УДК 519.673

МОДИФИКАЦИЯ АЛГОРИТМА РОЯ ЧАСТИЦ НА ОСНОВЕ МЕТОДА АНАЛИЗА ИЕРАРХИЙ

С. А .Королев*, Д. В. Майков**

**Ижевский государственный технический университет имени М. Т. Калашникова*
***Ижевский торгово-экономический техникум*

Поступила в редакцию 08.09.2019 г.

Аннотация. Существует ряд задач математического моделирования, при решении которых возникает необходимость найти точку максимума или минимума некоторой функции многих переменных. Например, это обучение нейронных сетей, идентификация параметров математических моделей по экспериментальным данным, нахождение оптимального управления для этих моделей. Для быстрого нахождения точки экстремума разработано множество популяционных алгоритмов оптимизации (алгоритм роя частиц (PSO), генетический алгоритм и другие). В работе предложена модификация алгоритма роя частиц, основанная на методе анализа иерархий. Схема разработанного алгоритма инспирирована поведением пресноводных гидр, поэтому авторы предлагают назвать его H-алгоритмом. Выполнены последовательная и параллельная (коалгоритмическая) гибридизация разработанного алгоритма с генетическим алгоритмом, использующим вещественное кодирование (RGA). Для рассмотренных алгоритмов реализована высокоуровневая гибридизация вложением, при которой в окрестности особи с наилучшим значением целевой функции осуществляется локальный поиск с помощью метода сопряженных градиентов (обучение лидера). Сравнение скорости сходимости рассматриваемых методов выполнялось на примере различных многоэкстремальных тестовых функций (функция Розенброка, Дэвиса, Экли, Расстригина). Базовые алгоритмы PSO, RGA и H-алгоритм для различных тестовых функций показали различную, но в среднем одинаковую скорость сходимости. Обучение лидера существенно повысило скорость сходимости всех алгоритмов. Наилучшие результаты по скорости сходимости показали параллельная и последовательная гибридизация алгоритма RGA и H-алгоритма.

Ключевые слова: задача оптимизации, метод роя частиц, генетический алгоритм, метод анализа иерархий, обучение лидера, метод сопряженных градиентов, гибридизация вложением, последовательная гибридизация, коалгоритмическая гибридизация.

ВВЕДЕНИЕ

В процессе численного решения многих задач математического моделирования необходимо находить точку экстремума некоторой функции многих переменных (целевой функции). Подобной задачей является обучение нейронных сетей для аппроксимации данных в различных предметных областях [1–3]. Другим примером является идентификация по экспериментальным данным параметров систем дифференциальных уравнений, описывающих тот или иной процесс [4, 5], а также задача построения оптимального управления для таких динамических систем [6–8]. Точку экстремума функции многих переменных необходимо находить в процессе управления организационными системами [9]. Целевые функции, встречающиеся в приведенных примерах, как правило, многоэкстремальны и имеют высокую размерность области определения.

В работе рассматривается задача условной многомерной оптимизации, которая формулируется следующим образом: необходимо найти вектор переменных $\mathbf{x}$, при котором заданная функция $f(\mathbf{x})$ достигает глобального экстремума (для определенности, минимума) в некоторой замкнутой области $D \subset R^n$, т. е. вектор $\mathbf{x}^{opt} = \arg\min_{\mathbf{x} \in D} f(\mathbf{x})$. При этом целевая

функция $f(\mathbf{x})$ может быть задана не аналитически и иметь множество локальных экстремумов.

Традиционные методы локальной оптимизации (Ньютона, сопряженных градиентов [10] и т. д.) обычно не позволяют решать подобные задачи, т. к. с их помощью, как правило, удается найти только один локальный экстремум. Наибольшую эффективность при решении многоэкстремальных задач высокой размерности показали популяционные алгоритмы оптимизации [1–16]. При этом популяция является множеством особей (агентов, частиц), представляющих собой векторы из области D.

Примером популяционного алгоритма является алгоритм роя частиц (PSO), предложенный в работе [12]. В его основе лежит модель, описывающая поведение индивидов в толпе или особей в стае. При нахождении скорости движения частицы учитывается ее скорость на предыдущих итерациях, направление от текущего положения частицы к ее положению с наилучшим значением целевой функции за все время поиска, а также координаты лучшего решения (лидера) за все время поиска.

Достаточно часто в вычислительной практике применяются генетические алгоритмы, имитирующие эволюционные процессы [4, 5, 7, 9, 11, 13–16], в частности, такие как естественный отбор, скрещивание и мутацию особей.

Целью данной работы является разработка алгоритма оптимизации, обладающего хорошими показателями скорости сходимости на основе алгоритма PSO, а для некоторых целевых функций и лучшими.

МЕТОДЫ ИССЛЕДОВАНИЯ

Алгоритм роя частиц (PSO) с управлением скоростью частиц

Для работы популяционных алгоритмов случайным образом создается популяция, особи которой представляют собой возможные решения задачи $\mathbf{x}_j \ \left(j=\overline{1,S}\right)$ с координатами x_{ij}, $i=\overline{1,n}$ [1–3, 11, 12]. Пусть область D является n-мерным параллелепипедом с ограничениями на значения переменных $x_i^{\min} < x_{ij} < x_i^{\max}$, тогда для создания начальной популяции используется следующая формула:

$$x_{ij} = x_i^{\min} + \alpha_i \cdot \left(x_i^{\max} - x_i^{\min}\right), \qquad (1)$$

где α_i – случайные величины, принимающие значения из диапазона $[0, 1]$.

