ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 10**, № 2(41), с. 33–65

ББК 3818.1:В185.42
УДК 004.023::519.853.4

К. В. Пушкарев

Глобальная оптимизация на основе нейросетевой аппроксимации инверсных зависимостей с эволюционным управлением параметрами

Аннотация. Представлен гибридный метод глобальной оптимизации НАИЗ-PSO на основе нейросетевой аппроксимации инверсных зависимостей (координат от значений целевой функции) и метода роя частиц, служащий для нахождения глобального минимума непрерывной целевой функции многих переменных в области, имеющей вид многомерного параллелепипеда. Целевая функция рассматривается как абстрактная вычислительная процедура («чёрный ящик»).

Метод использует группы пробных точек, движущихся как в методе роя частиц. Одна из возможных целей движения определяется через отображение пониженных значений целевой функции в координаты посредством модифицированных дуальных обобщённо-регрессионных нейронных сетей, конструируемых по пробным точкам.

Параметрами процесса управляет эволюционный алгоритм. В алгоритме управления популяция состоит из эволюционирующих правил, заключающих в себе наборы параметров. Для оценки приспособленности особи используются две числовые характеристики: краткосрочная (очарование) и долгосрочная (достоинство). По очарованию правила отбираются для размножения и применения. Достоинством определяется выживание особи при формировании новой популяции. Двойная оценка правил решает проблему вымирания потенциально полезных особей при краткосрочном изменении ситуации.

Преимущество эволюционного управления над случайным изменением параметров НАИЗ-PSO в процессе поиска, а также тенденция к уменьшению погрешности при повторном использовании базы правил показаны на тестовых задачах с целевыми функциями 100 переменных.

Ключевые слова и фразы: глобальная оптимизация, эвристические методы, эволюционные методы, нейронные сети, установка параметров, управление параметрами, метод роя частиц.

10.25209/2079-3316-2019-10-2-33-65

Введение

Глобальная оптимизация целевых функций (ЦФ) многих переменных, рассматриваемых как «чёрные ящики» при минимальных требованиях к свойствам функции, является актуальной проблемой, поскольку обещает универсальный подход к решению многих практических задач (инженерных, машинного обучения и т. д.).

В настоящее время активно развивается парадигма вычислений, вдохновлённых природой, в которой вычислительные задачи решаются на основе подражания природным явлениям. В оптимизации этот подход породил направление эволюционных алгоритмов (эволюционное программирование, эволюционные стратегии, генетические алгоритмы, дифференциальная эволюция) [1, 2] и дал целый ряд оптимизационных алгоритмов, основанных на различных физических, химических и биологических явлениях [3], таких как имитация отжига и метод роя частиц (PSO).

Хотя при абсолютном отсутствии требований к ЦФ эффективность всех методов в среднем по всем возможным задачам одинакова [4], представляет интерес поиск таких работающих в более узком классе практически возможных задач методов, которые извлекали бы максимальную пользу из той информации, которую они получают о ЦФ, вычисляя её значения в пробных точках. Важным шагом к этой цели видится долговременное сохранение имеющей значение для поиска глобального минимума информации о ЦФ.

Эффективность алгоритмов оптимизации существенно зависит от выбора параметров. Оптимальный выбор параметров сам по себе является задачей оптимизации, целевой функцией в которой является та или иная оценка эффективности в зависимости от параметров. Установка параметров может происходить вручную или автоматически, перед запуском алгоритма (задача настройки параметров, parameter tuning) или в процессе его работы (задача управления параметрами, parameter control) [5].

Настройка параметров вручную требует от пользователя понимания их влияния на поведение алгоритма в конкретной ситуации, что, вообще говоря, требует экспертных знаний и опыта. При этом данные о влиянии и взаимосвязи параметров, полученные в одних задачах, могут быть неприменимы к другим. Экспериментальное исследование этого влияния осложняется многократным ростом вычислительных затрат, по сравнению с исходной задачей, так как для оценки качества

набора параметров, в общем случае, требуется решить исходную задачу при данных параметрах не менее одного раза, что сопряжено с многократным вычислением исходной ЦФ.

Оптимальный набор параметров может отличаться не только в разных задачах и при разных критериях качества, но и на разных этапах вычислительного процесса [**6**, с. 1]. Например, для метода оптимизации роем частиц было предложено постепенное снижение коэффициента инерции или увеличение размера окрестности частицы в популяционной топологии в процессе работы алгоритма [**7**, с. 36, 40]. Кроме того, вызывает интерес возможность механизма управления параметрами активно влиять на процесс и обучаться. В связи с этим задача управления параметрами является особенно важной.

Тема автоматической установки параметров получила большое развитие в области эволюционной оптимизации. Обзор различных подходов представлен в [**5**, **6**]. Спектр подходов широк: от случайной вариации и детерминированных эвристик до контроллеров, основанных на нечёткой логике [**8**, **9**], предопределённых эвристических правилах [**10**], обучении с подкреплением [**11**–**13**] или вспомогательном метаэволюционном алгоритме [**12**]. В [**5**] приведена классификация методов установки параметров на детерминированные (нет обратной связи от настраиваемого алгоритма, параметры меняются по заранее установленному плану), адаптивные (есть обратная связь, параметры задаются с учётом этой информации), самоадаптивные (параметры закодированы в генотипе особей настраиваемого алгоритма и эволюционируют совместно с решениями исходной задачи оптимизации).

В [**14**, **15**] был предложен эвристический метод нейросетевой аппроксимации инверсных зависимостей (НАИЗ), предназначенный для нахождения глобального минимума непрерывных ЦФ многих переменных типа «чёрного ящика» в области, имеющей вид многомерного параллелепипеда. В дальнейшем метод был включён в состав гибридного эвристического параллельного метода [**16**], а для управления параметрами метода НАИЗ использовались фиксированные эвристики без возможности обучения.

В данной работе представлен гибридный метод глобальной оптимизации на основе НАИЗ и PSO с эволюционным управлением параметрами. В этом методе НАИЗ служит для управления целью, к которой стремится частица, движущаяся по правилам PSO. Управление параметрами НАИЗ осуществляется с помощью контроллера, реализующего эволюционный алгоритм.

Особенность задачи управления параметрами в данной работе состоит в том, что параметры относятся не к алгоритму в целом, а к отдельным агентам-частицам в его составе, поэтому в каждый момент времени, вообще говоря, не существует единственного оптимального набора параметров: для каждого агента оптимальный набор может быть отличен от других.

Постановка численной задачи глобальной оптимизации состоит в следующем. Рассматривается целевая функция $\Phi(\mathbf{x})\colon \Omega \to \mathbb{R}$, которая непрерывна в ограниченной области

$$\Omega = \left\{\mathbf{x}\colon L_i \le x_i \le U_i,\ i = \overline{1, D}\right\} \subset \mathbb{R}^D. \tag{1}$$

Необходимо найти приближённое минимальное значение функции Φ^*_{min} и точку $\mathbf{x}^*_{min}$, в которой достигается минимум:

$$\Phi_{min} = \min_{\mathbf{x}\in\Omega} \Phi(\mathbf{x}) = \Phi(\mathbf{x}_{min}), \tag{2}$$

$$\Phi^*_{min} = \Phi(\mathbf{x}^*_{min}) \le \Phi_{min} + \varepsilon_\Phi, \tag{3}$$

где $\varepsilon_\Phi > 0$ и определяет погрешность.

1. Метод нейросетевой аппроксимации инверсных зависимостей

В основе метода НАИЗ лежит итеративное понижение значения ЦФ с отображением его в координаты с помощью обобщённо-регрессионных нейронных сетей (Generalized Regression Neural Network, GRNN), аппроксимирующих инверсные зависимости (координат от значения ЦФ). GRNN обучаются по примерам вида $\langle \Phi(\mathbf{x}), \mathbf{x}\rangle$, состоящим из значения ЦФ и координат, в которых оно достигается. Важным достоинством этих сетей является одношаговое обучение — сеть может быть сконструирована из примеров за один проход.

Базовый алгоритм НАИЗ состоит в следующем:

(*1*) Инициализация: $k := 1$, $P^{[k]}$ — начальное множество пробных точек из области поиска, например случайно выбранных.

(*2*) Сеть $\mathrm{GRNN}^{[k]}$ обучается отображать значения ЦФ в соответствующие координаты по примерам $\left\{\langle \Phi(\mathbf{x}), \mathbf{x}\rangle : \mathbf{x} \in P^{[k]}\right\}$.

(*3*) Из множества $P^{[k]}$ выбирается точка $\mathbf{x}^{[k]}_{min}$ с наименьшим значением ЦФ, это значение понижается на некоторую величину $\varepsilon^{[k]}_f$ (*декремент*) и отображается в координаты с помощью

GRNN: $\mathbf{x}_{min}^{*[k]} = \mathrm{GRNN}^{[k]}\left(\Phi(\mathbf{x}_{min}^{[k]}) - \varepsilon_f^{[k]}, s_\varphi\right)$, где s_φ — параметр сглаживания, который регулирует гладкость аппроксимации.

(4) Новая точка включается во множество $P^{[k+1]}$, если значение ЦФ в ней меньше текущего наименьшего:

$$P^{[k+1]} = \begin{cases} P^{[k]} \cup R^{[k]} \cup \left\{\mathbf{x}_{min}^{*[k]}\right\}, & \text{если } \Phi(\mathbf{x}_{min}^{*[k]}) < \Phi(\mathbf{x}_{min}^{[k]}) \\ P^{[k]} \cup R^{[k]}, & \text{иначе,} \end{cases} \tag{4}$$

где $R^{[k]}$ — множество пробных точек, созданных случайным образом.

(5) $k := k + 1$.

(6) Шаги 2–5 повторяются до достижения условий останова.

Данные шаги для функции $y = x^2$ проиллюстрированы на рисунке 1.

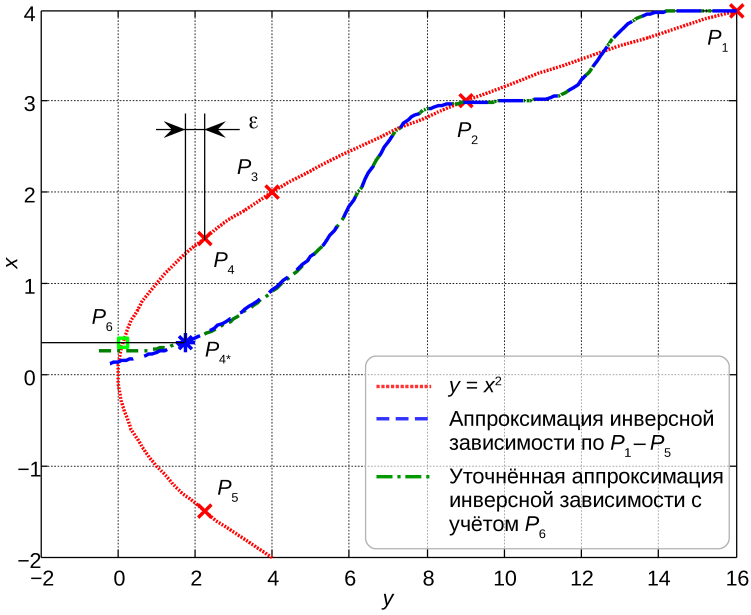

Рисунок 1. Иллюстрация принципа работы базового алгоритма НАИЗ

На рисунке 1 по пробным точкам P_1–P_5 построена аппроксимация инверсной зависимости с помощью GRNN. С её помощью получена новая точка P_6, с учётом которой построена уточнённая аппроксимация инверсной зависимости.

