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

Научная статья / Research article

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

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

Введение

Применение эволюционно-генетических методов AI, и в частности оптимизационных генетических алгоритмов — genetic algorithms (GA), обеспечивает построение достаточно универсальных систем оптимизации с различными архитектурами. В качестве примера таких систем можно привести гидромелиорацию в условиях нехватки оросительной воды. Известные аналитические методы математической оптимизации, применительно к решению задачи распределения водных ресурсов с нелинейной целевой функцией [1], не всегда могут обеспечить оптимальные результаты [2; 3]. Вследствие многочисленных технологических и экологических ограничений для решения задачи оптимизации распределения водных ресурсов требуется применение нелинейных методов, включая искусственного интеллекта — Artificial intelligence (AI) и эволюционно-генетического программирования.

Генетические алгоритмы — это стохастические методы глобальной эвристической оптимизации, аналогичные природной эволюции [4–7].

В стандартных GA параметры, такие как вероятность кроссовера P_c и мутации P_m, фиксированы. Однако в адаптивных GA эти параметры динамически изменяются в зависимости от состояния популяции (например, разнообразия, фитнеса индивидов), чтобы избежать преждевременной сходимости или застоя. При этом эффективным является адаптационный подход, обеспечивающий структурно-параметрическую адаптацию GA в процессе его функционирования при решении оптимизационных задач [9–13].

Известны многочисленные исследования методов модификации GA, которые исследуют следующие базовые варианты: проблемно-ориентированная адаптация; процедурно-ориентированная адаптация [14; 15], нечеткие контроллеры, включая реализуемые аппаратно.

Первый подход предполагает модификацию базовых компоненты GA (построение хромосомы особи, обоснование способов реализации процедуры отбора с учетом специфики оптимизационной задачи).

Процедурно-ориентированная адаптация реализует модификацию архитектуры GA и его параметров в процессе оптимизации, включая начальные значения параметров. К ним относятся адаптивные фитнес-функции [18], а также корректировка значений вероятностей скрещивания Pc и мутации Pm [19; 20], либо увеличение самого гиперпространства возможных решений. В [7] описан подход, использующий значения вероятностей P_c скрещивания и P_m мутации в зависимости от значения фитнес-функции текущих особей. Следует также отметить интересные результаты гибридных параллельных (hybrid parallel) GA [22], а также time series (TS) forecasting GA, основанный на AI [23].

Адаптивная адаптация использует обратную связь в процессе моделирования эволюции для корректировки изменения параметров. Примером такого подхода в эволюционных стратегиях является известное «правило Решенберга», применяемое для корректировки величины вероятности в операторе мутации. В случае, если улучшающая результат мутация превысит порог, например, 0,2, величину мутации следует увеличить, и наоборот.

Один из классических подходов к адаптации параметров, предложенный в [6], основан на анализе динамики фитнес-функции. Существуют также подходы на основе энтропии популяции или fuzzy-логики, но они сложнее в реализации.

Ряд алгоритмов адаптивной модификации параметров скрещивания и мутация GA реализованы аппаратно, например, по патентам США US-4593367-A «Синтез архитектуры нейронных сетей с использованием генетических алгоритмов»[1]; патент US 6553357 B2 «Метод улучшения архитектуры нейронных сетей с помощью эволюционных алгоритмов».[2]

Важной проблемой является выбор таких значений параметров проблемно-ориентированных GA, которые обеспечивают эффективную оптимизацию решения по совокупности критериев точности и быстродействия. Поскольку GA является динамическим и одновременно адаптивным, то неизменные значения его параметров противоречат самой идее эволюционной изменчивости. Отсюда возникает естественный эволюционный подход, основанный на адаптации компонентов GA в процессе его функционирования. Например, можно использовать обратную связь о текущем состоянии процедуры; определенные правила динамической модификации; механизм самоадаптации алгоритма.

В [14; 16] приведены базовые подходы к адаптации, системно обеспечивающие повышение эффективности GA, включающие детерминированную адаптацию, адаптивную адаптацию, самоадаптацию и нечеткие контроллеры.

Однако проблемы выбора метода модификации архитектуры GA и адаптации его параметров требуют дальнейшего исследования, чему посвящена настоящая статья.

