduce greater variation and reduce the risk of convergence to local optima. These update rules are applied iteratively: at each generation, $f_{\max}$ and f_{avg} are computed, after which the crossover and mutation parameters are adjusted for the corresponding operators. If the convergence condition is satisfied ($f_{\max} = f_{\backslash \text{avg}}$), both parameters P_c and P_m are set to their maximum values to restore population diversity.

In this study, the population size (N) was selected within the range of 40–100 individuals and remained fixed, although dynamic adjustment is possible. Smaller populations (20–50 individuals) are typically employed to accelerate computation, whereas larger populations (50–100 individuals) are preferable for high-dimensional problems in order to preserve diversity.

The number of epochs (generations) was selected in the range up to 500 depending on the stopping criterion, for example, no improvement in 50 generations.

A random uniform population was used for initialisation.

The study of adaptive GA modification methods was carried out in Python without using a specialised library, such as DEAP, to ensure the reproducibility of the results.

2. Results

Below are the specific features of GA that were investigated and tested on the OneMax test task, including changes in structure and parameters using fuzzy controllers.

2.1. Analysis of the Influence of Genetic Algorithm Structure

The plan for the numerical experiments based on the OneMax test problem comprised the following stages:

- evaluation of the effectiveness of the genetic mutation operator and analysis of the influence of the mutation probability (*Pm*);
- execution of a baseline genetic selection procedure without crossover or mutation;
- assessment of the impact of introducing the crossover operator without mutation;
- implementation and analysis of the classical version of the algorithm.

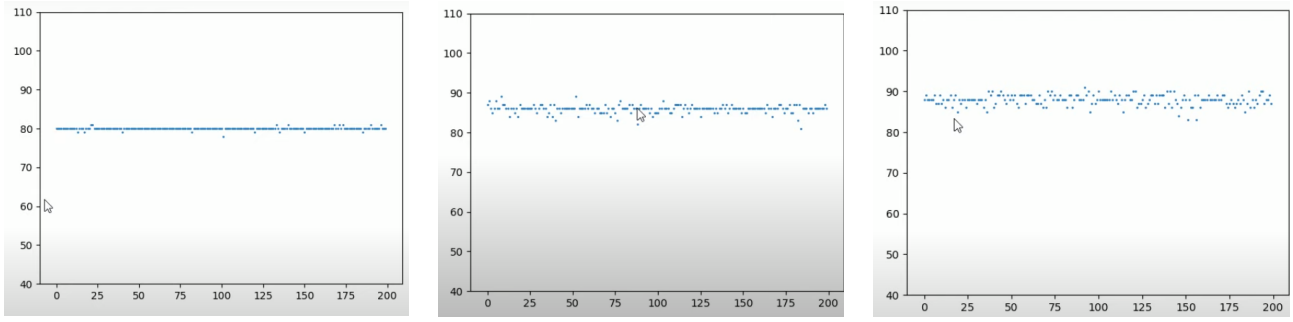
During the basic numerical experiments, the following values of the basic GA parameters were taken:

- GA constants: population.size = 200; p_crossover = 0.9; p_mutation = 0.1; max_generations = 50;
- a constant representing the length of the bit string to be optimized, one_nax_length = 70...100.

The modified program code was designed to record the set of fitness values for all individuals at each epoch, along with the corresponding average and maximum fitness values.