Применение вышеописанного метода на практике сопряжено с рядом трудностей. Его эффективность быстро падает с ростом числа переменных. В качестве стартовой на каждой итерации используется только одна (лучшая) пробная точка, а случайное вбрасывание является недостаточно эффективным источником дополнительных пробных точек, из-за чего поиск может зайти в тупик. Для преодоления этой проблемы метод НАИЗ был включён в состав гибридного эвристического параллельного метода [16], в котором поиск глобального минимума выполняется оптимизационными агентами, которые итеративно запускают ряд алгоритмов оптимизации (поисковых инструментов), обменивающихся информацией через общее множество пробных точек.

Когда ЦФ имеет несколько близких по значению функции минимумов, то линия инверсной зависимости может пройти между ними, не приблизившись ни к одному. Для преодоления этого была предложена модификация GRNN — дуальная обобщённо-регрессионная нейронная сеть (Dual GRNN, DGRNN) [17]. У этой сети, помимо входа φ для значения ЦФ, есть дополнительный вход $\mathbf{x}_f$, на который подаются координаты точки — *фокуса поиска*. Каждый пример при обучении DGRNN состоит из входного *эталона* φ_i и выходного $\mathbf{X}_i$. Чем дальше от фокуса находится запомненный нейросетью эталон, тем ниже его активация и, соответственно, вклад в результат. Таким образом, эталоны, находящиеся вблизи фокуса, приобретают преимущество. DGRNN имеет дополнительный параметр сглаживания s_x, являющийся аналогом s_φ для второго входа. В первом скрытом слое DGRNN формируются показатели близости входных значений к эталонам, которые перемножаются для каждой пары из входного и выходного эталона во втором слое. Полученные таким образом объединённые показатели близости используются для взвешенного суммирования выходных эталонов в третьем и четвёртом слоях.

На рисунке 2 приведена схема DGRNN для двух пар эталонов. Подписаны веса связей, отличные от 1.

Наконец, проблемой является управление параметрами: ε_f, s_φ, s_x, фокусом поиска. В [16] DGRNN не использовалась, а для управления ε_f и s_φ применялись фиксированные эвристики без возможности обучения. Так, декремент лучшего значения ЦФ вычислялся по формуле $\varepsilon_f^{[k]} = 3(\Phi_{max}^{[k]} - \Phi_{min}^{[k]})/N_p$, где $\Phi_{max}^{[k]}$ и $\Phi_{min}^{[k]}$ — соответственно наибольшее и наименьшее значение ЦФ по множеству пробных точек на k-й итерации, N_p — количество пробных точек. Для упрощения настройки s_φ в первом слое GRNN функция нахождения расстояния была

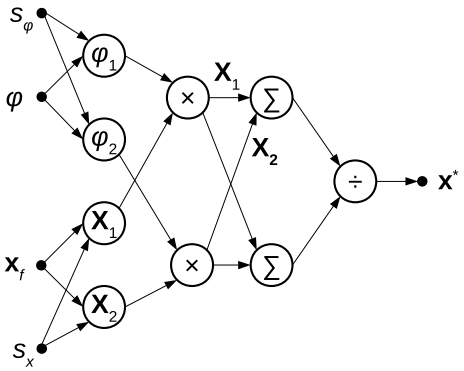

Рисунок 2. Схема DGRNN для двух пар эталонов

изменена на следующую:

$$\operatorname{dist}^{[k]}(\varphi_1, \varphi_2) = \frac{|\varphi_1 - \varphi_2|}{\Phi^{[k]}_{max} - \Phi^{[k]}_{min}}. \tag{5}$$

Это делает расстояния относительными и позволяет задавать s_φ в относительных величинах.

2. Гибридный метод НАИЗ-PSO с эволюционным управлением параметрами

Для повышения эффективности метод НАИЗ был соединён с методом PSO и дополнен эволюционным управлением параметрами.

Аналогом частиц PSO в описываемом методе являются *базовые точки* (БТ). Количество БТ N_b является параметром. Каждая БТ на k-й итерации характеризуется положением $\mathbf{b}^{[k]}$, скоростью $\mathbf{v}^{[k]}$ и *собственной целью* $\mathbf{g}_l^{[k]}$. Вначале

$$\mathbf{b}^{[1]} = \mathbf{g}_l^{[1]} = \mathrm{U}\left[\mathbf{L}, \mathbf{U}\right], \tag{6}$$

$$\mathbf{v}^{[1]} = \mathrm{U}\left[-k_{v1}\left(\mathbf{U} - \mathbf{L}\right), k_{v1}\left(\mathbf{U} - \mathbf{L}\right)\right], \tag{7}$$

где $\mathbf{L} = (L_1, \ldots, L_D)$, $\mathbf{U} = (U_1, \ldots, U_D)$ — границы области поиска; $\mathrm{U}\left[\mathbf{l}, \mathbf{u}\right]$ — вектор случайных чисел, равномерно распределённых в области $\left\{\mathbf{x}\colon l_i \le x_i \le u_i,\ i = \overline{1, D}\right\}$; k_{v1} — параметр алгоритма.

БТ делятся на непересекающиеся группы по S_{bg} штук, в каждой из которых определяется *групповая цель* $\mathbf{g}_g^{[k]}$, которая на каждой итерации выбирается по наименьшему значению ЦФ из списка: $\mathbf{g}_g^{[k-1]}$,

собственные цели БТ данной группы. Затем групповая цель может быть улучшена локальным поиском и экстраполяцией траектории, как описано далее.

Во избежание стагнации процесса, с периодичностью I_{rest} итераций происходит перезапуск неактивных БТ путём повторной случайной инициализации, описанной выше. БТ перезапускается на k-й итерации, если одновременно выполняются условия

$$\|\mathbf{v}^{[k]}\| < v_{min}, \tag{8}$$

$$\Phi_{min,b}\left(k - I_{rest}\right) - \Phi_{min,b}\left(k\right) < \varepsilon_b I_{rest}, \tag{9}$$

где $\Phi_{min,b}\left(k\right) = \min_{i=\overline{1,k}} \Phi(\mathbf{b}^{[i]})$ — минимум значений ЦФ по всем положениям БТ с 1-й по k-ю итерацию, v_{min}, ε_b, I_{rest} — параметры алгоритма.

Для метода НАИЗ-PSO была разработана модификация DGRNN — QDGRNN (Quantile DGRNN). Чтобы ограничить диапазон изменения параметров, разности значений ЦФ между входом сети и эталонами $\varphi - \varphi_i$, вычисленные внутри QDGRNN, делятся на квантили абсолютных значений разностей порядка p_{f1} для отрицательных разностей и p_{f2} для положительных. Величины p_{f1}, p_{f2} являются входными параметрами нейросети и заменяют собой s_φ. Декремент ЦФ ε_f вычитается после деления, поэтому также выражается в относительных величинах и является входным параметром. Расстояния по координатам $\|\mathbf{x}_f - \mathbf{X}_i\|$ делятся на собственный квантиль порядка p_x, который заменяет собой s_x. В итоге QDGRNN выполняет следующее отображение:

$$\mathbf{x}^* = \frac{\sum_{i=1}^{N_e} C_i^{(\varphi x)} \mathbf{X}_i}{\sum_{i=1}^{N_e} C_i^{(\varphi x)}}; \tag{10}$$

$$C_i^{(\varphi x)} = C_i^{(\varphi)} \cdot C_i^{(x)}; \tag{11}$$

(12)

$$C_i^{(\varphi)} = \exp\left(\ln 0.5 \left(\frac{\varphi - \varphi_i}{Q\left(\left\langle |\varphi - \varphi_j| : j = \overline{1, N_e}\right\rangle, p_f\left(\varphi, \varphi_i\right)\right)} - \varepsilon_f\right)^2\right),$$

$$p_f\left(\varphi, \varphi_i\right) = \begin{cases} p_{f1}, & \text{если } \varphi < \varphi_i; \\ p_{f2}, & \text{если } \varphi \geq \varphi_i; \end{cases} \tag{13}$$

$$C_i^{(x)} = \exp\left(\ln 0.5 \left(\frac{\|\mathbf{x}_f - \mathbf{X}_i\|}{Q\left(\left\langle \|\mathbf{x}_f - \mathbf{X}_j\| : j = \overline{1, N_e}\right\rangle, p_x\right)}\right)^2\right), \tag{14}$$

(15)

где $\mathbf{x}^*$ — выходной вектор; φ — входное значение ЦФ; $\mathbf{x}_f$ — входное значение фокуса; $C_i^{(\varphi)}$, $C_i^{(x)}$ — показатели близости входных значений к соответствующим эталонам φ_i, $\mathbf{X}_i$; $C_i^{(\varphi x)}$ — объединённый показатель близости для i-й пары эталонов; N_e — количество пар эталонов; $Q\,(A, p)$ — оценка квантиля порядка p по выборке A.

Управление параметрами основано на принципах эволюционных алгоритмов. Параметры задаются *правилами*, которые скрещиваются, мутируют и проходят отбор. Популяция состоит из N_r правил. Правило содержит условия применения (в данной работе не использовались), характеристики качества и набор значений параметров (генотип): ε_f, p_{f1}, p_{f2}, p_x, α_b, P_r.

Качество (приспособленность) каждого правила выражается в виде двух числовых характеристик: *очарования (charm)* и *достоинства (merit)*. Очарование является краткосрочной оценкой правила. По этой характеристике правила отбираются для размножения и применения. Достоинство является долгосрочной оценкой и включает в себя усреднение очарования за предыдущие периоды с экспоненциальным взвешиванием. Достоинством определяется выживание особи при формировании новой популяции.

Двойная оценка правил решает проблему вымирания потенциально полезных особей при краткосрочном изменении ситуации. Пока эффективные в данный момент правила активно размножаются и применяются, правила, которые перестали быть эффективными, «спят» в ожидании возможности снова стать эффективными.

Управление параметрами происходит одновременно с поиском глобального минимума. Итерации алгоритма делятся на *большие*, в которых происходит обязательное применение всех правил, их оценка и эволюция, и *малые*, в которых только применяются правила, выбираемые по очарованию. Большими являются первая и каждая I_{big}-я итерация.

Инициализация популяции происходит путём случайной генерации правил или загрузки сохранённой базы правил.

На каждой большой итерации к установленному количеству N_{top} БТ применяются все правила. В i-й большой итерации выбираются точки с номерами от $1+(i-1)\cdot N_{top}$ до $i\cdot N_{top}$, при достижении верхней границы счёт начинается сначала. К остальным (на малой итерации ко всем) БТ применяется одно случайное правило так, что вероятность

выбора пропорциональна k_{sel}^r, где $k_{sel} \in (0,1)$ — параметр алгоритма, а r — ранг правила в ряду по убыванию очарования, отсчитывающийся от нуля. Таким образом, все правила независимо от характеристик периодически испытываются для различных БТ, но правила с более высоким очарованием применяются чаще.