В алгоритме роя частиц (PSO) координаты j-й частицы на $k+1$-й итерации равны:

$$\mathbf{x}_j^{k+1} = \mathbf{x}_j^k + \mathbf{v}_j^k, \ \ j=\overline{1,S}. \qquad (2)$$

Компоненты вектора скорости движения j-й частицы находятся следующим образом:

$$v_{ij}^{k+1} = c_1 v_{ij}^k + c_2 \cdot \alpha_i \cdot \left(x_{ij}^* - x_{ij}\right) + c_3 \cdot \beta_i \cdot \left(x_i^{**} - x_{ij}\right), \qquad (3)$$

где x_{ij}^* – координаты j-й частицы с наилучшим значением целевой функции за все время поиска; x_i^{**} – координаты лучшего решения (лидера) за все время поиска; α_i и β_i – случайные величины, принимающие значения из диапазона $[0, 1]$; c_1, c_2 и c_3 – параметры алгоритма.

Первое слагаемое в правой части формулы (3) называется инерционной компонентой и служит для предотвращения резких изменений скорости частицы. Второе слагаемое – когнитивная компонента, отражает память частицы о ее наилучшем положении. Третье слагаемое получило название социальной компоненты и представляет собой память всего роя о наилучшем положении его частиц.

Недостатком классического алгоритма PSO является то, что скорость некоторых частиц может существенно возрастать и они могут покинуть область D [11]. Для предотвращения этого вводится ограничение на скорость частиц:

$$\left|v_{ij}^k\right| \le v_i^{\max}, \quad v_i^{\max} = r \cdot \left(x_i^{\max} - x_i^{\min}\right), \qquad (4)$$

где параметр r принимает значения из интервала $(0, 1)$.

Для популяционных алгоритмов существуют различные критерии окончания поиска. В данной работе было выбрано условие, когда изменение целевой функции лидера на k-й итерации (f^k) становится малым для ряда итераций:

$$\chi = \max_q \left|\frac{f^k - f^{k-q}}{f^k}\right| < 10^{-4}, \qquad (5)$$

где $q = 1,..., \min(k, \tau)$; τ – временной лаг, принимаемый равным 100.

Схема рассматриваемого алгоритма имеет вид:

1. Согласно формулы (1) создать начальную популяцию S частиц со случайными скоростями. Для каждой частицы определить значение целевой функции. Положить $k = 1$.

2. Выполнить поиск глобально оптимального решения (лидера) $\mathbf{x}^{**}$. Для каждой частицы определить локально оптимальное решение $\mathbf{x}^{*}_{j}$.

3. Для каждой частицы с помощью соотношения (3) с учетом ограничений (4) найти скорости движения и новые координаты по формуле (2). Вычислить значения целевой функции.

4. Проверить выполнение критерия останова (5). Если это условие выполняется, то завершить поиск. В противном случае положить $k = k + 1$ и перейти к пункту 2.

Генетический алгоритм с вещественным кодированием (RGA)

Генетические алгоритмы моделируют эволюционные процессы в живой природе [5, 9, 11, 13–16]. В работе использована классическая модель генетического алгоритма с вещественным кодированием (RGA), поскольку она обладает лучшей скоростью сходимости по сравнению с генетическим алгоритмом, использующим бинарное кодирование.

Используемая схема генетического алгоритма имеет вид:

1. Создать случайным образом начальную популяцию из S особей (векторов случайных чисел из промежутка $[0, 1]$). Согласно соотношения (1) найти соответствующие этим особям векторы в пространстве поиска и значения целевой функции. Положить $k = 1$.

2. Выполнить поиск глобально оптимального решения (лидера) $\mathbf{x}^{**}$.

3. Выполнить процедуру скрещивания случайно выбранных особей с вероятностью скрещивания в диапазоне 0,6–1,0. При этом из двух случайно выбранных особей найти особь с лучшим значением целевой функции (турнирный отбор). Повторить процедуру дважды и скрестить две полученные особи с помощью оператора SBX, имитирующего скрещивание для генетических алгоритмов с бинарным кодированием и характеризующегося хорошими показателями скорости сходимости [11]. Заменить родительских особей потомками и найти соответствующие векторы (1) в пространстве поиска и значения целевой функции.

4. Осуществить мутацию случайно взятой особи с вероятностью $10^{-2} - 10^{-4}$, применяя к случайно выбранной компоненте вектора особи оператор мутации Михалевича. Это позволяет предотвратить преждевременную сходимость алгоритма к локальному экстремуму и повышению обзора пространства поиска (диверсификации алгоритма)

5. Проверить выполнение критерия останова (5). Если это условие выполняется, то завершить поиск. В противном случае положить $k = k + 1$ и перейти к пункту 2.

Модификация алгоритма PSO – алгоритм гидр (H-алгоритм)

Авторами статьи предложена модификация алгоритма PSO, позволяющая повысить скорость сходимости по сравнению с базовым алгоритмом PSO. Суть модификации состоит в том, что вместо формулы (3) для определения направления движения выполняется несколько пробных шагов, для одного из которых используется метод анализа иерархий [17]. Помимо этого, имеется ряд отличий, описанных ниже.

Метод анализа иерархий служит для принятия решения в различных ситуациях. Пусть имеется p вариантов решения проблемы, предпочтения эксперта для которых задаются матрицей парных сравнений A. Ее элементы $a_{ij} \in [0,1; 10]$ показывают, насколько i-й вариант лучше j-го $\left(i, j = \overline{1, p}\right)$, причем $a_{ij} = \dfrac{1}{a_{ji}}$ и $a_{ii} = 1$.