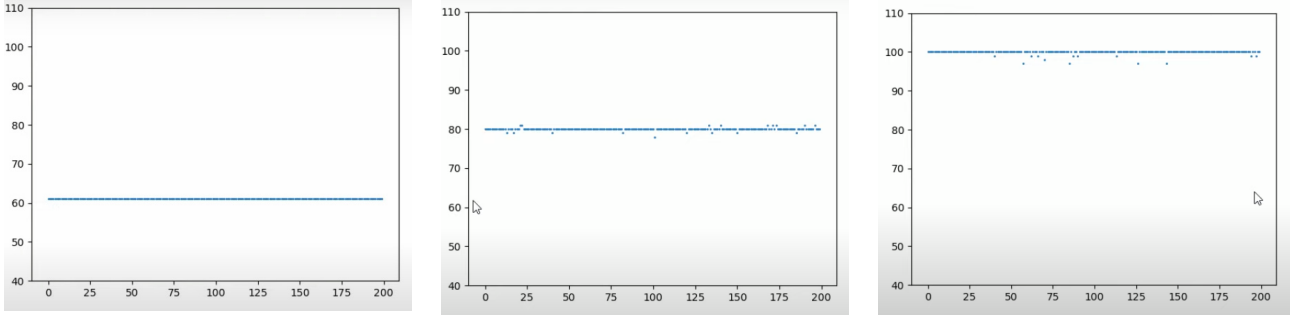
The results of the analysis of the GA configuration incorporating only the mutation operator (i.e., random search without crossover), examined as a function of the mutation probability parameter, are presented in Figure 1.

The maximum value attained was $Y_{max} = 90+$, accompanied by a substantial increase in the dispersion among individuals; however, the global optimum was not reached under this configuration. The graphical analysis of the search dynamics presented in Figure 1 further indicates that the selected number of epochs (50) was insufficient.



a *b* *c*

Figure 1. The results of the genetic search depending on the value of the probability of mutation of the genetic algorithm (random search without crossing):
a — $P_m = 0.1$; *b* — $P_m = 0.5$; *c* — $P_m = 0.9$
S o u r c e: by D.A. Rogachev.



a *b* *c*

Figure 2. The results of the genetic search for the maximum at the number of epochs $n = 50$:
a — using only the operator of 'pure' selection without crossing and mutation;
b — using the operator of 'pure' mutation (random search without crossing), *c* — the combined use of 3 genetic operators
S o u r c e: by D.A. Rogachev.

The performance of the algorithm using only the 'pure' selection operator, without crossover or mutation, over 50 epochs is illustrated in Figure 2, *a*.

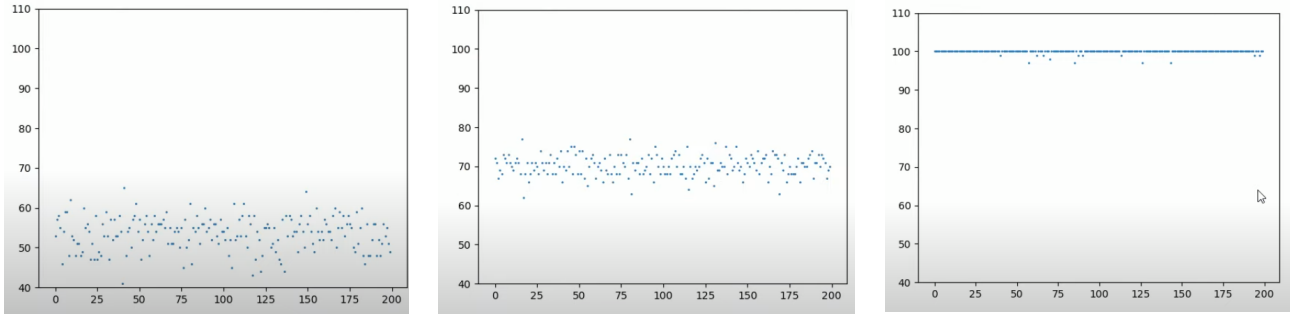
When only the genetic operator of 'pure' selection without crossover and mutation was applied, the highest value achieved was $Y_{max} = 61$, and the maximum search problem was not solved in this variant.

The effectiveness of using the genetic operator of 'pure' mutation (random search without crossover) is shown in Figure 2, *b*. The maximum value achieved was $Y_{max} = 82$, but the solution to the

maximum search problem was also not achieved in this variant.

The effectiveness of the combined application of all three genetic operators is shown in Figure 2, *c*. The maximum value achieved was $Y_{max} = 100$, which corresponds to the achievement of the solution to the test problem of finding the maximum.