1. Методы и материалы

Цель исследования — обоснование метода нечеткой модификации параметров GA как направления комбинаторного искусственного интел-

[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. URL: https://patents.google.com/patent/US5140530A/en (дата обращения: 20.11.2025).

лекта (AI), программная реализация и оценка его эффективности тестовой функции.

Анализ влияния базовых параметров GA на точность и производительность проводился на известной тестовой задаче OneMax, фитнес-функция для которой имеет вид [5], а максимум известен априори:

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

где i — номер гена в хромосоме; N — количество генов в хромосоме.

Генетический оператор селекции реализовывался методом «турнирного отбора» с параметром $n = 3$ при рекомендациях размера турнира 2–4. Число эпох для сопоставимости получаемых результатов составляло $n = 50$ во всех экспериментах.

Значение вероятности скрещивания (кроссовера) уменьшается для индивидов с высоким фитнесом, чтобы сохранить хорошие решения, и увеличивается для слабых, чтобы стимулировать поисковое разнообразие.

Значение параметра вероятности кроссовера адаптируется в зависимости от величины фитнеса родителей:

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

где $\bar{f}$ — среднее значение фитнес-функции.

Для модификации вероятности мутации P_m использовалась аналогичная зависимость, но для отдельного индивида:

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

Мутация усиливается для индивидов ниже среднего, чтобы ввести больше случайности и избежать локальных оптимумов. Эти формулы применяются итеративно: на каждом поколении вычисляются $f_{\max}$ и f_{avg}, затем для каждого соответствующего оператора (кроссовер/мутация) параметры корректируются. Если выполняется условие $f_{\max} = f_{\backslash \text{avg}}$ (популяция сходится), то P_c и P_m устанавливают в максимальные значения для «разгона» разнообразия.

Размер популяции N выбирался в диапазоне 40…100 индивидов в данном исследовании не адаптировался, хотя может быть и динамическим. Малые размеры популяции (20…50) принимают для ускорения вычислений. Большие размеры (50…100) используют для задач с высокой размерностью, чтобы поддерживать разнообразие.

Число эпох (поколений) выбиралось в диапазоне до 500 в зависимости от критерия остановки, например отсутствия улучшения в 50 поколениях.

Для инициализации использовалась случайная равномерная популяция.

Исследование методов адаптивной модификации GA осуществлялось на языке Python без использования специализированной библиотеки, например DEAP, для обеспечения воспроизводимости результатов.

2. Результаты

Ниже представлены исследованные специфические особенности GA, протестированные на тестовой задаче OneMax, включая изменение структуры и параметров посредством нечетких контроллеров.

2.1. Исследование влияния структуры генетического алгоритма

План проведения численных экспериментов с использованием тестовой задачи OneMax предусматривал:

- анализ эффективности генетического оператора мутации и исследование влияния вероятности мутации P_m;
- запуск «чистого» генетического оператора отбора без процедур скрещивания и мутации);
- результативность добавления процедуры «скрещивания», но без мутации;
- запуск и анализ классического варианта алгоритма.

При проведении основных численных экспериментов принимались следующие значения базовых параметров GA:

- константы GA: population.size = 200; p_crossover = 0,9; p_mutation = 0,1; max_generations = 50;

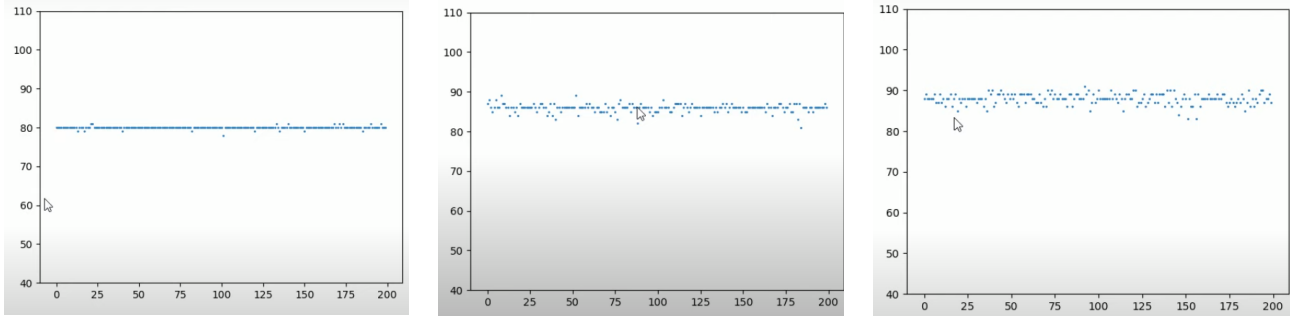
■ константа, представляющая длину подлежащей оптимизации битовой строк:

one_nax_length = 70…100.

Модифицированный код программы включал сохранение списка «values» — значений приспособленностей особей на каждой эпохе, а также их средних и максимальных значений.

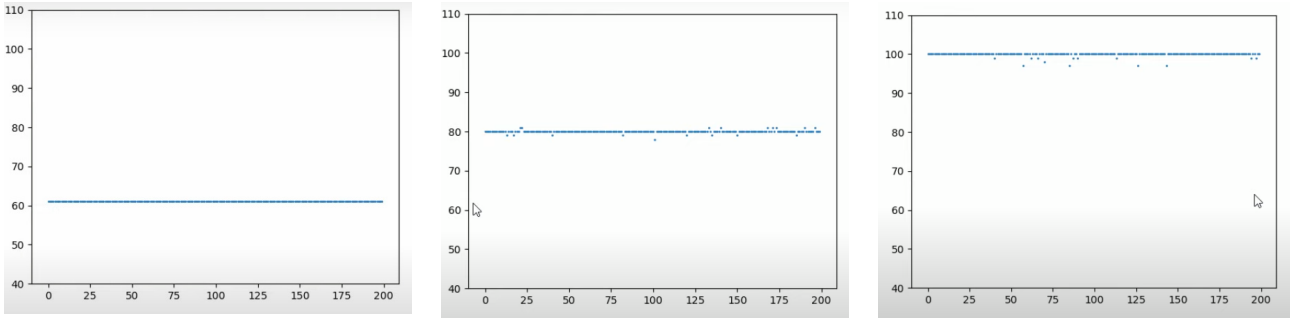
Результаты исследования конфигурации GA, включающей генетический оператор «чистой» мутации (случайный поиск без скрещивания) в зависимости от значения параметра вероятности мутации, представлены на рис. 1.

Максимальное достигнутое значение составило Y_{max} = 90+ при существенном возрастании разброса между индивидуумами, однако решение задачи поиска максимума в таком варианте не достигнуто. Графический анализ динамики поиска, представленного на рис. 1, также показывает недостаточность принятого количества эпох, равного 50.



a *b* *c*

Рис. 1. Результаты генетического поиска в зависимости от значения вероятности мутации генетического алгоритма (случайный поиск без скрещивания):
a — P_m = 0,1; *b* — P_m = 0,5; *c* — P_m = 0,9
И с т о ч н и к: выполнено Д.А. Рогачевым.



a *b* *c*

Рис. 2. Результаты генетического поиска максимума при числе эпох n = 50:
a — применение только оператора «чистого» отбора без скрещивания и мутации;
б — применение оператора «чистой» мутации (случайный поиск без скрещивания),
в — совместное применение трех генетических операторов
И с т о ч н и к: выполнено Д.А. Рогачевым.

Результативность применения только генетического оператора «чистого» отбора без скрещивания и мутации для числа эпох 50 представлена на рис. 2, *а*.