Сначала вычисляется сумма элементов каждого столбца:

$$S_j = a_{1j} + a_{2j} + ... + a_{pj}.$$

После этого выполняется нормировка матрицы A по правилу $b_{ij} = \dfrac{a_{ij}}{S_j}$.

Приоритет каждого варианта определяется его весовым коэффициентом, равным

среднему значению элементов i-й строки матрицы B:

$$w_i = \frac{1}{p}\sum_{j=1}^{p} b_{ij}. \qquad (6)$$

В разработанном методе вариантами служат направления движения особи: текущее, к лидеру и случайное. Численные исследования показали, что хорошие показатели сходимости достигаются, когда матрица A имеет вид, представленный в табл. 1.

В этой таблице символом K обозначено характерное количество итераций, т.е. число итераций, по достижении которого целевая функция перестает существенно изменяться. В данной работе принималось $K = 1000$.

Выражения для элементов матрицы A таковы, что на начальных этапах поиска $(k << K)$ приоритет случайного направления примерно сопоставим с текущим. За счет этого обеспечивается широта обзора пространства поиска. Приоритет направления к лидеру низок, т.к. в начале вычислительного процесса сложно получить хорошее приближение к глобальному экстремуму. На завершающих этапах вычислительного процесса $(k \to K)$ происходит возрастание приоритета направления к лидеру, что позволяет увеличить концентрацию особей вблизи лидера и уточнить решение задачи.

Для перемещения особей выполняется пробный шаг в направлении текущей скорости движения частицы заданной длины $\lambda \in R^n$:

$$\mathbf{x}_j^{k+1} = \mathbf{x}_j^k + \lambda^k \otimes \mathbf{v}_j^k, \qquad (7)$$

где знак $\otimes$ символизирует поэлементное произведение векторов. Далее вычисляется значение целевой функции в этой точке. Если оно улучшилось, то полагают координаты особи равными $\mathbf{x}_j^{k+1}$, иначе вычисляется случайный вектор скорости частицы с компонентами $\mathbf{v}_j^{rand} \in [-1, 1]$ и вектор скорости в направлении к лидеру:

$$\mathbf{v}_j^{leader} = \frac{\mathbf{x}^{**} - \mathbf{x}_j}{\left\|\mathbf{x}^{**} - \mathbf{x}_j\right\|}.$$

После этого с помощью метода анализа иерархий находится вектор $\mathbf{w}$, координаты которого вычисляются по формуле (6), и определяется скорость частицы:

$$\mathbf{v}_j^{k+1} = w_1\mathbf{v}_j^k + w_2\mathbf{v}_j^{leader} + w_3\mathbf{v}_j^{rand}$$

и ее положение

$$\mathbf{x}_j^{k+1} = \mathbf{x}_j^k + \lambda^k \otimes \frac{\mathbf{v}_j^{k+1}}{\left\|\mathbf{v}_j^{k+1}\right\|}. \qquad (8)$$

Если значение целевой функции по-прежнему не улучшается, то выполняется пробный шаг в случайном направлении (генерируемом заново):

$$\mathbf{x}_j^{k+1} = \mathbf{x}_j^k + \lambda^k \otimes \mathbf{v}_j^{rand}. \qquad (9)$$

Если значение целевой функции не улучшается, то особь не перемещается.

Одной из особенностей предлагаемого алгоритма является процедура переноса особей, которая позволяет повысить охват про-

Таблица 1

Матрица парных сравнений

Направление	Текущее	К лидеру	Случайное
Текущее	1	$\frac{K}{K+9k},\ k < K$ $0{,}1,\ k \ge K$	$\frac{K}{K+4k},\ k < K$ $0{,}2,\ k \ge K$
К лидеру	$1+\frac{9k}{K},\ k < K$ $10,\ k \ge K$	1	$1+\frac{4k}{K},\ k < K$ $5,\ k \ge K$
Случайное	$1+\frac{4k}{K},\ k < K$ $5,\ k \ge K$	$\frac{K}{K+4k},\ k < K$ $0{,}2,\ k \ge K$	1

странства поиска, предотвращая преждевременную сходимость алгоритма к локальному экстремуму. Эта процедура применяется в двух случаях. Во-первых, когда количество итераций, в течение которых не происходило улучшение значения целевой функции особи, превышает пороговое значение τ_0. Во-вторых, когда количество итераций алгоритма кратно заданному числу k_τ. При этом выполняется перенос случайно выбранных m^k особей. Полученные при этом особи вносятся в популяцию и для них вычисляется значение целевой функции. Затем m^k особей с худшим значением целевой функции удаляется. Величина m^k может уменьшаться по закону вида $m^k = m_0 \cdot g(k)$, где, например, $m_0 = 0{,}1 \cdot S$, а $g(k)$ – монотонно убывающая функция, принимающая значения из промежутка $[0, 1]$.

При реализации оператора переноса для каждой координаты особи $\mathbf{x}_j^k$ генерируется случайное число $\alpha_i \in [-0{,}5,\ 0{,}5]$ и эта особь заменяется новой особью с координатами:

$$x_{ij}^{k+1} = \begin{cases} x_{ij}^k + \alpha_i \cdot \left(x_i^{\max} - x_{ij}^k\right) \cdot g(k), & \alpha_i > 0, \\ x_{ij}^k + \alpha_i \cdot \left(x_{ij}^k - x_i^{\min}\right) \cdot g(k), & \alpha_i < 0. \end{cases} \quad (10)$$

Если $g(k) \in [0, 1]$, то $x_{ij}^{k+1} \in \left(x_i^{\min},\ x_i^{\max}\right)$.