A visual sequence of the phases of solving the OneMax problem using a classical GA with three genetic operators is shown in Figure 3.



a *b* *c*

Figure 3. Phases of solving the maximum search for different values of the number of epochs: *a* — $n = 5$; *b* — $n = 25$; *c* — $n = 50$
Source: by D.A. Rogachev.

It should be noted that the selected number of epochs was sufficient to obtain a satisfactory solution to the OneMax test problem.

The results of the numerical experiments indicate that the combined use of all principal GA operators (selection, crossover, and mutation) is advisable. The following subsections examine various approaches to modifying GA parameters in order to enhance algorithmic efficiency and stability.

2.2. Adaptation of Population Size

The study hypothesized, and numerical experiments were conducted to verify, that varying the population size during GA execution would influence its performance.

Reference [19] describes a "sawtooth" GA in which changes in population size are combined with periodic reinitialization, thereby enhancing algorithmic efficiency. In this approach, the population size is defined by a function, for example, relationship (4), parameterized by D and T.

$$n(t) = int\left(\frac{\overline{n} + D - 2D\left(t - T\,int\left(\frac{t-1}{T}\right) - 1\right)}{(T-1)}\right), \quad (4)$$

where D denotes the maximum population size (in individuals), and T represents the reinitialization period, expressed in epochs.

According to equation (4), the population size varies over time in a 'sawtooth' pattern, providing a compromise between expansion of the search space and computational efficiency.

A hardware implementation of adaptive GA modification is also feasible, as described, for example, in Russian Patent No. 2602973 [25]. To enhance performance, the controller incorporates a genetic algorithm parameter optimization unit with three outputs corresponding to the selection, crossover, and mutation operators, as well as an iteration counter.

2.3. Adjustment of Crossover and Mutation Parameters Using a Fuzzy Controller

Fuzzy logic controllers (FLCs) provide an effective mechanism for dynamically adjusting GA parameters, such as crossover and mutation probabilities. Based on L. Zadeh's theory of fuzzy logic, they enable the representation of uncertainty and qualitative knowledge (e.g., "population diversity is low") through linguistic variables and rule-based inference. This capability is particularly valuable in genetic algorithms, where fixed para-

meter settings may lead to premature convergence or stagnation. By adapting parameters in response to the current state of the population, such as diversity levels or changes in fitness, FLCs help balance exploration and exploitation.

In GAs, the application of an FLC typically follows a structured procedure: at each generation (or after a predefined number of generations), the controller evaluates relevant population metrics, applies a set of fuzzy rules, and determines parameter adjustments accordingly. This adaptive mechanism enhances GA performance, particularly in complex optimization problems involving multimodal objective functions. In the literature, this approach is commonly referred to as a fuzzy adaptive genetic algorithm (FAGA).

The FLC application procedure comprises six stages.

1. Identification of input variables, including metrics that characterize the state of the GA.

Typical inputs include:

- population diversity, measured either genotypically (e.g., Euclidean distance between chromosomes, ED) or phenotypically (e.g., $\mathrm{PDM1} = \frac{f\max}{f\mathrm{avg}}$, where $f$max and $f$avg denote the maximum and average values of the fitness function (FF), respectively);
- change in the fitness function (Δf), for example, defined as

$$\Delta f = (f\mathrm{avg}^{t} - f\mathrm{avg}^{t-1}) / f\mathrm{avg}^{t-1}, \quad (5)$$

where t represents the current generation.

Additional metrics may include VAC (variance of allele distribution within chromosomes) and AVA (average variance of allele distribution).

Input variables are typically normalized to the range [0, 1].

2. The fuzzification stage involves defining fuzzy sets through membership functions (MFs) for each input variable. Triangular or trapezoidal membership functions are most commonly employed. For example, population diversity may be described using the linguistic terms ‘Low’ (0–0.4), ‘Medium’ (0.3–0.7), and ‘High’ (0.6–1.0). Mathematically, the membership degree $\mu(x)$ for a triangular MF with center c and width w can be expressed as

$$\mu(x) = \max(0.1 - |x - c|/w). \quad (6)$$

3. Rule base: A set of “if — then” rules is constructed on the basis of expert knowledge, linking input variables to output parameters. For example: ‘If diversity is Low and Δf is Small, then significantly increase Pm’ (to enhance diversity). The rule set may be defined using expert judgment or optimized, for instance, by means of another GA.