Применение правила в базовой точке $\mathbf{b}$ состоит в следующем:

(1) Сначала формируется *присоединённое множество* пробных точек

$$P_b = \left\{ \mathbf{b} + i \cdot \mathbf{p} \colon i = \overline{-N_s, N_s}, \mathbf{p} \in P_r \right\}, \tag{16}$$

где P_r — *шаблон* данного правила; $\mathbf{p}$ — *вектор шаблона*; N_s — параметр алгоритма.

(2) По присоединённому множеству обучается QDGRNN, отображающая значения ЦФ в координаты. Значение ЦФ в БТ с помощью нейросети отображается в координаты

$$\mathbf{b}^* = \mathrm{QDGRNN}\left(\Phi(\mathbf{b}), \mathbf{b}, \varepsilon_f, p_{f1}, p_{f2}, p_x\right). \tag{17}$$

При этом БТ используется в качестве фокуса поиска, а значения параметров ε_f, p_{f1}, p_{f2}, p_x задаются правилом. *Кандидатом от правила* будем называть точку $\breve{\mathbf{b}} = \mathbf{b} + \alpha_b \left(\mathbf{b}^* - \mathbf{b}\right)$, где параметр α_b также задаётся правилом, а *прогрессом кандидата* — разность $\rho = \Phi(\mathbf{b}) - \Phi(\breve{\mathbf{b}})$.

Точки из присоединённого множества не только служат для обучения нейросети, но также рассматриваются как потенциальные решения.

После применения правил для каждой БТ определяется собственная цель $\mathbf{g}_l^{[k]}$. На каждой итерации в качестве нового значения цели выбирается лучшая по значению ЦФ точка из следующего списка: $\mathbf{g}_l^{[k-1]}$, кандидаты от правил для БТ на данной итерации, $\mathbf{b}^{[k]}$. Наконец, каждая БТ сдвигается в сторону цели по правилу, аналогичному используемому в PSO:

$$\mathbf{v}^{[k]} = \omega_i \mathbf{v}^{[k-1]} + \omega_l \mathbf{R}_1 \otimes \left(\mathbf{g}_l^{[k]} - \mathbf{b}^{[k]}\right) + \omega_g \mathbf{R}_2 \otimes \left(\mathbf{g}_g^{[k]} - \mathbf{b}^{[k]}\right), \tag{18}$$

$$\mathbf{b}^{[k+1]} = \mathbf{b}^{[k]} + \mathbf{v}^{[k]}, \tag{19}$$

где k — номер итерации; ω_i, ω_l, ω_g — параметры алгоритма; $\mathbf{g}_l^{[k]}$ — собственная цель БТ; $\mathbf{g}_g^{[k]}$ — групповая цель; $\mathbf{R}_1$, $\mathbf{R}_2$ — векторы равномерно распределённых случайных чисел от 0 до 1; $\otimes$ — покомпонентное произведение векторов.

Правила оцениваются путём сравнения друг с другом по прогрессу их кандидатов для выбранных БТ на больших итерациях. Принцип состоит в том, что одно первое место по прогрессу кандидата (максимальный прогресс) даёт правилу более высокое очарование, чем ни одного первого при любом количестве вторых и т. д. Кроме того, правило, давшее только отрицательные прогрессы, получает меньшее очарование, чем правило, давшее хотя бы один положительный прогресс. Достигается это следующим образом: на большой итерации каждому правилу в результате его применения ко всем выбранным БТ сопоставляется вектор очков $(\theta_1, \ldots, \theta_{2N_r})$, в котором

$$\theta_i = \sum_{j=1}^{N_{top}} \delta_{s_j i}, \qquad s_i = \begin{cases} r_i, & \text{если } \rho_i \geq 0; \\ r_i + N_r, & \text{если } \rho_i < 0, \end{cases} \tag{20}$$

где N_{top} — количество выбранных БТ (параметр алгоритма); δ_{ij} — символ Кронекера; ρ_i — прогресс кандидата от данного правила для i-й выбранной БТ; r_i — ранг ρ_i в списке упорядоченных по убыванию прогрессов кандидатов для i-й выбранной БТ, отсчитываемый от нуля.

Затем правила упорядочиваются по убыванию вектора очков в словарном порядке. Пусть r — ранг правила в полученном списке, отсчитывающийся от нуля, тогда очарование правила

$$charm^{[k+1]} = k_{cd}^r, \tag{21}$$

где $k_{cd} \in (0, 1)$ — параметр алгоритма.

Достоинство правила определяется по формуле:

$$merit^{[k+1]} = merit^{[k]} \cdot k_{md}^a + charm^{[k]} + ppe^{[k]}, \tag{22}$$

где $k_{md} \in (0, 1)$ — параметр алгоритма; a — возраст правила в итерациях (отсчитывается от 0 и увеличивается на 1 в конце каждой итерации); $ppe^{[k]}$ — отношение среднего прогресса кандидатов от данного правила на итерацию к максимальному среднему прогрессу на итерацию среди всех правил.

Новые правила порождаются посредством случайной генерации, мутации и скрещивания. Вероятность выбора правила в качестве родителя для мутации и скрещивания пропорциональна его очарованию.

При случайной генерации каждый скалярный параметр правила β выбирается как равномерно распределённое случайное число в интервале $\left[l_{rand}^{(\beta)}, u_{rand}^{(\beta)}\right]$. Количество векторов шаблона равномерно распределено от 1 до D. Для определения каждого вектора шаблона

сначала генерируются равномерно распределённые случайные числа $p_{01}, \ldots, p_{0D} \in [-1, 1]$. Затем компоненты вектора шаблона определяются по формуле: $p_i = k_{mxp} \min_{j=\overline{1,D}} \left(\frac{U_j - L_j}{|p_{0j}|} \right) p_{0i}$, где k_{mxp} — коэффициент, равный k_{mxp1} для доли k_{mxpf} от общего числа создаваемых случайных правил и k_{mxp2} для остальных. Таким образом, у вектора шаблона один компонент по модулю равен $k_{mxp}\,(U_j - L_j)$, а остальные по модулю не больше этой величины. Использование двух значений k_{mxp} позволяет генерировать случайные правила с маленькими и большими векторами шаблона.

При мутации создаётся новое правило путём случайных возмущений параметров родительского правила. Поскольку мутация в данном алгоритме порождает новую особь, то она является бесполым размножением, как в эволюционном программировании. Значение скалярного параметра потомка в результате мутации является случайной величиной с усечённым нормальным распределением

$$\beta_o \sim \mathcal{N}_{l_{mut}^{(\beta)}, +\infty} \left(\beta_p, \sigma_{mut}^{(\beta)} \right), \tag{23}$$

где β_o, β_p — значение параметра соответственно у потомка и родителя; $\mathcal{N}_{l,u}\,(\mu, \sigma)$ — усечённое нормальное распределение с математическим ожиданием μ и среднеквадратичным отклонением σ на интервале $[l, u]$; $l_{mut}^{(\beta)}$, $\sigma_{mut}^{(\beta)}$ — параметры алгоритма, устанавливаемые независимо для каждого параметра правил β.

Для векторов шаблона используется нормальное распределение

$$p_{o,ij} \sim \mathcal{N} \left(p_{p,ij}, \sigma_{mut}^{(p)} \left(U_j - L_j \right) \right), \tag{24}$$

где $p_{o,ij}$, $p_{p,ij}$ — значение j-го компонента i-го вектора шаблона соответственно у потомка и родителя; $\mathcal{N}\,(\mu, \sigma)$ — нормальное распределение с математическим ожиданием μ и среднеквадратичным отклонением σ; $\sigma_{mut}^{(p)}$ — параметр алгоритма.

Правила, порождённые мутацией, проходят проверку на жизнеспособность: потомок отбрасывается, если у него хотя бы один из параметров p_{f1}, p_{f2}, p_x меньше либо равен нулю.

Скрещивание скалярных параметров происходит по формуле $\beta_o = u\beta_{p1} + (1 - u)\beta_{p2}$, где β_o, β_{p1}, β_{p2} — значение параметра соответственно у потомства, первого и второго родителя, $u \in [0, 1]$ — случайный равномерно распределённый коэффициент, выбираемый для каждого параметра независимо. Скрещивание множеств векторов шаблона происходит путём случайного равномерного выбора пар векторов

шаблонов первого и второго родителя, после чего соответствующие компоненты векторов скрещиваются как скаляры, как описано выше. Количество векторов шаблона потомства определяется как случайное число с биномиальным распределением $B(n, 0.5)$, где n — сумма размеров шаблонов родителей.

На каждой большой итерации формируется новая популяция из N_r правил. Сначала создаётся $N_{nr} = \lfloor (1 - k_{elt})\, N_r \rfloor$ новых правил, среди которых $\lfloor k_{rand} N_{nr} \rfloor$ случайных, $\lfloor k_{cros} N_{nr} \rfloor$ получены скрещиванием и $\lfloor k_{mut} N_{nr} \rfloor$ — мутацией. Из шаблонов новых правил удаляются векторы, норма которых меньше ε_{pat}. Затем старые правила упорядочиваются по убыванию достоинства. Из этого списка $\lfloor k_{elt} N_r \rfloor$ первых правил добавляются в новую популяцию (элита). Новые правила проверяются на корректность. Правило считается корректным, если p_{f1}, p_{f2}, p_x положительны и шаблон не пуст. Некорректные правила удаляются.

В качестве вспомогательных мер используются локальный поиск и экстраполяция траектории групповой цели.

Локальный поиск применяется к каждой групповой цели с установленной периодичностью I_{loc} итераций. В данной работе использовался модифицированный алгоритм BFGS [**18**, с. 136] с остановкой после 100 итераций или падения нормы градиента ниже 10^{-12}. Последнее значение приближения обратной матрицы Гессе H_k, сформированное алгоритмом BFGS, сохраняется и восстанавливается при следующем запуске локального поиска. Если групповая цель сильно сдвигается между запусками локального поиска, то запомненное значение H_k становится неактуальным. Сделанная в данной работе модификация алгоритма обеспечивает забывание H_k следующим образом: если на очередной итерации не происходит улучшение решения, следующее приближение обратной матрицы Гессе определяется по формуле $H_{k+1} = k_h H_k + (1 - k_h)E$, где $k_h \in [0, 1)$ — параметр алгоритма (в экспериментах, описанных в данной работе, имел значение 0.75), E — единичная матрица. Для линейного поиска в алгоритме BFGS применялся метод золотого сечения с остановкой, если число итераций больше 10 или выполнены условия Вольфе с коэффициентами $c_1 = 10^{-4}$, $c_2 = 0.9$ [**18**, с. 33]. Линейный поиск перезапускался с укорочением отрезка в 10 раз, если в текущей точке значение ЦФ больше, чем в начальной.

Для экстраполяции траектории групповой цели $\mathbf{g}_g^{*[k]}$ на k-й итерации формируется множество из $n \le N_{ext}$ последних её лучших положений $G_t^{[k]} = \left\{ \mathbf{g}_g^{[k_1]}, \ldots, \mathbf{g}_g^{[k_n]} \right\}$, где $k > k_1 > \cdots$, $\Phi(\mathbf{g}_g^{[k_1]}) < \cdots < \Phi(\mathbf{g}_g^{[k_n]})$,

N_{ext} — параметр алгоритма. После чего делаются пробные шаги в точки $P_{ext} = \left\{ \mathbf{g}_g^{*[k]} + \left(\mathbf{g}_g^{*[k]} - \mathbf{g}_g^{[k_i]} \right) : i = \overline{1,n} \right\}$ и выбирается новая групповая цель $\mathbf{g}_g^{[k]} = \arg\min_{\mathbf{x} \in \left\{ \mathbf{g}_g^{*}[k] \right\} \cup P_{ext}} \Phi(\mathbf{x})$.