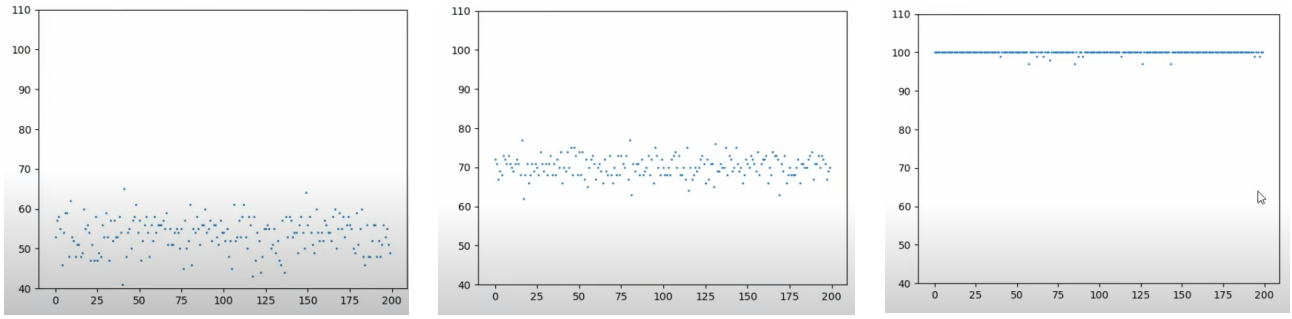
В случае применения только генетического оператора «чистого» отбора без скрещивания и мутации наибольшее достигнутое значение составило Y_{max} = 61, при этом решение задачи поиска максимума в таком варианте не достигнуто.

Результативность применения генетического оператора «чистой» мутации (случайный поиск без скрещивания) представлена на рис. 2, *б*. Максимальное достигнутое значение составило

$Y_{max} = 82$, но решение задачи поиска максимума в таком варианте также не достигнуто.

Результативность совместного применения всех трех генетических операторов пред-ставлен на рис. 2, *в*. Максимальное достигнутое значение составило $Y_{max} = 100$, что соответствует достижению решения тестовой задачи поиска максимума.

Наглядная последовательность фаз решения задачи OneMax классическим GA с тремя генетическими операторами представлена на рис. 3.



a *b* *c*

Рис. 3. Фазы решения поиска максимума при различных значениях числа эпох:
а — *n* = 5; *б* — *n* = 25; *в* — *n* = 50
И с т о ч н и к: выполнено Д.А. Рогачевым.

Отметим, что для отыскания удовлетворительного решения тестовой задачи OneMax принятого количества эпох оказалось достаточно.

По результатам проведенных численных экспериментов можно сделать вывод о целесообразности совместного использования всех основных генетических операторов GA (Отбор; Скрещивание; Мутация). В следующих подразделах будут исследованы различные варианты модификации параметров GA, позволяющие повысить его эффективность и устойчивость.

2.2. Адаптация размера популяции особей

Гипотезой исследования, проверенной численным исследованием, явилось предположение, что изменение размера численности популяции в процессе реализации GA повлияет на эффективность его применения.

В [19] описан алгоритм «пилообразного» GA, в котором объем популяции комбинируется с периодической инициализацией (изменением), что обеспечивает возрастание эффективности алгоритма. Функция изменения объема популяции задается, например, зависимостью (4) с параметрами D и T.

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

где D — наибольший объем популяции, штук; T — период новой инициализации, эпох.

Согласно (4), график изменения размера популяции во времени имеет «пилообразную» форму, что обеспечивает компромисс между расширением области поиска и производительностью алгоритма.

Возможно также аппаратное решение для адаптивной модификации GA, например, по патенту RU № 2602973 [25]. Для повышения эффективности работы, упомянутый контроллер снабжен блоком оптимизации параметров генетического алгоритма с тремя выходами, соответствующими генетическим операторам отбора, скрещивания и мутации, а также счетчиком итераций.

2.3 Корректировка параметров скрещивания и мутации методом нечеткого контроллера

Нечеткие контроллеры (FLC, fuzzy logic controllers) — это мощный инструмент для динамической модификации параметров GA, таких как вероятность скрещивания и мутации. Они основаны на нечеткой логике Л. Заде, которая позволяет обрабатывать неопределенности и качественные знания (например, «разнообразие популяции низкое») через лингвистические переменные и правила. Это особенно полезно в GA, где фиксированные параметры могут привести к преждевременной сходимости или застою, а FLC

адаптирует их на основе текущего состояния популяции (например, разнообразия или изменения фитнеса), балансируя между потенциалом поиска (exploration) и популяционным улучшением (exploitation).