4. Inference: Mamdani or Sugeno inference schemes are typically employed to aggregate the rules. For each rule, the activation degree is computed (e.g., as the minimum of the corresponding membership function values $\mu(x)$), after which the rule outputs are combined.

5. Defuzzification: The fuzzy inference results are converted into crisp parameter values. A commonly used method is the centroid (center-of-gravity) approach.

$$y = \sum(\mu_i \cdot y_i)/\sum\mu_i, \quad (7)$$

where y_i are the centres of the output MF.

New parameters:

$$P_c^{\mathrm{new}} = P_c^{\mathrm{old}} \Delta P_c \quad (8)$$

or

$$P_c^{\mathrm{new}} = P_c^{\mathrm{old}} + \Delta P_c \quad (9)$$

with constraints (e.g., $P_c \in [0.5, 0.9]$).

6. Integration into the GA: The FLC is invoked periodically, for example, every k generations. The recommended initial parameter settings are $P_c \approx 0.6 - 0{,}8$ and $P_m \approx 0.01$, depending on the type of fitness function.

2.4. Numerical Study of Crossover and Mutation Probability Adjustment Using a Fuzzy Logic Controller

The following input variables were used:

- Diversity (PDM1 = $f$max/$f$avg), normalized from the interval [1, 2] to [0, 1]);
- Δf (change in average fitness, normalized within the range [–1, 1])

Triangular membership functions were defined with the following parameters:

- for diversity: Low (center 0.2), Medium (0.5), High (0.8).
- for Δf: Negative (center –0.5), Zero (0), Positive (0.5).
- for ΔP_c, ΔP_m: Small (0.7), Medium (1.0), Big (1.3).

The fuzzy inference rule base comprised a complete set of nine if–then rules corresponding to the two inputs (3×3 combinations). The rule set, implemented as a dictionary in the Python language, is presented in Figure 4.

The expert rule base designed to maintain the exploration/exploitation balance is structured with keys defined as pairs (Diversity_Label, Delta_f_Label) and corresponding output values (Small, Medium, Large) specified in the dictionaries for 'delta_pc' и 'delta_pm'.

```
rules = {
('low', 'negative'): {'delta_pc': 'small', 'delta_pm': 'big'},
('low', 'zero'): {'delta_pc': 'medium', 'delta_pm': 'big'},
('low', 'positive'): {'delta_pc': 'big', 'delta_pm': 'medium'},
('medium', 'negative'): {'delta_pc': 'small', 'delta_pm': 'medium'},
('medium', 'zero'): {'delta_pc': 'medium', 'delta_pm': 'medium'},
('medium', 'positive'): {'delta_pc': 'big', 'delta_pm': 'small'},
('high', 'negative'): {'delta_pc': 'small', 'delta_pm': 'small'},
('high', 'zero'): {'delta_pc': 'medium', 'delta_pm': 'small'},
('high', 'positive'): {'delta_pc': 'big', 'delta_pm': 'small'},
}
```

Figure 4. A fragment of the Database of expert rules in the code of the GA program
Source: by D.A. Rogachev.

2.5. Python Code for a Fuzzy Adaptive Genetic Algorithm

As an illustration, consider the implementation of a simple genetic algorithm incorporating a fuzzy controller to adapt the parameters P_c (crossover probability) and P_m (mutation probability). The NumPy library was used for array operations, while Matplotlib was employed for visualization of the fitness function dynamics. Fuzzy logic was implemented using triangular membership functions, and inference was performed according to the Mamdani method without relying on external fuzzy logic libraries.

The OneMax problem was addressed, involving maximization of the sum of bits in a binary chromosome of length 20 or 40. The population consisted of 50 individuals and was evolved over 50 generations. Parameter adaptation was based on diversity metrics and the change in the average fitness value.

Code fragments illustrating the single-point crossover and mutation procedures are presented in Figure 5.