Процесс поиска глобального минимума продолжается до выполнения условий останова. Проверка условий происходит каждые I_{stop} итераций. Остановка происходит, если выполнено любое из условий: 1) среднее убывание лучшего значения ЦФ за I_{stop} последних итераций меньше ε_{stop}, 2) среднее перемещение точки с лучшим значением ЦФ за I_{stop} последних итераций меньше δ_{stop}; 3) число итераций больше I_{max}. После завершения процесса итоговая популяция правил сохраняется в виде базы правил, которая может быть загружена в начале нового процесса.

3. Эксперименты

Для оценки эффективности метода были проведены эксперименты по решению тестовых задач минимизации при 100 переменных ($D = 100$). За основу были взяты известные тестовые ЦФ [19]. Во всех задачах глобальный минимум расположен в точке $(0, 0, \ldots)$ со значением ЦФ 0.

По умолчанию выбирались области поиска от -100 до 100 по каждой координате. Для некоторых задач брались области, распространённые в литературе. Для функции Экли эффективность рассматриваемого метода существенно зависит от размера области, поэтому было выбрано две области поиска: от -100 до 100 и от -50 до 50. Это можно объяснить тем, что у функции Экли при большом удалении от глобального минимума начинает преобладать колебательная составляющая, которая практически полностью скрывает глобальную тенденцию.

Использовались следующие тестовые задачи:

(*1*) Функция Растригина в области $[-100; 100]^D$:

$$f_1(\mathbf{x}) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right].$$

(*2*) Сдвинутая функция Розенброка в области $[-100; 100]^D$:

$$f_2(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100 \left((x_{i+1} + 1) - (x_i + 1)^2 \right)^2 + x_i^2 \right].$$

(3) Функция Экли в области $[-100; 100]^D$:

$$f_3(\mathbf{x}) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e.$$

(4) Функция Экли в области $[-50; 50]^D$.

(5) Функция Гриванка в области $[-600; 600]^D$:

$$f_4(\mathbf{x}) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1.$$

(6) Функция Вейерштрасса в области $[-0.5; 0.5]^D$:

$$f_5(\mathbf{x}) = \sum_{i=1}^{D}\left[\sum_{k=0}^{k_{max}} a^k \cos\left(2\pi b^k (x_i + 0.5)\right)\right] - D\sum_{k=0}^{k_{max}} a^k \cos\left(\pi b^k\right),$$

где $a = 0.5$, $b = 3$, $k_{max} = 20$.

(7) Функция Кацууры в области $[-100; 100]^D$:

$$f_6(\mathbf{x}) = 10D^{-2}\prod_{i=1}^{D}\left(1 + i\sum_{j=1}^{32}\left|2^j x_i - \text{round}\left(2^j x_i\right)\right| 2^{-j}\right)^{10D^{-1.2}} - 10D^{-2},$$

где $\text{round}(x)$ — округление к ближайшему целому.

(8) Сдвинутая функция Леви в области $[-100; 100]^D$:

$$f_7(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + \\ + (w_D - 1)^2\left[1 + \sin^2(2\pi w_D)\right],$$

где $w_i = 1 + 0.25x_i$.

(9) Эллиптический параболоид вращения в области $[-100; 100]^D$:

$$f_8(\mathbf{x}) = \sum_{i=1}^{D} x_i^2.$$

(10) Диск в области $[-100; 100]^D$:

$$f_9(\mathbf{x}) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2.$$

(11) Изогнутая сигара в области $[-100;100]^D$:

$$f_{10}(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2.$$

(12) Сумма разных степеней в области $[-100;100]^D$:

$$f_{11}(\mathbf{x}) = \sum_{i=1}^{D} |x_i|^{i+1}.$$

(13) Эллиптическая функция с высокой обусловленностью в области $[-100;100]^D$:

$$f_{12}(\mathbf{x}) = \sum_{i=1}^{D} 10^{6(i-1)/(D-1)} x_i^2.$$

Классические тестовые функции часто обладают «удобными» свойствами: глобальный минимум находится в нуле или другой точке с равными координатами или минимум можно найти независимо по отдельным координатам:

$$\mathbf{x}_{min} = \left(\underset{L_1 \le x \le U_1}{\arg\min}\, \Phi(x, x_2, \ldots), \underset{L_2 \le x \le U_2}{\arg\min}\, \Phi(x_1, x, \ldots), \ldots \right). \tag{25}$$

Поэтому ко всем базовым ЦФ были применены случайные сдвиги и ортогональные преобразования: $f_i^*(\mathbf{x}) = f_i(M(\mathbf{x} - \mathbf{x}_0))$, $\mathbf{x}_0 = \mathrm{U}\left[\mathbf{L} + 0.4\left(\mathbf{U} - \mathbf{L}\right), \mathbf{L} + 0.6\left(\mathbf{U} - \mathbf{L}\right)\right]$. Матрица M получена путём ортонормализации матрицы случайных чисел, равномерно распределённых от 0 до 1, по алгоритму Грама-Шмидта. Аналогичный подход к созданию тестовых функций применён в [19].

Были проведены эксперименты следующих типов:

(1) Контрольные эксперименты со случайным изменением параметров. Для каждой задачи алгоритм запускался 100 раз со случайной инициализацией. База правил, M, $\mathbf{x}_0$ при каждом запуске инициализировались случайно. Использовался описанный выше алгоритм управления параметрами, из которого была исключена эволюция. Очарованию и достоинству правил при каждой процедуре оценки присваивались случайные числа, равномерно распределённые от 0 до 1. Пополнение базы происходило только за счёт случайных правил, а мутация и скрещивание не применялись.

(2) Эксперименты для общей оценки эффективности метода. Для каждой задачи алгоритм запускался 100 раз со случайной

инициализацией. База правил, M, $\mathbf{x}_0$ при каждом запуске инициализировались случайно.

(3) Эксперименты для выявления долговременной адаптации. Для каждой задачи было проведено 50 серий по 10 запусков алгоритма. База правил инициализировалась случайно в начале серии, после каждого запуска она сохранялась и загружалась вновь в начале следующего запуска (*сквозная база правил*). M, $\mathbf{x}_0$ инициализировались случайно перед первым и далее каждые R_t запусков в серии. Эксперименты были проведены при $R_t = 0$ (M, $\mathbf{x}_0$ не обновляются внутри серии) и $R_t = 1$.

Для каждого запуска алгоритма определялись найденное минимальное значение ЦФ (Φ_{min}), количество вычислений ЦФ (N_{eval}), количество итераций (N_{iter}) и количество вычислений ЦФ на итерацию ($N_{epi} = N_{eval}/N_{iter}$).

Результаты экспериментов со случайным изменением параметров использовались для контроля эффективности предлагаемого эволюционного алгоритма управления, так как известно [**20**], что в некоторых случаях само по себе изменение параметров в процессе работы алгоритма оптимизации может улучшать результаты, по сравнению со статическими параметрами. В данной работе случайное изменение параметров было реализовано путём модификации эволюционного алгоритма управления, чтобы свести отличия между сравниваемыми конфигурациями к единственному фактору — способу выбора параметров.

В экспериментах использовались следующие значения параметров:

- размера и состава популяции: $N_r = 100$, $k_{elt} = 0.25$, $k_{cros} = 0.5$, $k_{mut} = 0.25$, $k_{rand} = 0.25$;
- мутации: $l_{mut}^{(\varepsilon_f)} = l_{mut}^{(p_{f1})} = l_{mut}^{(p_{f2})} = l_{mut}^{(p_x)} = 10^{-2}$, $\sigma_{mut}^{(\varepsilon_f)} = \sigma_{mut}^{(p_{f1})} = \sigma_{mut}^{(p_{f2})} = \sigma_{mut}^{(p_x)} = \sigma_{mut}^{(\alpha_b)} = 0.05$, $l_{mut}^{(\alpha_b)} = 10^{-2}$, $\sigma_{mut}^{(\alpha_b)} = 0.05$, $\sigma_{mut}^{(p)} = 5 \times 10^{-3}$;
- оценки правил: $k_{cd} = 0.95$, $k_{md} = 0.9999$;
- применения правил: $I_{big} = 10$, $k_{sel} = 0.95$, $N_{top} = 10$, $N_s = 1$;
- случайной генерации правил: $k_{mxp1} = 0.2$, $k_{mxp2} = 10^{-6}$, $k_{mxpf} = 0.5$, $\varepsilon_{pat} = 10^{-6}$, $l_{rand}^{(\varepsilon_f)} = 10^{-2}$, $u_{rand}^{(\varepsilon_f)} = 10$, $l_{rand}^{(p_{f1})} = l_{rand}^{(p_{f2})} = l_{rand}^{(p_x)} = 10^{-2}$, $u_{rand}^{(p_{f1})} = u_{rand}^{(p_{f2})} = u_{rand}^{(p_x)} = 1$, $l_{rand}^{(\alpha_b)} = 10^{-2}$, $u_{rand}^{(\alpha_b)} = 10$;
- локального поиска: $I_{loc} = 25$, $k_h = 0.75$;
- базовых точек: $N_b = 100$, $k_{v1} = 1$, $S_{bg} = 10$, $v_{min} = 10^{-5}$, $\varepsilon_b = 5 \times 10^{-6}$, $I_{rest} = 10$, $\omega_i = 0.9$, $\omega_l = 0.5$, $\omega_g = 0.5$, $N_{ext} = 10$;
- условий останова: $I_{stop} = 100$, $\varepsilon_{stop} = 10^{-7}$, $\delta_{stop} = 0$, $I_{max} = \infty$.

4. Результаты

Итоговая статистика для экспериментов со случайным изменением (СИ) и эволюционным управлением (ЭУ) параметрами без загрузки базы правил приведена соответственно в таблицах 1 и 2.