Оператор переноса может быть построен несколько иным образом. Пусть

$$z_{ij}^{\max} = \min\left(x_{ij}^k + 0{,}5 \cdot \left(x_i^{\max} - x_i^{\min}\right) \cdot g(k),\ x_i^{\max}\right),$$

$$z_{ij}^{\min} = \max\left(x_{ij}^k - 0{,}5 \cdot \left(x_i^{\max} - x_i^{\min}\right) \cdot g(k),\ x_i^{\min}\right).$$

Тогда

$$x_{ij}^{k+1} = z_{ij}^{\min} + \alpha_i \cdot \left(z_i^{\max} - z_i^{\min}\right),$$

где α_i – случайная величина, равномерно распределенная на промежутке $[0,\ 1]$.

Оператор переноса должен обладать следующими свойствами. В начале вычислительного процесса (при малых значениях k) новая особь $\mathbf{x}_j^{k+1}$ может располагаться в достаточно широкой области $A \subset D$, чем обеспечивается диверсификация поиска. По мере увеличения номера итерации k эта область должна сужаться (существенно, начиная, например, с $k = \frac{K}{2}$), а при $k \to \infty$ $\mathbf{x}_j^{k+1} \to \mathbf{x}_j^k$. Это способствует интенсификации поиска, т.е. более полному исследованию окрестности каждой особи.

Данный эффект достигается, если в качестве функции $g(k)$ использовать рациональную сигмоиду вида:

$$g(q) = \frac{1}{2} \cdot \left(1 - \frac{q}{\mu + |q|}\right),\ q = k - \frac{K}{2}.$$

или, если раскрыть знак модуля:

$$g(k) = \begin{cases} \dfrac{\mu - 2k + K}{2\mu - 2k + K}, & k < \dfrac{K}{2}, \\ \dfrac{\mu}{2\mu + 2k - K}, & k \geq \dfrac{K}{2}. \end{cases}$$

Параметр μ определяет степень кривизны сигмоиды. Хорошие показатели сходимости получены при $\mu = 100$.

Разработанный алгоритм выглядит следующим образом:

1. По формуле (1) создать начальную популяцию. Для каждой особи определить значение целевой функции и положить количество итераций τ_j, в течение которых не происходило улучшение значения целевой функции, равным нулю. Найти лидера. Положить $k = 1$.

2. Определить элементы матрицы A (табл. 1) и вектор приоритетов (6) для всех частиц. Положить $j = 1$.

3. Для j-й частицы выполнить пробный шаг (7). Если значение целевой функции не улучшилось, то осуществить следующий шаг (8). Если значение целевой функции вновь не улучшилось, то сделать шаг (9). Если на некотором шаге целевая функция улучшается, то переместить особь в этом направлении. В противном случае особь не перемещается.

4. Если при выполнении пункта 3 j-я особь не переместилась, то увеличить τ_j на единицу, иначе положить его равным нулю.

5. Если $\tau_j = \tau_0$, то положить τ_j равным нулю и выполнить оператор переноса (10).

6. Если $\tau_j < \tau_0$ и $j < S$, то увеличить индекс j на единицу и перейти к пункту 3.

7. Если $j = S$ и k кратно k_τ, то применить оператор переноса к случайно выбранным m^k особям. Полученных особей внести в популяцию и вычислить для них значение целевой функции. Удалить m^k особей с худшим значением целевой функции.

8. Найти лидера и проверить выполнение критерия останова (5). В случае выполнения завершить поиск, иначе уменьшить длину шага по правилу $\lambda^{k+1} = \delta\lambda^k,\ \delta \in (0, 1)$. Положить $k = k+1$ и перейти к шагу 2 (следующей итерации алгоритма).

Схема разработанного алгоритма напоминает поведение пресноводных гидр [18]. Значение целевой функции можно интерпретировать как благоприятность условий среды. Гидра перемещается в направлении улучшения условий среды (соотношения (7)). Если в результате очередного «шага» оказывается, что эти условия ухудшились, то гидра выбирает новое направление движения (формулы (8)–(9)). Если гидра не может найти перспективное направление движения (пункт 5 алгоритма) или наступают неблагоприятные условия (пункт 7), то приступает к размножению половым путем, выпуская в воду яйцо и при этом погибая. Яйцо переносится водой (оператор переноса (10)). Из яйца развивается новая особь гидры.

На основе этой интерпретации авторы предлагают назвать разработанный метод алгоритмом гидр или H-алгоритмом (от английского hydra – «гидра»).

Для повышения скорости сходимости выполняется гибридизация рассмотренных алгоритмов. Она может осуществляться различными способами [11]:

I. Гибридизация вложением.

1. Высокоуровневая гибридизация вложением, при которой после каждой итерации популяционного алгоритма выполняется итерация алгоритма локального поиска (например, метода сопряженных градиентов) для особи с наилучшим значением целевой функции. Эта процедура называется обучением лидера.

2. Низкоуровневая гибридизация вложением характеризуется высокой степенью интеграции комбинируемых алгоритмов, так что декомпозиция полученного алгоритма на отдельные алгоритмы затруднительна. При этом можно, например, получать новые особи в пункте 7 на основе генетических операторов (скрещивания, мутации, инверсии и т. д.) или операторов других алгоритмов.