Simple membership functions $\mu(x)$ in [0, 1] of triangular shape with *vertices* a, b, c are defined by the following fragment (Figure 6).

The fuzzyfying function for the diversity value breaks it down into linguistic variables *low, medium,* and *high*. It takes the normalised value of diversity div in the range [0,1] as input and returns a dictionary with linguistic degrees of membership. Similarly, for delta, it takes as input a change delta normalised in the range [–1, 1] (Figure 7).

The output values are defined as follows: small = 0.7 (decrease), medium = 1.0 (no change), and big = 1.3 (increase). These coefficients serve as multipliers for P_c and P_m and are specified in the code as

```
output_values = {"small": 0.7,
"medium": 1.0, "big": 1.3}.
```

Inference and defuzzification are implemented using the Mamdani method, which approximates the centroid of the aggregated output. The rules are combined using minimum activation, and the final crisp value is obtained as a weighted average (Figure 8).

```
def crossover(parent1, parent2, pc):
    if np.random.rand() < pc:
       point = np.random.randint(1, len(parent1) - 1)
       child1 = np.concatenate((parent1[:point], parent2[point:]))
       child2 = np.concatenate((parent2[:point], parent1[point:]))
       return child1, child2
    return parent1, parent2
# Mutation function: flip bit with probability pm.
# Inverts the bit (0->1 or 1->0) for each gene independently.
def mutation(individual, pm):
    for i in range(len(individual)):
       if np.random.rand() < pm:
             individual[i] = 1 - individual[i]
return individual
```

Figure 5. Code snippets for single-point crossing and mutation functions

Source: by D.A. Rogachev.

```
def triangular_mf(x, a, b, c):
    if x <= a or x >= c:
       return 0.0
    elif a < x <= b:
       return (x - a) / (b - a)
    else:
       return (c - x) / (c - b)
```

Figure 6. Code snippets for calculating membership functions

Source: by D.A. Rogachev.

```
def fuzzify_diversity(div):
    low = triangular_mf(div, 0, 0.2, 0.4)
    medium = triangular_mf(div, 0.3, 0.5, 0.7)
    high = triangular_mf(div, 0.6, 0.8, 1.0)
    return {'low': low, 'medium': medium, 'high': high}
# Fuzzification: for delta_f (fitness function changes).
    def fuzzify_delta_f(delta):
    negative = triangular_mf(delta, -1, -0.5, 0)
    zero = triangular_mf(delta, -0.2, 0, 0.2)
    positive = triangular_mf(delta, 0, 0.5, 1)
    return {'negative': negative, 'zero': zero, 'positive': positive}
```

Figure 7. Code snippets for fuzzification

Source: by D.A. Rogachev.

```
def fuzzy_inference(div_fuzzy, delta_fuzzy):
    delta_pc_num, delta_pc_den = 0, 0
    delta_pm_num, delta_pm_den = 0, 0
for (div_l, delta_l), outputs in rules.items():
    activation = min(div_fuzzy.get(div_l, 0), delta_fuzzy.get(delta_l, 0))
       if activation > 0:
        delta_pc_num += activation * output_values[outputs['delta_pc']]
        delta_pc_den += activation
        delta_pm_num += activation * output_values[outputs['delta_pm']]
        delta_pm_den += activation
        delta_pc = delta_pc_num / delta_pc_den if delta_pc_den > 0 else 1.0
        delta_pm = delta_pm_num / delta_pm_den if delta_pm_den > 0 else 1.0
return delta_pc, delta_pm
```

Figure 8. A fragment of the defuzzification code

Source: by D.A. Rogachev.

This function takes as input the membership dictionaries of the variables div_fuzzy and delta_fuzzy and returns delta_pc and delta_pm as output multipliers.

In the program code, the linguistic output values are generated according to the rule base (see Figure 4). For example, if the first input variable is assigned the linguistic value ‘small’ and the second input has the linguistic value ‘negative’, then the parameter ‘delta_Pc’ is assigned the linguistic value ‘small’, while ‘delta_Pm’ is assigned the value “high.”