Применение FLC в GA основано на следующем подходе: на каждом поколении (или через фиксированное число поколений) контроллер оценивает входные метрики популяции, применяет нечеткие правила и формирует корректировки параметров. Это улучшает производительность GA на сложных задачах, таких как оптимизация с мультимодальными функциями. Такой подход описан в литературе как fuzzy adaptive GA (FAGA).

Алгоритм применение FLC включает шесть этапов.

1. Определение входов (inputs) включая метрики, отражающие состояние GA. Типичными входами являются:

- разнообразие популяции (diversity): генотипическое (например, евклидова дистанция между хромосомами, ED) или фенотипическое (например, $\mathrm{PDM1} = \frac{f\max}{f\mathrm{avg}}$, где $f\max$ и $f\mathrm{avg}$ — максимальное и среднее значение фитнес-функции (ФФ);
- изменение ФФ (Δf), например, по зависимости

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

где t — текущее поколение.

Другие метрики: VAC (variance of alleles in chromosomes) или AVA (average variance of alleles).

Входы обычно нормализуют в диапазон [0,1].

2. Фаззификация (fuzzification) определяет нечеткие множества (membership functions, MF) для каждого входа. Обычно используют треугольные или трапецеидальные функции. Например, для diversity: «Low» (0–0,4), «Medium» (0,3–0,7), «High» (0,6–1,0). Математически степень принадлежности $\mu(x)$ для треугольной MF с центром c и шириной w можно задать зависимостью

$$\mu(x) = \max(0{,}1 - |x - c|/w). \quad (6)$$

3. База правил (rule base): создают правила вида «if-then» на основе экспертного знания. Правила связывают входы с выходами. Например: «(If diversity is Low and Δf is Small), then increase Pm strongly» (чтобы добавить разнообразия).

Правила могут быть получены экспертными методами или оптимизированы (например, другим GA).

4. Инференс (inference): используют методы Мамдани или Сугено для агрегации правил. Для каждого правила вычисляют степень активации (min входных значений функций $\mu(x)$), затем агрегируют выходы.

5. Дефаззификация (defuzzification): преобразуют нечеткий вывод в crisp-значение. Популярный метод — центр тяжести (centroid):

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

где y_i — центры выходных MF.

Новые параметры:

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

или

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

с ограничениями (например, $P_c \in [0{,}5,\ 0{,}9]$).

6. Интеграция в GA. Периодически вызывают FLC каждые k поколений. Начальные значения вероятностей рекомендуют принимать $P_c \approx 0{,}6 - 0{,}8$, $P_m \approx 0{,}01$ в зависимости от вида ФФ.

2.4. Численное исследование корректировки вероятностей скрещивания и мутации, реализуемой методом нечеткого контроллера

Входами являются следующие величины:

- Diversity (PDM1 = $f\max/f\mathrm{avg}$), нормализовано в [1, 2] → [0, 1]);
- Δf (изменение среднего фитнеса, нормализовано в [–1, 1]).

Параметры треугольных функций принадлежности (**Membership functions**):

- для diversity: Low (центр 0,2), Medium (0,5), High (0,8);
- для Δf: Negative (центр –0,5), Zero (0), Positive (0,5);
- для ΔP_c, ΔP_m: Small (0,7), Medium (1,0), Big (1,3).

База продукционных правил нечеткого вывода включала полный набор из 9 правил «if-then» для двух входов, 3×3. Набор правил, записанный в форме словаря на языке Pethon, представлен на рис. 4.

```
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'},
}
```

Рис. 4. Фрагмент Базы экспертных правил в коде программы ГА

И с т о ч н и к: выполнено Д.А. Рогачевым.

База экспертных правил для баланса exploration/exploitation включает Ключи: (Diversity_Label, Delta_f_Label) и соответствующие им Значения (small, medium, big) словаря для 'delta_pc' и 'delta_pm'.

2.5. Код на Python для нечеткого адаптивного генетического алгоритма

Рассмотрим пример реализации простого генетического алгоритма с нечетким контроллером для адаптации параметров P_c (вероятность кроссовера) и P_m (вероятность мутации). Для работы с массивами использовалась библиотека NumPy, а для визуализации Matplotlib, отображающая эволюцию фитнес-функции). Fuzzy logic реализован с треугольными функциями принадлежности. Нечеткий вывод использует метод Мамдани без внешних fuzzy-библиотек.