Таблица 1. Результаты экспериментов с СИ

Задача	Параметр	Минимум	Медиана	Максимум	Среднее	Среднекв. откл.
1	Φ^*_{min}	196	326.8	448.7	329.6	58.45
	N_{eval}	9.817×10^{6}	1.238×10^{7}	2.468×10^{7}	1.224×10^{7}	2.162×10^{6}
	N_{iter}	400	500	1000	490	87.75
2	Φ^*_{min}	1.691×10^{-12}	3.744×10^{-11}	3.987	1.316	1.875
	N_{eval}	7.356×10^{6}	7.663×10^{6}	1.257×10^{7}	8.545×10^{6}	1.262×10^{6}
	N_{iter}	300	300	500	340	50.99
3	Φ^*_{min}	20	20	20	20	0.000 200 3
	N_{eval}	4.451×10^{6}	4.681×10^{6}	1.206×10^{7}	5.309×10^{6}	1.41×10^{6}
	N_{iter}	200	200	500	228	58.45
4	Φ^*_{min}	1.315×10^{-8}	19.81	19.91	19.2	3.378
	N_{eval}	4.42×10^{6}	4.866×10^{6}	1.771×10^{7}	6.826×10^{6}	2.828×10^{6}
	N_{iter}	200	200	700	285	113.5
5	Φ^*_{min}	1.221×10^{-14}	3.847×10^{-14}	2.244×10^{-13}	5.116×10^{-14}	3.622×10^{-14}
	N_{eval}	4.86×10^{6}	5.033×10^{6}	5.277×10^{6}	5.033×10^{6}	7.935×10^{4}
	N_{iter}	200	200	200	200	0
6	Φ^*_{min}	71.3	93.86	106.3	93.57	6.629
	N_{eval}	1.421×10^{7}	1.9×10^{7}	5.104×10^{7}	2.164×10^{7}	7.776×10^{6}
	N_{iter}	600	800	2100	893	316.6
7	Φ^*_{min}	0.017 05	0.0621	0.3598	0.079 02	0.061 23
	N_{eval}	6.731×10^{6}	1.134×10^{7}	1.409×10^{7}	1.059×10^{7}	1.497×10^{6}
	N_{iter}	300	500	600	459	61.8
8	Φ^*_{min}	4343	9905	1.225×10^{4}	9622	1535
	N_{eval}	4.692×10^{6}	7.331×10^{6}	9.822×10^{7}	1.137×10^{7}	1.193×10^{7}
	N_{iter}	200	300	3900	463	474.1
9	Φ^*_{min}	5.492×10^{-16}	1.843×10^{-15}	2.566×10^{-15}	1.764×10^{-15}	4.518×10^{-16}
	N_{eval}	4.961×10^{6}	5.255×10^{6}	5.493×10^{6}	5.252×10^{6}	8.844×10^{4}
	N_{iter}	200	200	200	200	0
10	Φ^*_{min}	3.787×10^{-7}	5.37×10^{-7}	8.014×10^{-7}	5.362×10^{-7}	8.902×10^{-8}
	N_{eval}	4.655×10^{6}	4.913×10^{6}	5.112×10^{6}	4.909×10^{6}	9.112×10^{4}
	N_{iter}	200	200	200	200	0
11	Φ^*_{min}	2.172×10^{-9}	1.879×10^{-7}	4.297×10^{-6}	5.649×10^{-7}	8.42×10^{-7}
	N_{eval}	4.948×10^{6}	5.17×10^{6}	7.846×10^{6}	5.609×10^{6}	9.719×10^{5}
	N_{iter}	200	200	300	218	38.42
12	Φ^*_{min}	1.381×10^{-10}	1.144×10^{-7}	0.000 75	1.284×10^{-5}	7.662×10^{-5}
	N_{eval}	3.262×10^{7}	5.563×10^{7}	1.011×10^{8}	5.803×10^{7}	1.656×10^{7}
	N_{iter}	1100	1900	3500	1976	580.2
13	Φ^*_{min}	3.87×10^{-7}	1.066×10^{-6}	4.416×10^{-6}	1.185×10^{-6}	5.696×10^{-7}
	N_{eval}	4.89×10^{6}	5.096×10^{6}	7.596×10^{6}	5.137×10^{6}	3.564×10^{5}
	N_{iter}	200	200	300	202	14

Таблица 2. Результаты экспериментов с ЭУ без загрузки базы правил

Задача	Параметр	Минимум	Медиана	Максимум	Среднее	Среднекв. откл.
1	Φ^*_{min}	5.862×10^{-14}	3.428×10^{-13}	54.72	15.68	17.44
	N_{eval}	8.175×10^{6}	9.86×10^{6}	1.677×10^{7}	1.072×10^{7}	2.073×10^{6}
	N_{iter}	300	300	600	358	73.73
2	Φ^*_{min}	1.606×10^{-12}	2.991×10^{-11}	3.987	1.037	1.749
	N_{eval}	8.071×10^{6}	9.312×10^{6}	1.522×10^{7}	1.002×10^{7}	1.523×10^{6}
	N_{iter}	300	300	500	339	50.78
3	Φ^*_{min}	20	20	20	20	0.000 319 6
	N_{eval}	4.15×10^{6}	5.458×10^{6}	1.441×10^{7}	6.26×10^{6}	1.739×10^{6}
	N_{iter}	200	200	500	239	63.08
4	Φ^*_{min}	8.689×10^{-9}	19.76	19.93	17.58	6.182
	N_{eval}	4.211×10^{6}	7.423×10^{6}	1.876×10^{7}	7.969×10^{6}	3.271×10^{6}
	N_{iter}	200	300	800	296	118.3
5	Φ^*_{min}	1.044×10^{-14}	3.825×10^{-14}	4.118×10^{-13}	5.061×10^{-14}	4.683×10^{-14}
	N_{eval}	5.194×10^{6}	6.228×10^{6}	6.819×10^{6}	6.193×10^{6}	3.46×10^{5}
	N_{iter}	200	200	200	200	0
6	Φ^*_{min}	39.17	73.61	87.24	72.53	9.259
	N_{eval}	1.751×10^{7}	2.312×10^{7}	1.056×10^{8}	3.031×10^{7}	1.767×10^{7}
	N_{iter}	600	750	2900	956	498.7
7	Φ^*_{min}	0.009 948	0.066 76	0.2953	0.082 15	0.052 62
	N_{eval}	7.304×10^{6}	1.177×10^{7}	2.132×10^{7}	1.175×10^{7}	2.315×10^{6}
	N_{iter}	300	500	800	468	81.09
8	Φ^*_{min}	4.268	9257	1.178×10^{4}	7998	3726
	N_{eval}	4.664×10^{6}	1.064×10^{7}	8.407×10^{7}	1.332×10^{7}	1.2×10^{7}
	N_{iter}	200	400	2800	479	421.5
9	Φ^*_{min}	4.741×10^{-16}	1.195×10^{-15}	2.149×10^{-15}	1.236×10^{-15}	4.056×10^{-16}
	N_{eval}	5.094×10^{6}	6.505×10^{6}	7.379×10^{6}	6.483×10^{6}	4.42×10^{5}
	N_{iter}	200	200	200	200	0
10	Φ^*_{min}	3.54×10^{-7}	5.437×10^{-7}	7.794×10^{-7}	5.491×10^{-7}	8.739×10^{-8}
	N_{eval}	4.857×10^{6}	5.465×10^{6}	6.317×10^{6}	5.487×10^{6}	3.104×10^{5}
	N_{iter}	200	200	200	200	0
11	Φ^*_{min}	1.617×10^{-9}	2.423×10^{-7}	7.641×10^{-6}	5.306×10^{-7}	9.742×10^{-7}
	N_{eval}	5.512×10^{6}	6.435×10^{6}	1.014×10^{7}	6.69×10^{6}	9.355×10^{5}
	N_{iter}	200	200	300	209	28.62
12	Φ^*_{min}	8.579×10^{-11}	3.468×10^{-7}	0.004 809	8.098×10^{-5}	0.000 542 7
	N_{eval}	3.623×10^{7}	5.498×10^{7}	1.267×10^{8}	6.038×10^{7}	1.812×10^{7}
	N_{iter}	1100	1600	3800	1778	543.1
13	Φ^*_{min}	3.427×10^{-7}	1.03×10^{-6}	2.895×10^{-6}	1.085×10^{-6}	4.042×10^{-7}
	N_{eval}	5.155×10^{6}	5.95×10^{6}	8.479×10^{6}	6.033×10^{6}	4.624×10^{5}
	N_{iter}	200	200	300	202	14

Результаты экспериментов с СИ и ЭУ без загрузки базы правил сопоставлены на рисунке 3–5 и в таблице 3. На рисунке 3 данные представлены в виде коробчатых диаграмм: прямоугольники показывают

область между первым (Q_1) и третьим (Q_3) квартилями, оранжевая линия — медиана, выносные элементы охватывают область значений с отклонением не более $1.5(Q_3 - Q_1)$ ниже Q_1 или выше Q_3, более далёкие значения (выбросы) отмечены круглыми маркерами, зелёные треугольные маркеры показывают среднее. На рисунках 4, 5 планки погрешности показывают стандартную ошибку.

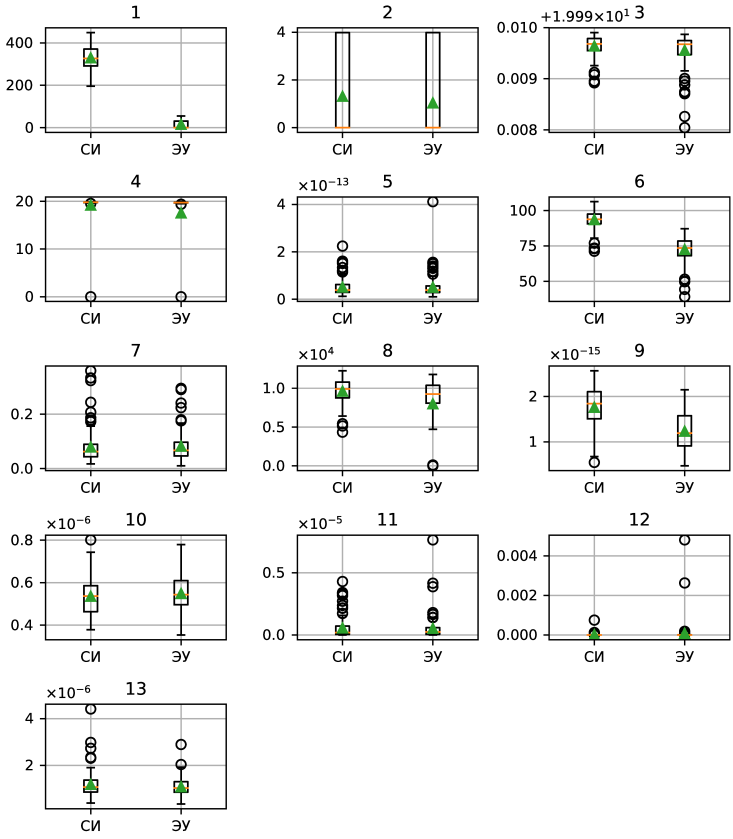

Рисунок 3. Φ^*_{min} в экспериментах с СИ и ЭУ без загрузки базы правил

Значимость отличий результатов по параметрам Φ^*_{min}, N_{eval}, N_{epi} проверялась с помощью двустороннего критерия Манна—Уитни[1].

[1]Использовалась функция `scipy.stats.mannwhitneyu` из библиотеки SciPy [21]

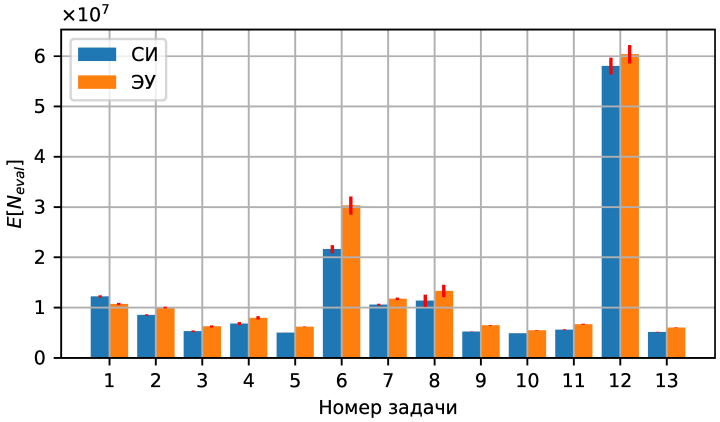

Рисунок 4. Средние N_{eval} в экспериментах с СИ и ЭУ без загрузки базы правил

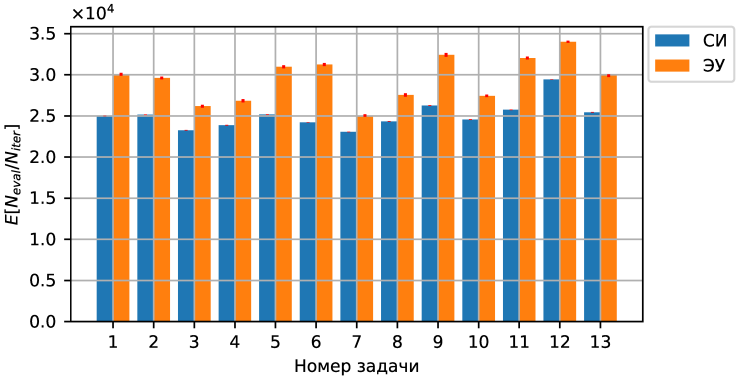

Рисунок 5. Средние N_{epi} в экспериментах с СИ и ЭУ без загрузки базы правил

Для каждого из трёх параметров в отдельности делалась поправка p-значений на множественность сравнений по методу Холма—Бонферрони[2].