The main GA loop with fuzzy adaptation accepts the chromosome length (chrom_length = 70), population size (pop_size), and number of generations ($T = 150$) as input parameters and implements all genetic operators. The population is initialized with random binary vectors.

Typical GA performance results for different population sizes are presented in Figure 9. The output is defined as the best fitness value obtained in the final generation, and the evolution of both the best individual fitness and the population average fitness is illustrated graphically.

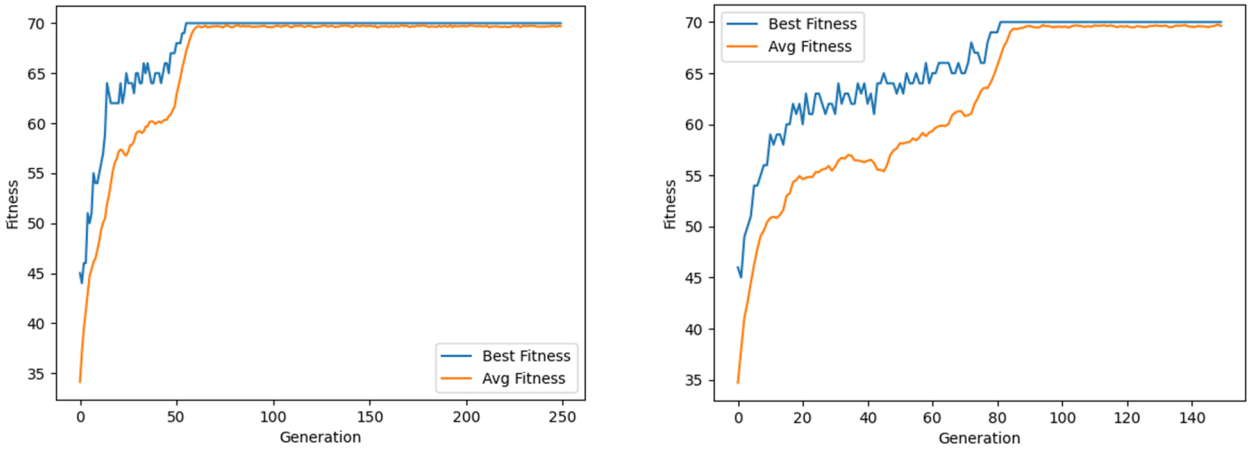


Figure 9. Results of the GA operation with a fuzzy controller:
a — chrom_length = 70, pop_size = 70; *b* — chrom_length = 70, pop_size = 90
S o u r c e: by D.A. Rogachev.

As shown in Figure 9, the algorithm successfully converges to the global optimum of the OneMax problem, where the fitness value equals 70 when all bits are set to 1. Although such convergence is expected for a relatively simple problem with a chromosome length of 70 over 150 generations, the fuzzy controller enables adaptive adjustment of parameters, particularly in situations of low diversity when the population begins to converge. In such cases, the controller increases P_m to introduce additional randomness. When the fitness function improves, it increases P_c to enhance exploitation. On average, convergence occurs within 80 to 100 generations based on typical runs, although the results may vary due to the stochastic nature of the initialization process.

When the algorithm is executed multiple times, the average fitness value increases rapidly at the initial stage, primarily due to the relatively high initial value of P_c, and subsequently stabilizes. Fuzzy adaptation improves the robustness of the GA compared to implementations with fixed parameter settings.

The adaptation mechanism can be implemented in the form of a fuzzy logic ‘controller’ that realizes a system of fuzzy production rules through software or hardware solutions.

Conclusion

A systematic analysis of GA structures, parameters, and performance outcomes has made it possible to generalize key directions for GA

modification. An investigation of the influence of GA structure and parameter settings on the time and accuracy of fitness function optimization, conducted using the OneMax problem with binary chromosome encoding, enabled the formulation of recommendations for selecting macroparameters and for substantiating algorithm modification strategies in specific application domains, including the allocation of limited water resources in arid conditions. The solution of the evolutionary genetic optimization problem confirmed the applicability of a fuzzy controller within the GA framework. Numerical experiments demonstrated that the average fitness value increases rapidly at the initial stage of the optimization process due to the relatively high initial crossover probability (P_c), after which it stabilizes. Fuzzy adaptation enhances the robustness of the GA compared with fixed parameter settings.