Решается задача OneMax: максимизация суммы битов в бинарной хромосоме длины 20/40. Популяция включает 50 индивидов, оптимизируемых в течение 50 поколений. Адаптация параметров происходит на основе метрик разнообразия (diversity) и значения изменения средней величины фитнес-функции.

Фрагменты кода для функций одноточечного скрещивания и мутации приведены на рис. 5.

Простые функции принадлежности (membership functions) $\mu(x)$ *в* [0, 1] треугольного вида *с вершинами a, b, c* задаются следующим фрагментом (рис. 6).

Функция фаззификации для величины разнообразия (diversity) разбивает ее на лингвистические переменные low, medium, high. Она принимает на вход нормализованное в [0,1] значение разнообразия div, а возвращает словарь с лингвистическими степенями принадлежности. Аналогично для delta принимает на вход нормализованное в диапазон [–1, 1] изменение delta (рис. 7).

Значения для выходов: small = 0,7 (снижение), medium = 1,0 (без изменений), big = 1,3 (увеличение) определяет множители для P_c и P_m посредством строки:

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

Инференс и дефаззификация реализуются методом Mamdani, вычисляющего приближенный центр (centroid), при этом агрегируются правила с min-активацией и вычисляет взвешенное среднее (рис. 8).

Такая функция принимает на вход словари принадлежностей переменных div_fuzzy и delta_fuzzy, а возвращает в качестве выходов delta_pc, delta_pm — множители.

В коде программы согласно правилам из базы (см. рис. 4) формируются лингвистические значения выходов. Например, если 1-й вход функции принимает лингвистическое значение 'small' и 2-й вход имеет лингвистическое значение 'negative', то параметр 'delta_Pc' принимает лингвистическое значение 'small', а параметр 'delta_Pm' — 'big'.

Основной цикл GA с нечеткой адаптацией принимает на вход значение длины хромосомы chrom_length = 70, размер популяции pop_size и число эпох (поколений) T = 150 и реализует все генетические операторы.

Инициализация популяции задает случайные бинарные векторы.

```
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
```

Рис. 5. Фрагменты кода для функций одноточечного скрещивания и мутации

И с т о ч н и к: выполнено Д.А. Рогачевым.

```
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)
```

Рис. 6. Фрагменты кода для вычисления функций принадлежности

И с т о ч н и к: выполнено Д.А. Рогачевым.

```
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}
```

Рис. 7. Фрагменты кода для фаззификации

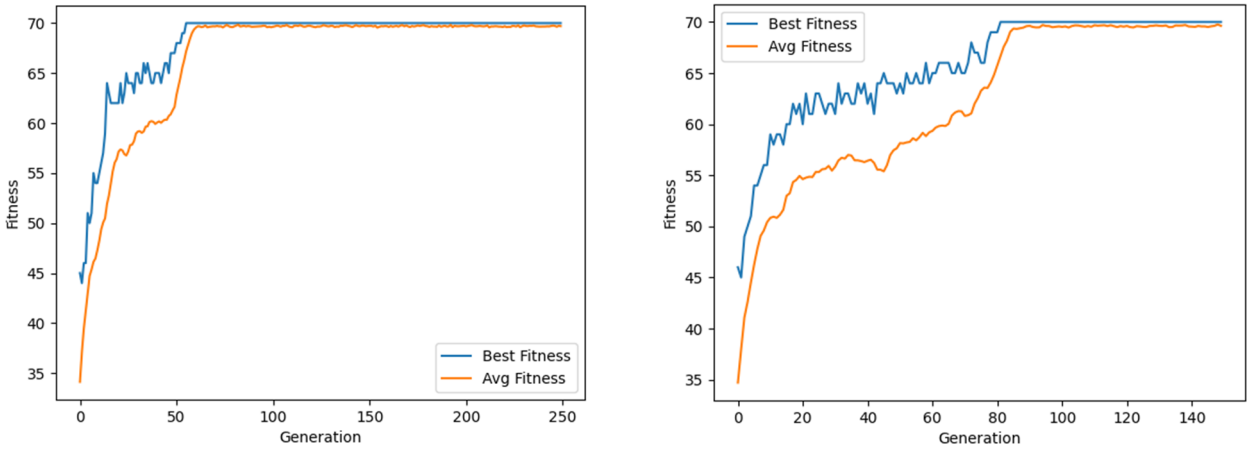
И с т о ч н и к: выполнено Д.А. Рогачевым.

```
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
```

Рис. 8. Фрагмент кода дефаззификации

И с т о ч н и к: выполнено Д.А. Рогачевым.