II. Последовательная гибридизация, когда на первом этапе поиска используется алгоритм, осуществляющий широкий обзор исследуемой области (например, генетический алгоритм), а на втором этапе применяется алгоритм, уточняющий найденные решения (H-алгоритм). В данной статье запуск второго алгоритма осуществлялся при стагнации вычислительного процесса для первого алгоритма.

III. Коалгоритмическая гибридизация, когда происходит параллельное решение задачи разными алгоритмами и обмен найденными решениями.

Процедура обучения лидера

С целью повышения скорости сходимости всех рассматриваемых алгоритмов используются классические методы оптимизации для уточнения лучшего решения – обучение лидера. В качестве метода обучения лидера применялся метод сопряженных градиентов [10].

Сначала задается начальное направление минимизации целевой функции $f(\mathbf{x})$: $\mathbf{d}^k = -\nabla f(\mathbf{x}^k),\ k = 0$. В процессе одной итерации метода $n+1$ раз выполняется следующее:

1. Вычисляется шаг $\delta^k = \arg\min_{\delta}\left(\mathbf{x}^k + \delta\mathbf{d}^k\right)$ с помощью какого-либо метода одномерной оптимизации (например, метода «золотого сечения»).

2. Определяется новое приближение $\mathbf{x}^{k+1} = \mathbf{x}^k + \delta^k \cdot \mathbf{d}^k$.

3. Находится коэффициент $w^k = \dfrac{\nabla^T f\left(\mathbf{x}^{k+1}\right)\cdot\nabla f\left(\mathbf{x}^{k+1}\right)}{\nabla^T f\left(\mathbf{x}^k\right)\cdot\nabla f\left(\mathbf{x}^k\right)}$ и направление минимизации $\mathbf{d}^{k+1} = -\nabla f\left(\mathbf{x}^{k+1}\right) + w^k \cdot \mathbf{d}^k$. Сопряженность направлений $\mathbf{d}^{k+1}$ и $\mathbf{d}^k$ означает выполнение равенства $\left(\mathbf{d}^{k+1}\right)^T \cdot H \cdot \mathbf{d}^k = 0$, где $H = \nabla^2 f(\mathbf{x})$ – матрица вторых частных производных.

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ И ИХ ОБСУЖДЕНИЕ

Сравнение скорости сходимости рассматриваемых методов выполнялось на примере следующих тестовых функций [11]:

1) $F_1(\mathbf{x}) = \sum_{i=1}^{n-1}\left(100\cdot\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i\right)^2\right)$ – функция Розенброка. Оптимальное решение $x_i^{opt}=1$, $i=\overline{1,\,n}$, $F_1\left(\mathbf{x}^{opt}\right)=0$.

2) $F_2(\mathbf{x}) = \sum_{i=1}^{n-1}\left(x_{i+1}^2+x_i^2\right)^{0,25}\times$

$\times\left(\sin^2\left(50\left(x_{i+1}^2+x_i^2\right)^{0,1}\right)+1\right)$ – функция Дэвиса. Оптимальное решение $x_i^{opt}=0$, $i=\overline{1,\,n}$, $F_2\left(\mathbf{x}^{opt}\right)=0$.

3) $F_3(\mathbf{x}) = -20\exp\left(-0,2\cdot\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-$

$-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+\exp(1)+20$ – функция Экли. Оптимальное решение $x_i^{opt}=0$, $i=\overline{1,\,n}$, $F_3\left(\mathbf{x}^{opt}\right)=0$.

4) $F_4(\mathbf{x}) = \sum_{i=1}^{n}10\left(1-\cos\left(2\pi x_i\right)\right)+x_i^2$ – функция Растригина. Оптимальное решение $x_i^{opt}=0$, $i=\overline{1,\,n}$, $F_4\left(\mathbf{x}^{opt}\right)=0$.

Во всех случаях предполагалось, что $x_i\in[-10,\,10]$, $i=\overline{1,\,n}$.

Графики скорости сходимости для данных функций (средние значения целевой функции лидера за 50 запусков алгоритма) приведены на рис. 1, 2. Рассматривалось два случая: без обучения лидера и с обучением (т. е. высокоуровневая гибридизация вложением) методом сопряженных градиентов. Во всех случаях размерность пространства поиска составляла $n=10$, популяция состояла из 100 особей.

Рассматривались следующие алгоритмы оптимизации:

1) PSO – алгоритм роя частиц;

2) RGA – генетический алгоритм с вещественным кодированием;

3) H – алгоритм гидр;