версии 0.19.1.

[2]Здесь и далее использовалась функция `p.adjust` с параметром `method = "holm"` из статистической среды R [22] версии 3.4.4.

Таблица 3. Результаты сравнения Φ^*_{min} в экспериментах с СИ и ЭУ без загрузки базы правил

Задача	P-значение	P-значение с поправкой
1	<0.0001	<0.0001*
2	0.0893	0.6850
3	0.0856	0.6850
4	0.0004	0.0042*
5	0.7545	1
6	<0.0001	<0.0001*
7	0.2785	1
8	0.0160	0.1444
9	<0.0001	<0.0001*
10	0.2433	1
11	0.9912	1
12	0.2009	1
13	0.3242	1

Примечание: *) статистически значимые результаты ($p < 0.05$).

Как следует из таблицы 3, различие Φ^*_{min} между экспериментами с СИ и ЭУ является статистически значимым при уровне значимости 0.05. Значимые отличия наблюдались в задачах № 1, 4, 6, 9. В этих задачах минимальные значения ЦФ в экспериментах с СИ были выше, чем с ЭУ (рисунок 3). Дополнительно следует обратить внимание на задачу № 8. В ней исправленное p-значение 0.14 оказалось выше уровня значимости, однако 15 значений в режиме с ЭУ оказались меньше 50 (наименьшее 4.268), в то время как наименьший результат с СИ составляет 4343.

В режиме с ЭУ наблюдается статистически значимое ($p < 0.05$) увеличение (в задачах № 2–11, 13) или уменьшение (№ 1) N_{eval} (рисунок 4). В задаче № 12 значимых отличий не выявлено. Для всех задач значимо увеличивается N_{epi} (рисунок 5).

Наиболее существенным различие между режимами с СИ и ЭУ было в задаче № 1 с ЦФ Растригина. Одновременно со значительным уменьшением погрешности, в режиме с ЭУ также уменьшилось количество вычислений ЦФ и итераций, чего не наблюдалось в других задачах.

Следовательно, результаты экспериментов не противоречат гипотезе о том, что в некоторых задачах при эволюционном управлении

Таблица 4. Результаты сравнения Φ^*_{min} между первым и последним запуском в серии в экспериментах с ЭУ и сквозной базой правил в серии

Задача	$R_t = 0$		$R_t = 1$	
	P-значение	P-значение с поправкой	P-значение	P-значение с поправкой
1	<0.0001	<0.0001*	<0.0001	<0.0001*
2	0.7030	1	0.1410	0.9333
3	0.0002	0.0022*	0.0012	0.0105*
4	<0.0001	<0.0001*	<0.0001	<0.0001*
5	0.5955	1	0.1333	0.9333
6	<0.0001	<0.0001*	<0.0001	<0.0001*
7	0.3466	1	0.6958	1
8	0.0022	0.0180*	0.0002	0.0022*
9	0.9961	1	0.2408	1
10	0.8734	1	0.4602	1
11	0.6191	1	0.5023	1
12	0.4840	1	0.9654	1
13	<0.0001	0.0002*	0.0341	0.2728

Примечание: *) статистически значимые результаты ($p < 0.05$).

параметрами метода НАИЗ-PSO Φ^*_{min} ниже, чем при случайном изменении параметров.

Итоговая статистика для последнего запуска в серии в экспериментах с ЭУ и сквозной базой правил в режимах со случайным преобразованием ЦФ в начале серии ($R_t = 0$) и перед каждым запуском ($R_t = 1$) приведена соответственно в таблицах 5 и 6.

Результаты экспериментов с ЭУ и сквозной базой правил в серии в режимах $R_t = 0$ и $R_t = 1$ сопоставлены на рисунках 6–9 и в таблице 4. На рисунках 7–9 затенённые области показывают стандартную ошибку.

Значимость отличий результатов по параметрам Φ^*_{min}, N_{eval}, N_{epi} между первым и последним запуском в серии проверялась с помощью двустороннего знаково-рангового критерия Уилкоксона для парных выборок[3]. Для каждого из трёх параметров в отдельности делалась поправка p-значений на множественность сравнений по методу Холма—Бонферрони.

[3]Использовалась функция `scipy.stats.wilcoxon` с параметром `alternative = 'two-sided'` из библиотеки SciPy [21] версии 0.19.1.

Таблица 5. Результаты последнего запуска в серии в экспериментах с ЭУ и сквозной базой правил при $R_t = 0$

Задача	Параметр	Минимум	Медиана	Максимум	Среднее	Среднекв. откл.
1	Φ^*_{min}	6.395×10^{-14}	9.415×10^{-14}	1.634×10^{-13}	1.006×10^{-13}	2.451×10^{-14}
	N_{eval}	1.075×10^{7}	1.171×10^{7}	1.234×10^{7}	1.162×10^{7}	4.48×10^{5}
	N_{iter}	300	300	300	300	0
2	Φ^*_{min}	2.191×10^{-12}	3.458×10^{-11}	3.987	1.037	1.749
	N_{eval}	9.954×10^{6}	1.111×10^{7}	1.675×10^{7}	1.202×10^{7}	1.882×10^{6}
	N_{iter}	300	300	500	340	56.57
3	Φ^*_{min}	20	20	20	20	0.000 609 6
	N_{eval}	4.734×10^{6}	6.804×10^{6}	1.28×10^{7}	7.36×10^{6}	2.116×10^{6}
	N_{iter}	200	200	400	258	75.07
4	Φ^*_{min}	8.239×10^{-9}	1.779×10^{-8}	19.86	6.682	9.31
	N_{eval}	5.553×10^{6}	1.306×10^{7}	2.688×10^{7}	1.344×10^{7}	4.848×10^{6}
	N_{iter}	200	400	800	414	154.9
5	Φ^*_{min}	1.232×10^{-14}	3.531×10^{-14}	2.001×10^{-13}	4.908×10^{-14}	4.164×10^{-14}
	N_{eval}	6.779×10^{6}	7.703×10^{6}	8.103×10^{6}	7.685×10^{6}	3.045×10^{5}
	N_{iter}	200	200	200	200	0
6	Φ^*_{min}	29.23	42.8	52.7	42.91	5.748
	N_{eval}	2.017×10^{7}	3.231×10^{7}	7.287×10^{7}	3.696×10^{7}	1.242×10^{7}
	N_{iter}	500	800	1800	912	307
7	Φ^*_{min}	0.018 59	0.063 55	0.2023	0.077 94	0.047 29
	N_{eval}	7.554×10^{6}	1.332×10^{7}	2.502×10^{7}	1.382×10^{7}	3.14×10^{6}
	N_{iter}	400	500	800	472	84.95
8	Φ^*_{min}	4573	7670	1.141×10^{4}	7610	1625
	N_{eval}	5.139×10^{6}	1.515×10^{7}	8.605×10^{7}	1.873×10^{7}	1.412×10^{7}
	N_{iter}	200	450	2200	570	392.6
9	Φ^*_{min}	5.241×10^{-16}	1.192×10^{-15}	2.145×10^{-15}	1.198×10^{-15}	3.635×10^{-16}
	N_{eval}	7.097×10^{6}	7.904×10^{6}	8.587×10^{6}	7.883×10^{6}	3.433×10^{5}
	N_{iter}	200	200	200	200	0
10	Φ^*_{min}	3.2×10^{-7}	4.975×10^{-7}	8.597×10^{-7}	5.253×10^{-7}	1.154×10^{-7}
	N_{eval}	4.731×10^{6}	5.474×10^{6}	6.203×10^{6}	5.469×10^{6}	3.423×10^{5}
	N_{iter}	200	200	200	200	0
11	Φ^*_{min}	2.003×10^{-9}	2.199×10^{-7}	1.65×10^{-5}	8.341×10^{-7}	2.444×10^{-6}
	N_{eval}	7.014×10^{6}	7.874×10^{6}	1.222×10^{7}	7.942×10^{6}	6.694×10^{5}
	N_{iter}	200	200	300	202	14
12	Φ^*_{min}	7.689×10^{-11}	2.102×10^{-7}	0.000 694 7	1.686×10^{-5}	9.707×10^{-5}
	N_{eval}	3.901×10^{7}	6.153×10^{7}	1.137×10^{8}	6.553×10^{7}	1.751×10^{7}
	N_{iter}	1100	1750	3600	1800	523.5
13	Φ^*_{min}	4.134×10^{-7}	9.624×10^{-7}	1.968×10^{-6}	9.968×10^{-7}	3.303×10^{-7}
	N_{eval}	5.578×10^{6}	6.512×10^{6}	7.56×10^{6}	6.493×10^{6}	4.409×10^{5}
	N_{iter}	200	200	200	200	0

Как следует из таблицы 4, статистически значимые отличия Φ^*_{min} между первым и последним запуском в серии при уровне значимости 0.05 наблюдались как при $R_t = 0$, так и при $R_t = 1$. При этом в задачах

Таблица 6. Результаты последнего запуска в серии в экспериментах с ЭУ и сквозной базой правил при $R_t = 1$