Типичные результаты работы GA для разных значений размера популяции представлены на рис. 9. В качестве выходного значения определяется лучшее рассчитанное значение фитнес-функции, получаемое в последнем поколении, и отрисовывает графики лучшего значения ФФ индивида и ее среднее значение по всей популяции.



a *b*

Рис. 9. Результаты работы GA с нечетким контроллером:
а — chrom_length = 70, pop_size = 70; *б* — chrom_length = 70, pop_size = 90
И с т о ч н и к: выполнено Д.А. Рогачевым.

На рис. 9 алгоритм успешно сходится к глобальному оптимуму задачи OneMax (фитнес-функция = 70, когда все биты = 1). Это ожидаемо для простой задачи с длиной хромосомы 70 при 150 поколениях, но нечеткий контроллер помогает адаптировать параметры, например, при низком разнообразии (популяция сходится). Нечеткий контроллер увеличивает P_m для добавления случайности, а при улучшении фитнес-функция — усиливает P_c для exploitation. В среднем сходимость происходит за 80…100 поколений (на основе типичных запусков), но из-за стохастичности инициализации алгоритма результат может варьироваться.

Если запустить несколько раз, среднее значение фитнес-функции растет быстро в начале (благодаря высокой начальной P_c), затем стабилизируется. Нечеткая адаптация делает GA более робастным по сравнению с архитектурой с фиксированными параметрами.

Возможно применение средства адаптации в форме нечеткого логического «контроллера», программно/аппаратно реализующего систему нечетких продукционных правил логического вывода.

Заключение

Проведенные на основе системного подхода исследования структур, параметров и результатов функционирования GA позволили обобщить направления модификации GA. Исследование влияния структуры и параметров GA на время и точность оптимизации функции приспособленности, проведенные на задаче OneMax с бинарным кодированием хромосомы, позволили сформулировать рекомендации по выбору макропараметров и обоснованно выбирать варианты модификации алгоритмов для конкретных предметных областей, включая распределение ограниченных водных ресурсов в острозасушливых условиях. Решение задачи эволюционно-генетической оптимизации показало применимость нечеткого контроллера для GA. Численными экспериментами показано, что среднее значение фитнес-функции быстро растет в начале процесса оптимизации благодаря высокому значению начальной вероятности P_c), затем стабилизируется. Нечеткая адаптация делает GA более робастным по сравнению с фиксированными параметрами.

Список литературы

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, 2023. p. 888–893. http://doi.org/10.1109/RusAutoCon58002.2023.10272907 EDN: PWNESZ

2. *Melikhova E.V., Rogachev A.F.* Computer simulation and optimization of parameters of configuration of the contour of moistening under drip irrigation of agricultures // Ubiquitous Computing and the Internet of Things: Prerequisites for the Development of ICT. Studies in Computational Intelligence / Popkova E., (ed). Cham : Springer Publ.; 2019. Vol. 826. P.1193–1201. http://doi.org/10.1007/978-3-030-13397-9_122 EDN: FJWSYW

3. *Melikhova E.V., Rogachev A.F, Skiter N.N.* Information system and database for simulation of irrigated crop growing // Ubiquitous Computing and the Internet of Things: Prerequisites for the Development of ICT. Studies in Computational Intelligence / Popkova E. (ed). Cham : Springer Publ.; 2019. Vol. 826. P. 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 // Advances in Intelligent Systems and Computing / Silhavy R., Senkerik R., Oplatkova Z.K., Silhavy P., Prokopova Z. (eds). 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. Vol. 5. No. 4. P. 1–33. https://doi.org/10.1186/s40411-017-0037-x

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

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

8. *Grefenstette J.J.* Lamarkian learning in multi-agent environment // Proceedings of the 4th international conference on genetic algorithms. 1991. P. 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. Vol. 866. P. 5–15. https://doi.org/10.1007/3-540-58484-6_245

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

11. *Moscato P., Norman M.G.* A memetic approach for the traveling salesman problem: implementation of computational ecology for combinatorial optimization on message-passing systems // In: Fogarty T.C. (eds) Parallel computing and transputer applications. Berlin, Heidelberg : Springer Publ.; 1992.Vol. 1. P. 177–186.