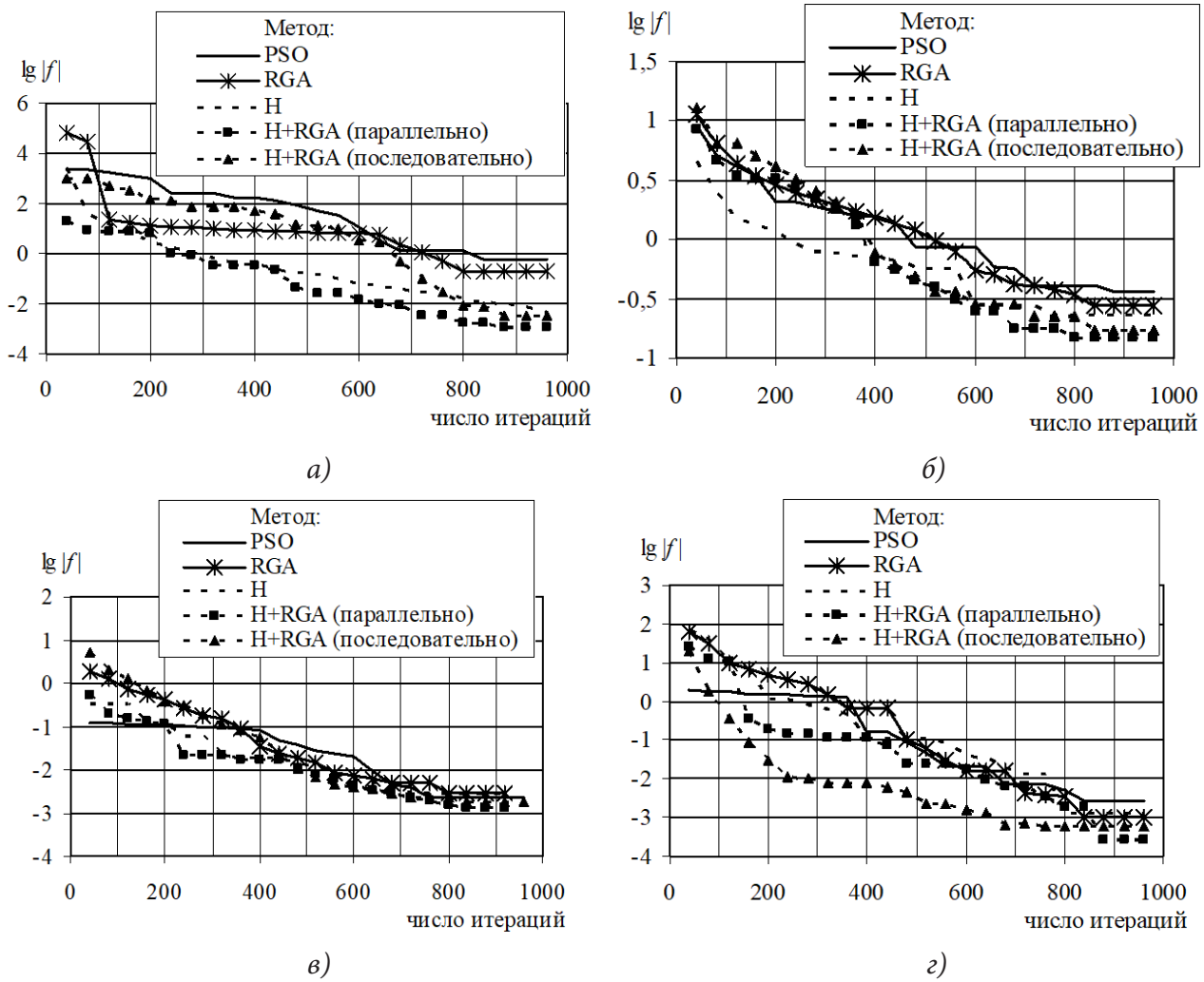

Рис. 1. Графики скорости сходимости для методов без обучения лидера а) F_1; б) F_2; в) F_3; г) F_4