Задача	Параметр	Минимум	Медиана	Максимум	Среднее	Среднекв. откл.
1	Φ^*_{min}	4.796×10^{-14}	1.075×10^{-13}	2.363×10^{-13}	1.058×10^{-13}	3.215×10^{-14}
	N_{eval}	1.084×10^{7}	1.162×10^{7}	1.241×10^{7}	1.158×10^{7}	3.96×10^{5}
	N_{iter}	300	300	300	300	0
2	Φ^*_{min}	1.456×10^{-12}	1.868×10^{-11}	3.987	0.3987	1.196
	N_{eval}	9.828×10^{6}	1.097×10^{7}	1.804×10^{7}	1.14×10^{7}	1.534×10^{6}
	N_{iter}	300	300	500	316	41.76
3	Φ^*_{min}	20	20	20	20	0.000 451 7
	N_{eval}	4.5×10^{6}	6.291×10^{6}	1.371×10^{7}	7.163×10^{6}	2.146×10^{6}
	N_{iter}	200	200	400	252	64
4	Φ^*_{min}	7.62×10^{-9}	2.076×10^{-8}	19.86	7.934	9.636
	N_{eval}	5.842×10^{6}	1.275×10^{7}	2.632×10^{7}	1.33×10^{7}	5.439×10^{6}
	N_{iter}	200	400	700	416	151.5
5	Φ^*_{min}	8.216×10^{-15}	3.014×10^{-14}	1.402×10^{-13}	4.073×10^{-14}	2.919×10^{-14}
	N_{eval}	7.109×10^{6}	7.672×10^{6}	8.296×10^{6}	7.674×10^{6}	2.754×10^{5}
	N_{iter}	200	200	200	200	0
6	Φ^*_{min}	31.01	43.91	54.41	42.82	4.86
	N_{eval}	2.425×10^{7}	3.221×10^{7}	6.958×10^{7}	3.442×10^{7}	9.846×10^{6}
	N_{iter}	600	800	1700	848	241.9
7	Φ^*_{min}	0.015 14	0.061 95	0.2733	0.080 82	0.056 64
	N_{eval}	7.564×10^{6}	1.351×10^{7}	2.24×10^{7}	1.379×10^{7}	3.266×10^{6}
	N_{iter}	300	500	800	468	88.18
8	Φ^*_{min}	55.9	7049	1.001×10^{4}	6852	2034
	N_{eval}	5.144×10^{6}	1.739×10^{7}	6.373×10^{7}	1.95×10^{7}	1.118×10^{7}
	N_{iter}	200	600	1700	594	299.6
9	Φ^*_{min}	2.653×10^{-16}	1.149×10^{-15}	2.311×10^{-15}	1.151×10^{-15}	4.224×10^{-16}
	N_{eval}	6.766×10^{6}	7.846×10^{6}	8.454×10^{6}	7.831×10^{6}	3.881×10^{5}
	N_{iter}	200	200	200	200	0
10	Φ^*_{min}	3.488×10^{-7}	5.145×10^{-7}	8.542×10^{-7}	5.298×10^{-7}	9.497×10^{-8}
	N_{eval}	4.412×10^{6}	5.495×10^{6}	6.564×10^{6}	5.498×10^{6}	3.806×10^{5}
	N_{iter}	200	200	200	200	0
11	Φ^*_{min}	3.557×10^{-9}	3.093×10^{-7}	9.728×10^{-6}	7.629×10^{-7}	1.571×10^{-6}
	N_{eval}	6.495×10^{6}	7.887×10^{6}	1.2×10^{7}	7.972×10^{6}	8.765×10^{5}
	N_{iter}	200	200	300	206	23.75
12	Φ^*_{min}	2.277×10^{-10}	2.427×10^{-7}	0.005 974	0.000 123 7	0.000 835 8
	N_{eval}	3.3×10^{7}	6.262×10^{7}	1.433×10^{8}	6.693×10^{7}	2.334×10^{7}
	N_{iter}	900	1700	4200	1854	689.1
13	Φ^*_{min}	2.684×10^{-7}	8.417×10^{-7}	2.753×10^{-6}	9.329×10^{-7}	4.336×10^{-7}
	N_{eval}	5.121×10^{6}	6.492×10^{6}	7.811×10^{6}	6.53×10^{6}	4.678×10^{5}
	N_{iter}	200	200	200	200	0

№ 1, 3, 4, 6, 8 отличие наблюдается в обоих режимах, а в задаче № 13 только при $R_t = 0$. В этих задачах отчётливо видна тенденция к понижению Φ^*_{min} в серии рисунков 6, 7.

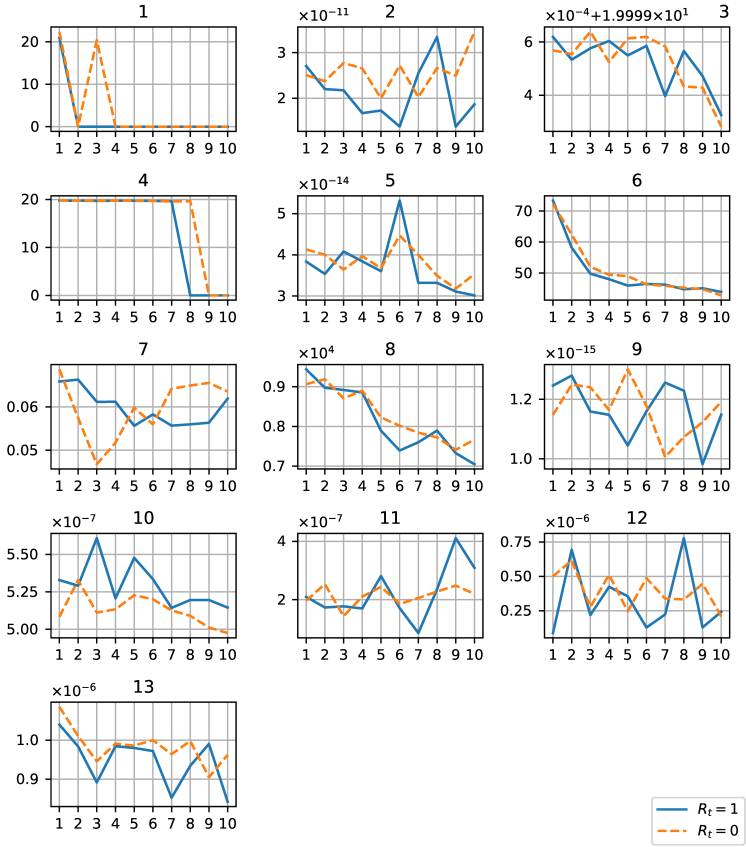

Рисунок 6. Медианы Φ^*_{min} в зависимости от номера запуска в серии в экспериментах с ЭУ и сквозной базой правил в серии

В большинстве задач N_{eval} возрастает в серии рисунка 8, при этом статистически значимыми ($p < 0.05$) отличия между первым и последним запуском в серии являются для задач № 1, 2, 4, 5, 6, 9, 11, 13 при $R_t = 0$ и № 1, 2, 3, 4, 5, 7, 9, 11, 13 при $R_t = 1$.

Значение N_{epi} возрастает в серии в обоих режимах во всех задачах, кроме № 10 (рисунок 9). Отличия по N_{epi} между первым и последним запуском в серии также статистически значимы ($p < 0.05$) в обоих режимах во всех задачах, кроме № 10. В задаче № 10 значимых

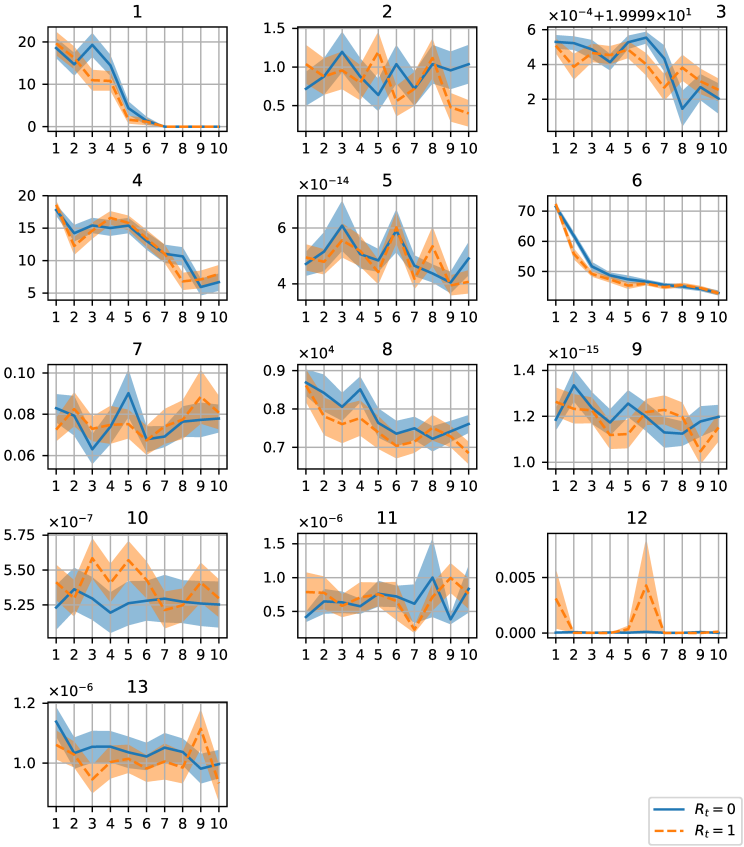

Рисунок 7. Средние Φ^*_{min} в зависимости от номера запуска в серии в экспериментах с ЭУ и сквозной базой правил в серии

отличий нет ни в одном режиме.

Сравнение режимов $R_t = 0$ и $R_t = 1$ между собой по результатам последнего запуска в серии с помощью критерия Манна—Уитни не показало значимых отличий по Φ^*_{min}, N_{eval}, N_{epi} ($p > 0.05$).

Таким образом, результаты экспериментов не противоречат гипотезе о том, что в некоторых задачах при эволюционном управлении параметрами метода НАИЗ-PSO со сквозной базой правил Φ^*_{min} ниже, чем при создании новой базы при каждом запуске.

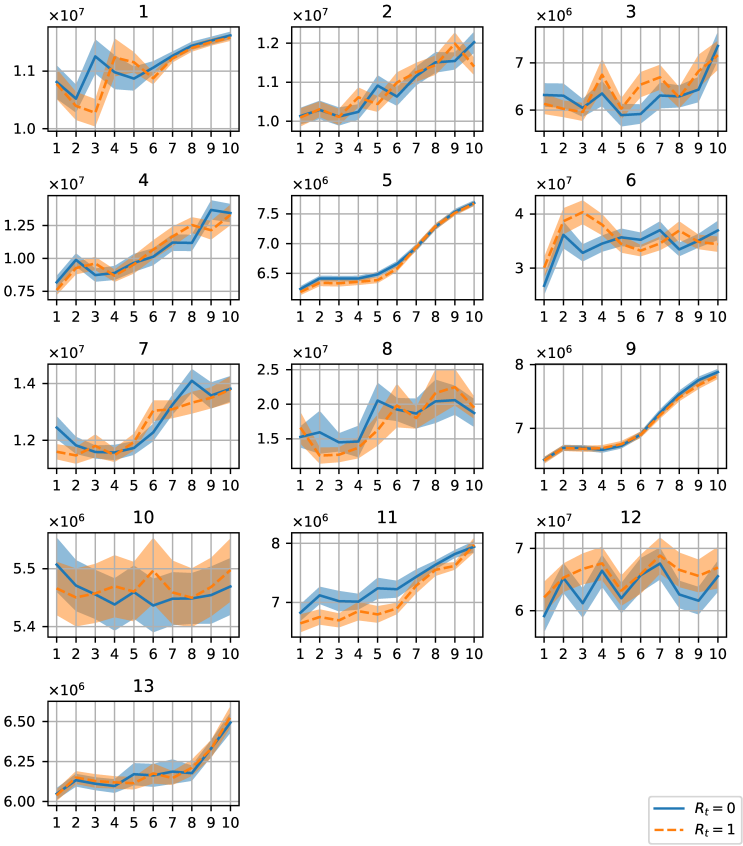

Рисунок 8. Средние N_{eval} в зависимости от номера запуска в серии в экспериментах с ЭУ и сквозной базой правил в серии

5. Выводы

Эволюционное управление параметрами повышает точность метода НАИЗ-PSO, по сравнению со случайным изменением параметров в процессе поиска. Дальнейшее уменьшение погрешности значений ЦФ в режиме со сквозной базой правил говорит о постепенной адаптации метода к решаемой задаче.