12. *Radcliffe N.J., Surry P.D.* Formal memetic algorithm // Evolutionary Computing.AISB EC 1994. Lecture Notes in Computer Science / Fogarty T.C. (eds). Berlin, Heidelberg : Springer Publ.; 1994. Vol. 865. P. 1–16 https://doi.org/10.1007/3-540-58483-8_1

13. *Singh B.K., Misra A.K.* Software effort estimation by genetic algorithm tuned parameters of modified constructive cost model for NASA software projects // International Journal of Computer Applications. 2012. Vol. 59. No. 9. P. 22–26. https://doi.org/10.5120/9577-4053

14. *Herera F., Lozano M.* Adaptation of genetic algorithm parametres based on fuzzy logic controllers // Genetic Algorithms and Soft Computing / Herera F., Verdegay J. (eds). 1996. p. 95–125.*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

15. *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. p. 65–69. http://doi.org/10.1109/ICEC.1997.592270

16. *Davis L.* Handbook of genetic algorithms. New York : Van Nostrand Rein-hold; 1991. ISBN10 0442001738

17. *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

18. *Koumousis V.K., Katsaras C.P.* Asow-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance // IEEE Transactions on Evolutionary Computation. 2006. Vol. 10. No. 1. P. 19–28. http://doi.org/10.1109/TEVC.2005.860765

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

20. *Moel M.C., Stewart C.V., Kelly R.B.* 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

21. *Gladkov L.A., Gladkova N.V., Semushin E.Y.* Parallel hybrid genetic algorithm for solving design and optimization problems // Advances in Intelligent Systems and Computing / Hu Z., Petoukhov S., He M. (eds). 2020. Vol. 1127. P. 249–258.2020. Vol. 1127. P. 249–258. http://doi.org/10.1007/978-3-030-39216-1_23 EDN: PMLJYG

22. *Kureychick V.M., Kaplunov T.G.* Time series forecasting method based on genetic algorithm for predicting the conditions of technical systems // Journal of Physics: Conference Series. 2019. Article no. 032046. http://doi.org/10.1088/1742-6596/1333/3/032046 EDN: OZIEUE

23. *Лабинский А.Ю.* Перспективные направления использования генетических алгоритмов // Природные и техногенные риски (физико-математические и прикладные аспекты). 2023. № 2 (46). С. 81–88. EDN: SXNYDF

24. *Рогачев Д.А., Кирейчева Л.В., Юрченко И.Ф. Тимошкин А.Д., Кузнецов Ю.С., Фролина Е.А.* Устройство для управления обучением нейронной сети с генетическим алгоритмом. Патент Российская Федерация № 2843987 C1, МПК G06N 3/08, № 2024120049: заявл. 17.07.2024 : опубл. 23.07.2025. EDN: ZSPIAC

Сведения об авторах

Рогачев Дмитрий Алексеевич, кандидат технических наук, ведущий научный сотрудник, Федеральный научный центр гидротехники и мелиорации имени А.Н. Костякова, Российская Федерация, 127434, г. Москва, ул. Большая Академическая, д. 44, корп. 2; eLIBRARY SPIN-code: 1948-7354, ORCID: 0009-0003-4014-4770; e-mail: Rogachev.soft@gmail.com

Рогачев Алексей Фруминович, доктор технических наук, профессор кафедры математического моделирования и информатики, Волгоградский государственный аграрный университет, Российская Федерация, 400005, г. Волгоград, пр-т Ленина, д. 28; eLIBRARY SPIN-код: 8413-5020, ORCID: 0000-0002-3077-6622; e-mail: rafr@mail.ru