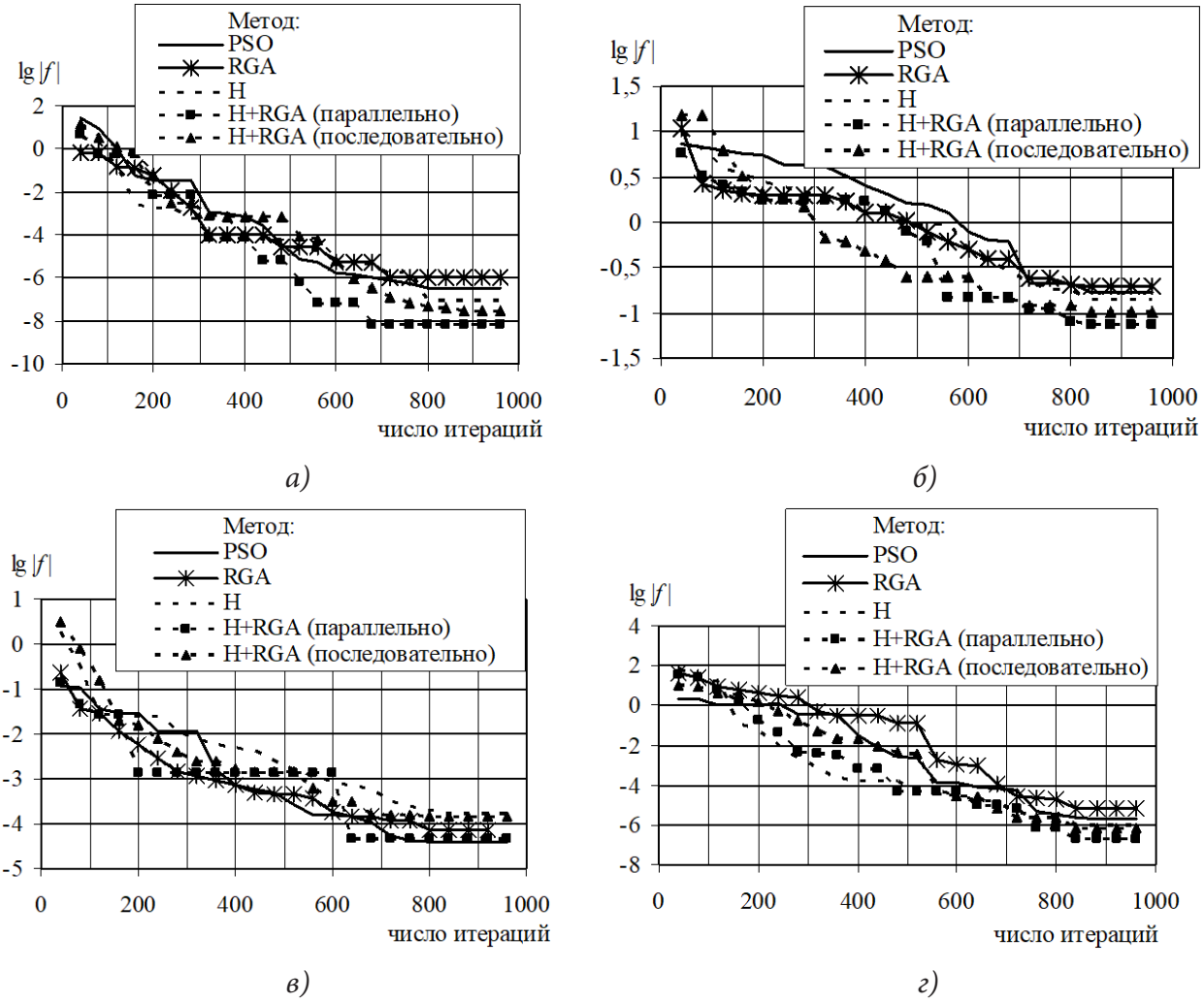

Рис. 2. Графики скорости сходимости для методов с обучением лидера а) F_1; б) F_2; в) F_3; г) F_4

4) H+RGA (параллельно) – коалгоритмическая гибридизация H-алгоритма и алгоритма RGA;

5) H+RGA (последовательно) – последовательная гибридизация: в начале поиск осуществляется алгоритмом RGA, а в конце H-алгоритмом.

Результаты сравнения скорости сходимости алгоритмов в порядке убывания скорости сходимости (по количеству итераций) приведены в табл. 2.

Для всех рассматриваемых тестовых функций лучшие результаты показала коалгоритмическая гибридизация алгоритма RGA и H-алгоритма, несколько худшие – их последовательная гибридизация. Алгоритмы PSO, RGA и H-алгоритм для различных тестовых функций показали различную, но в среднем одинаковую скорость сходимости. Обучение лидера существенно повысило скорость сходимости всех алгоритмов. При этом все рассматриваемые алгоритмы в среднем показали одинаковую скорость сходимости.

ЗАКЛЮЧЕНИЕ

В работе рассмотрены популяционные алгоритмы решения задачи оптимизации, такие как алгоритм роя частиц и генетический алгоритм. Представлен разработанный на основе алгоритма роя частиц и метода анализа иерархий алгоритм пресноводных гидр (H-алгоритм) и предложена его биологическая интерпретация. Описаны подходы для гибридизации данного алгоритма с другими (высоко- и низкоуровневая гибридизация вложением, последовательная и коалгоритмическая гибридизация).

Таблица 2

Сравнение скорости сходимости алгоритмов

Функция	Расположение алгоритмов оптимизации в порядке убывания скорости сходимости				
	1	2	3	4	5
Без обучения лидера					
F_1	H+RGA (параллельно)	H+RGA (последовательно)	H	RGA	PSO
F_2	H+RGA (параллельно)	H+RGA (последовательно)	H	RGA	PSO
F_3	H+RGA (параллельно)	H+RGA (последовательно)	PSO	RGA	H
F_4	H+RGA (параллельно)	H+RGA (последовательно)	RGA	PSO	H
С обучением лидера					
F_1	H+RGA (параллельно)	H+RGA (последовательно)	H	PSO	RGA
F_2	H+RGA (параллельно)	H+RGA (последовательно)	H	PSO	RGA
F_3	H+RGA (параллельно)	H+RGA (последовательно)	PSO	RGA	H
F_4	H+RGA (параллельно)	H+RGA (последовательно)	H	PSO	RGA

Проведено тестирование скорости сходимости рассмотренных алгоритмов на примере различных многоэкстремальных функций. Показано, что скорости сходимости алгоритмов PSO, RGA и H-алгоритма приблизительно сопоставимы и отличаются для разных тестовых функций. Скорость сходимости этих методов ниже по сравнению с методами, основанными на последовательной и параллельной гибридизации RGA и H-алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. *Mahmood, S. A.* Neural Network with PSO based training to detect spoofing websites / S. A. Mahmood, N. G. M. Jameel, S. J. Sayda // Journal of Theoretical and Applied Information Technology. – 2018. – Vol. 96(12). – P. 3612–3622.

2. *Syulistyo, A. R.* Particle swarm optimization (PSO) for training optimization on convolutional neural network (CNN) / A. R. Syulistyo, D. M. J. Purnomo, M. F. Rachmadi, A. Wibowo // Journal of Computer Science and Information. – 2016. – Vol. 9(1). – P. 52–58. – DOI: http://dx.doi.org/10.21609/jiki.v9i1.366.

3. *Ibrahim, A. M.* Particle Swarm Optimization trained recurrent neural network for voltage instability prediction / A. M. Ibrahim, N. H. El-Amary // Journal of Electrical Systems and Information Technology. – 2018. – Vol. 5(2). – P. 216–228. – DOI: 10.1016/j.jesit.2017.05.001.

4. *Королев, С. А.* Метод идентификации параметров модели метаногенеза в виде системы дифференциальных уравнений на основе генетического алгоритма / С. А. Королев, Д. В. Майков // Интеллектуальные системы в производстве. – 2012. – №1. – С. 29–35.

5. *Кетова, К. В.* Идентификация и прогнозирование обобщающих показателей развития региональной экономической системы / К. В. Кетова, И. Г. Русяк // Прикладная эконометрика. – 2009. – №3 (15). – С. 56–71.