При использовании эволюционного управления и сквозной базы правил не наблюдается различий в результатах между случайным

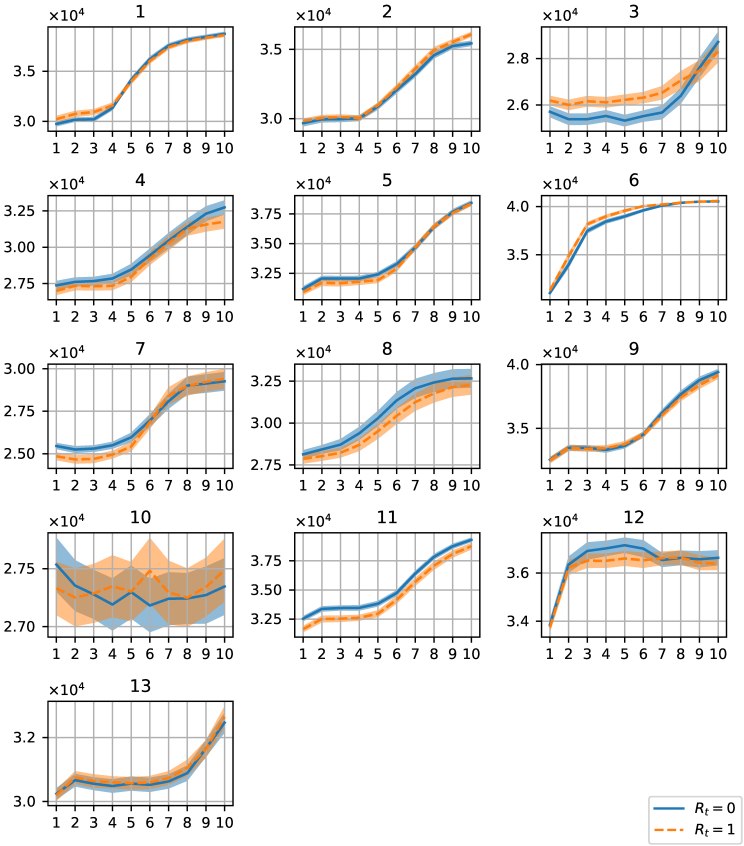

Рисунок 9. Средние N_{epi} в зависимости от номера запуска в серии в экспериментах с ЭУ и сквозной базой правил в серии

преобразованием ЦФ перед первым ($R_t = 0$) или перед каждым запуском в серии ($R_t = 1$). Следовательно, процесс устойчив к ортогональным преобразованиям и ограниченным сдвигам ЦФ.

При случайном изменении параметров наименьшую погрешность значений ЦФ метод НАИЗ-PSO показал в задачах № 5, 9–13 (по среднему), а также № 2 (по медиане). Из этих задач только в задаче № 9 эволюционное управление параметрами привело к статистически значимому снижению погрешности. Также в этих задачах, кроме № 13

при $R_t = 0$, не обнаружилось эффекта от использования сквозной базы правил в течение 10 запусков алгоритма. Общее для этих задач то, что ЦФ являются или близки к полиномиальным, в том числе квадратичным. Например, в задаче № 5 преобразованная ЦФ Гриванка при 100 переменных почти во всей области поиска, за исключением небольшой окрестности глобального минимума, является квадратичной. Высокая успешность в этих задачах, по-видимому, достигается за счёт вспомогательного квазиньютоновского локального поиска по модифицированному методу BFGS.

Метод показал хорошие результаты для сложной преобразованной функции Растригина (задача № 1). В этом случае эволюционное управление параметрами привело к значительному снижению погрешности, по сравнению со случайным изменением. В режиме со сквозной базой правил средняя погрешность быстро снижалась примерно с 19 в конце 1-го запуска до 10^{-13} в конце 7-го запуска в серии рисунка 7. Также в этом режиме хорошие результаты наблюдались для преобразованной функции Экли (задача № 4), где за 10 запусков медиана погрешности снизилась с 20 до 2×10^{-8}.

Статистическая значимость полученных результатов высокая: p-значения, как правило, были существенно ниже выбранного порога 0.05.

Эволюционное управление параметрами в методе НАИЗ-PSO обладает рядом преимуществ перед непосредственной ручной настройкой. Оно позволяет сохранять опыт решения задачи или класса задач в базе правил, использовать его повторно и постепенно наращивать. Также применение эволюционного подхода открывает путь к использованию достижений из области эволюционных вычислений.

Благодаря эволюционному контроллеру, пользователю не требуется самостоятельно настраивать параметры метода НАИЗ-PSO, хотя всё ещё требуется задать области случайной генерации и мутации и другие параметры самого контроллера (*метапараметры*). Тем не менее, эта задача представляется более простой, чем непосредственная ручная настройка, так как после начального выбора метапараметров процесс сам адаптируется к решаемой задаче. При этом не вполне оптимальный выбор метапараметров будет замедлять, но не останавливать эволюцию. В этом случае метапараметры можно корректировать, анализируя направление эволюции по базе правил.

Вопросами для дальнейших исследований являются: 1) причины эффективности или неэффективности метода НАИЗ-PSO с эволюционным управлением параметрами в различных задачах; 2) подходы

к уменьшению количества вычислений ЦФ; 3) пути сокращения количества метапараметров.

Список литературы

[1] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, H. De Garis. "An overview of evolutionary computation", European Conference on Machine Learning, Lecture Notes in Computer Science, vol. **667**, Springer, 1993, pp. 442–459. ↑34

[2] F. Neri, V. Tirronen. "Recent advances in differential evolution: a survey and experimental analysis", *Artificial Intelligence Review*, **33**:1–2 (2010), pp. 61–106. ↑34

[3] N. Siddique, H. Adeli. "Nature inspired computing: an overview and some future directions", *Cognitive computation*, **7**:6 (2015), pp. 706–714. ↑34

[4] D. H. Wolpert, W. G. Macready. "No free lunch theorems for optimization", *IEEE transactions on evolutionary computation*, **1**:1 (1997), pp. 67–82. ↑34

[5] G. Karafotias, M. Hoogendoorn, A. E. Eiben. "Parameter control in evolutionary algorithms: Trends and challenges", *IEEE Transactions on Evolutionary Computation*, **19**:2 (2015), pp. 167–187. ↑34,35

[6] A. Aleti, I. Moser. "A systematic literature review of adaptive parameter control methods for evolutionary algorithms", *ACM Computing Surveys (CSUR)*, **49**:3 (2016), 56. ↑35

[7] R. Poli, J. Kennedy, T. Blackwell. "Particle swarm optimization", *Swarm intelligence*, **1**:1 (2007), pp. 33–57. ↑35

[8] V. D. Koshur. "Reinforcement swarm intelligence in the global optimization method via neuro-fuzzy control of the search process", *Optical Memory and Neural Networks*, **24**:2 (2015), pp. 102–108. ↑35

[9] Sh. A. Akhmedova, V. V. Stanovov, E. S. Semenkin. "Cooperation of bio-inspired and evolutionary algorithms for neural network design", *Journal of Siberian Federal University. Mathematics & Physics*, **11**:2 (2018), pp. 148–158. ↑35

[10] E. Semenkin, M. Semenkina. "Self-configuring genetic algorithm with modified uniform crossover operator", *Advances in Swarm Intelligence*, ICSI 2012, Lecture Notes in Computer Science, vol. **7331**, Springer, 2012, pp. 414–421. ↑35

[11] G. Karafotias, A. E. Eiben, M. Hoogendoorn. "Generic parameter control with reinforcement learning", *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ACM, 2014, pp. 1319–1326. ↑35

[12] G. Karafotias, M. Hoogendoorn, B. Weel. "Comparing generic parameter controllers for EAs", *2014 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, IEEE, 2014, pp. 46–53. ↑35

[13] A. Rost, I. Petrova, A. Buzdalova. “Adaptive Parameter Selection in Evolutionary Algorithms by Reinforcement Learning with Dynamic Discretization of Parameter Range”, *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, ACM, 2016, pp. 141–142. ↑35

[14] В. Д. Кошур, К. В. Пушкарев. «Глобальная оптимизация на основе нейросетевой аппроксимации инверсных зависимостей», *XIII Всероссийская научно-техническая конференция «Нейроинформатика-2011»*, Сборник научных трудов: в 3 ч. Т. 1, ред. О. А. Мишулина, НИЯУ МИФИ, М., 2010, с. 89–98. ↑35

[15] V. Koshur, K. Pushkaryov. “Global optimization via neural network approximation of inverse coordinate mappings”, *Optical Memory and Neural Networks*, **20**:3 (2011), pp. 181–193. ↑35

[16] К. В. Пушкарев, В. Д. Кошур. «Гибридный эвристический параллельный метод глобальной оптимизации», *Вычислительные методы и программирование*, **16** (2015), с. 242–255. ↑35,38

[17] В. Д. Кошур, К. В. Пушкарев. «Дуальные обобщённо-регрессионные нейронные сети для решения задач глобальной оптимизации», *XII Всероссийская научно-техническая конференция «Нейроинформатика-2010»*, Сборник научных трудов: в 2 ч. Т. 2, ред. О. А. Мишулина, НИЯУ МИФИ, М., 2010, с. 219–227. ↑38

[18] J. Nocedal, S. J. Wright. *Numerical Optimization*, Second Edition, Springer-Verlag, New York, 2006. ↑45

[19] N. H. Awad, M. Z. Ali, P. N. Suganthan, J. J. Liang, B. Y. Qu. “Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization”, 2016. ↑46,48

[20] G. Karafotias, M. Hoogendoorn, A. E. Eiben. “Why parameter control mechanisms should be benchmarked against random variation”, *IEEE Congress on Evolutionary Computation*, IEEE, 2013, pp. 349–355. ↑49

[21] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*, 2001. ↑52,55

[22] R Core Team. *R: A language and environment for statistical computing*, 2018. ↑53

Поступила в редакцию 27.04.2019
Переработана 09.05.2019
Опубликована 26.06.2019

Рекомендовал к публикации *д.т.н. В. М. Хачумов*

Пример ссылки на эту публикацию:

К. В. Пушкарев. «Глобальная оптимизация на основе нейросетевой аппроксимации инверсных зависимостей с эволюционным управлением параметрами». *Программные системы: теория и приложения*, 2019, **10**:2(41), с. 33–65. 10.25209/2079-3316-2019-10-2-33-65
http://psta.psiras.ru/read/psta2019_2_33-65.pdf

Эта же статья по-английски: 10.25209/2079-3316-2019-10-2-3-31

Об авторе:

Кирилл Владимирович Пушкарев

Старший преподаватель кафедры вычислительной техники Института космических и информационных технологий Сибирского федерального университета. Научные интересы: эвристические методы глобальной оптимизации, нейронные сети.

0000-0002-1138-886X
e-mail: cyril.pushkaryov@yandex.ru

Sample citation of this publication:

Kirill V. Pushkaryov. “Global optimization via neural network approximation of inverse coordinate mappings with evolutionary parameter control”. *Program Systems: Theory and Applications*, 2019, **10**:2(41), pp. 33–65. (*In Russian*).
10.25209/2079-3316-2019-10-2-33-65
http://psta.psiras.ru/read/psta2019_2_33-65.pdf

The same article in English: 10.25209/2079-3316-2019-10-2-3-31