References

1. Rogachev D, Yurchenco I, Rogachev A. Management and optimization of sistematic water adjustment by economic-mathematic modeling methods and AI. *International Russian Automation Conference;* 2023 Sept 10–16: Sochi, Russian Federation. 2023. p. 888–893. http://doi.org/10.1109/RusAutoCon58002.2023.10272907 EDN: PWNESZ

2. Melikhova EV, Rogachev AF. Computer simulation and optimization of parameters of configuration of the contour of moistening under drip irrigation of agricultures. In: Popkova E, editor. Ubiquitous Computing and the Internet of Things: Prerequisites for the Development of ICT. Studies in Computational Intelligence. Cham: Springer; 2019. p. 1193–1201. http://doi.org/10.1007/978-3-030-13397-9_122 EDN: FJWSYW

3. Melikhova EV, Rogachev AF, Skiter NN. Information system and database for simulation of irrigated crop growing. *Studies in Computational Intelligence*. 2019; 826:1185–1191. http://doi.org/10.1007/978-3-030-13397-9_121 EDN: WGYXJK

4. Skobtsov Y, Sekirin A, Zemlyanskaya S, Chengar O, Skobtsov V, Potryasaev S. Application of object-oriented simulation in evolutionary algorithms. In: Silhavy, R., Senkerik, R., Oplatkova, Z.K., Silhavy, P., Prokopova, Z. editors. *Advances in Intelligent Systems and Computing*. Cham: Springer. 2016. p. 453–462. http://doi.org/10.1007/978-3-319-33389-2_43 EDN: WWDZIT

5. Murillo-Morera J, Quesada-López C, Castro-Herrera C, Jenkins M. A genetic algorithm based framework for software effort prediction. *Journal of Software Engineering Research and Development*. 2017;5(4):1–33. https://doi.org/10.1186/s40411-017-0037-x

6. Patnaik LM, Mandavilli S. Adaptation in genetic algorithms. In: *Genetic Algorithms for pattern Recognition*. CRC Press; 1996. 1st ed. p. 45–64. http://doi.org/10.1201/9780203713402-3

7. Mahfoud SW. A comparison of parallel and sequential niching methods. *Proceedings of VI International conference on genetic algorithms.* 1995;136–143. Available from: https://dblp.org/rec/conf/icga/Mahfoud95.html (accessed: 12.05.2025)

8. Grefenstette JJ. Lamarkian learning in multi-agent environment. *Proceedings of the 4th international conference on genetic algorithms*. 1991;303–310. Available from: https://dblp.org/rec/conf/icga/Grefenstette91 (accessed: 12.05.2025).

9. Whitley D, Gordon V, Mathias K. Lamarkian evolution, the Baldwin effect & function optimization. *Parallel Problem Solving from Nature — PPSN III. PPSN 1994. Lecture Notes in Computer Science.* Berlin, Heidelberg: Springer Publ.; 1994;866:5–15. https://doi.org/10.1007/3-540-58484-6_245

10. Davidor Y. A genetic algorithm applied to robot trajectory generation. In: Davis L, editor. *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold Publ.; 1991.

11. Moscato P, Norman MG. A memetic approach for the traveling salesman problem: implementation of computational ecology for combinatorial optimization on message-passing systems. *Parallel computing and transputer applications.* Fogarty TC, editor. Berlin, Heidelberg: Springer Publ.; 1992;1:177–186.