6. *Королев, С. А.* Оптимизация двухстадийного режима метаногенеза на основе принципа максимума Понтрягина / С. А. Королев, Д. В. Майков // Вестник Донского государственного технического университета. – 2019. – Т. 19, №2. – С. 195–203.

7. *Nezhadhosein, S.* Integrating differential evolution algorithm with modified hybrid GA for solving nonlinear optimal control problems / S. Nezhadhosein, A. Heydari, R. Ghanbari // Iranian Journal of Mathematical Sciences and Informatics. – 2017. – Vol. 12, No. 1. – P. 47–67. – DOI: 10.7508/ijmsi.2017.01.005.

8. *Salehpour, E.* Solving optimal control problems by PSO-SVM / E. Salehpour, J. Vahidi, H. Hosseinzadeh // Computational Methods for Differential Equations. – 2018. – Vol. 6(3). – P. 312–325.

9. *Сайранов, А. С.* Применение генетических алгоритмов для управления организационными системами при возникновении нештатных ситуаций / А. С. Сайранов, Е. В. Касаткина, Д. Г. Нефедов, И. Г. Русяк // Компьютерные исследования и моделирование. – 2019. – Т. 11, № 3. – С. 533–556.

10. *Yao, S. W.* An adaptive three-term conjugate gradient method based on self-scaling memoryless BFGS matrix / S. W. Yao, L. S. Ning // Journale Computational and Applied Mathematics. – 2018. – Vol. 332. – P. 72–85.

11. *Карпенко, А. П.* Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учебное пособие / А. П. Карпенко. – 2-е изд. – Москва: Изд-во МГТУ имени М. Э. Баумана, 2017. – 446 с.

12. *Kennedy, J.* Particle swarm optimization / J. Kennedy, R. C. Eberhart // Proceedings of IEEE International Conference on Neural Networks. – 1995. – P. 1942–1948.

13. *Hussain, A.* Optimization through Genetic Algorithm with a New and Efficient Crossover Operator / A. Hussain, Y. S. Muhammad, A. Nawaz // International Journal of Advances in Mathematics. – 2018. – Vol. 2018(1). – P. 1–14.

14. *Li, X.* An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem / X. Li, L. Gao // International Journal of Production Economics. – 2016. – Vol. 174. – P. 93–110.

15. *Kundakci, N.* Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem / N. Kundakci, O. Kulak // Computers & Industrial Engineering. – 2016. – Vol. 96. – P. 31–51.

16. *Sano, M.* Balancing setup workers load of flexible job shop scheduling using hybrid genetic algorithm with tabu search strategy / M. Sano, M. Nagao, Y. Morinaga // International Journal of Decision Support Systems. – 2016. – Vol. 2(1). – P. 71–90.

17. *Саати, Т.* Принятие решений. Метод анализа иерархий / Т. Саати. – Москва: Радио и связь, 1993. – 278 с.

18. *Дауда, Т. А.* Зоология беспозвоночных: учебное пособие / Т. А. Дауда, А. Г. Кощаев. – Москва: Лань, 2014. – 208 с.

Королев Станислав Анатольевич – канд. физ.-мат. наук, доцент кафедры «Математическое обеспечение информационных систем» ФГБОУ ВО Ижевский государственный технический университет имени М. Т. Калашникова.
E-mail: stkj@mail.ru

Майков Дмитрий Владимирович – преподаватель математики БПОУ УР Ижевский торгово-экономический техникум.
E-mail: MaykovD@yandex.ru

MODIFICATION OF PARTICLE SWARM ALGORITHM BASED ON HIERARCHY ANALYSIS METHOD

S. A. Korolev*, D. V. Maykov**

**Kalashnikov Izhevsk State Technical University*
***Izhevsk Trade and Economic College*

Annotation. There are problems of mathematical modeling, the solution of which necessitates finding the maximum or minimum point of some function of many variables. For example, this is training artificial neural networks, identifying the parameters of mathematical models from experimental data, and finding the optimal control for these models. To quickly find the extremum point, a lot of population optimization algorithms have been developed (particle swarm optimization (PSO), genetic algorithm and others). A modification of the particle swarm optimization algorithm based on the hierarchy analysis method is proposed. The scheme of the developed algorithm is inspired by the behavior of freshwater hydras; therefore, the authors propose to call it the H-algorithm. Successive and parallel (co-algorithmic) hybridization of the developed algorithm with the genetic algorithm using real coding (RGA) is performed. For the considered algorithms, a high-level hybridization by embedding is implemented, in which a local search is carried out in the vicinity of an agent with the best value of the objective function using the conjugate gradient method (leader training). Comparison of the effectiveness of the considered methods was performed on the example of various multi-extreme test functions (the function of Rosenbrock, Davis, Ackley, Rastrigin). PSO, RGA and H-algorithm for different test functions showed different, but on average the same convergence rate. Leader training significantly increased the convergence rate of all algorithms. The best results in convergence rate were shown by parallel and serial hybridization of the RGA and the H-algorithm.

Keywords: optimization problem, particle swarm optimization algorithm, genetic algorithm, analytic hierarchy process, leader training, conjugate gradient method, embedment hybridization, sequential hybridization, coalgorithmic hybridization.

Korolev Stanislav Anatolevich – Candidate of Physics and Mathematics, Associate Professor, Department of Software for Information Systems, Kalashnikov Izhevsk State Technical.
E-mail: stkj@mail.ru

Maikov Dmitriy Vladimirovich – mathematics teacher, Izhevsk Trade and Economic College.
E-mail: MaykovD@yandex.ru