12. Radcliffe NJ, Surry PD. Formal memetic algorithm. In: Fogarty TC, editor. *Proceeding — Selected Papers from AISB Workshop on Evolutionary Computing*. Berlin, Heidelberg: Springer Publ.; 1994. p. 1–16. https://doi.org/10.1007/3-540-58483-8_1

13. Singh BK, Misra AK. Software effort estimation by genetic algorithm tuned parameters of modified constructive cost model for NASA software projects. *International Journal of Computer Applications*. 2012;59(9):22–26. https://doi.org/10.5120/9577-4053

14. Herera F, Lozano M. Adaptation of genetic algorithm parametres based on fuzzy logic controllers. In: Herera F, Verdegay J, editors. *Genetic algorithms and soft computing*. 1996;95–125.

15. Mitsuo G, Runwei C, Lin L. *Network models and optimization*. London: SpringerVerlag London Limited; 2008. https://doi.org/10.1007/978-1-84800-181-7

16. Hinterding R, Michalevicz Z, Eiben A. Adaptation in evolutionary computation: a survey. *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97);* 1997 Apr 3–16; Indianapolis, IN, USA; IEEE; 2002 Aug 06 p. 65–69. http://doi.org/10.1109/ICEC.1997.592270

17. Davis L. *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold; 1991. ISBN 10 0442001738

18. Juldstrom B. What have you done for me lately adapting operator probabilities in a steady-state genetic algorithm. *Proceedings 6th international conference on Gas*. San Francisco: Morgan Kaufmann Publ.; 1995. p. 81–87. ISBN 1-55860-370-0

19. Koumousis VK, Katsaras CP. Asow-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Transactions on Evolutionary Computation*. 2006;10(1): 19–28. http://doi.org/10.1109/TEVC.2005.860765

20. Lin L, Gen M. Auto-tuning strategy for evolutionary algorithms: Balancing between exploration and exploitation. *Soft Computing*. 2009;13:157–168. http://doi.org/10.1007/s00500-008-0303-2 EDN: IYKVAZ

21. Moel MC, Stewart CV, Kelly RB. Reducing the search time of a steady state genetic algorithm using the immigration operator. *Proceedings IEEE International Conference Toolsfor AI*. 1991 Nov 10–13; Danbury, CT, USA; IEEE; 1991. p. 500–501. http://doi.org/10.1109/TAI.1991.167032

22. Gladkov LA, Gladkova NV, Semushin EY. Parallel hybrid genetic algorithm for solving design and optimization problems. *Advances in Intelligent Systems and Computing*. In: Hu Z, Petoukhov S, He M. editors. 2020; 1127:249–258. http://doi.org/10.1007/978-3-030-39216-1_23 EDN: PMLJYG

23. Kureychick VM, Kaplunov TG. Time series forecasting method based on genetic algorithm for predicting the conditions of technical systems. *Journal of Physics: Conference Series*. 2019:032046. http://doi.org/10.1088/1742-6596/1333/3/032046 EDN: OZIEUE

24. Labinskiy A. The perspective trend of use the genetic algorithm. *Natural and Man-Made Risks (Physico-Mathematical and Applied Aspects)*. 2023;2(45):81–88. (In Russ.) EDN: SXNYDF

25. Rogachev DA, Kirejcheva LV, Yurchenko IF, Timoshkin AD, Kuznetsov YuS, Frolina EA. *Neural network with genetic algorithm training control device*. Patent of the Russian Federation No. 2843987 C1, IPC G06N 3/08, No. 2024120049: application 07.17.2024: published 07.23.2025. EDN: ZSPIAC

About the authors

Dmitry A. Rogachev, PhD in Technical Sciences, Senior Researcher, Federal Scientific Center for Hydraulic Engineering and Land Reclamation named after A.N. Kostyakov, 44/2 Bolshaya Akademicheskaya St, Moscow, 127434, Russian Federation; eLIBRARY SPIN-code: 8413-5020, ORCID: 0009-0003-4014-4770; e-mail: Rogachev.soft@gmail.com

***Aleksey F. Rogachev*,** Doctor of Technical Sciences, Professor of the Department of Mathematical Modeling and Computer Science, Volgograd State Agrarian University, 28 Lenina avenue, Volgograd, 400005, Russian Federation; eLIBRARI SPIN-code: 8413-5020, ORCID: 0000-0002-3077-6622; e-mail: rafr@mail.ru