ОБЗОР МЕМЕТИЧЕСКИХ И МУЛЬТИ-МЕМЕТИЧЕСКИХ МЕТОДОВ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

В.А. Маслий, *магистрант*
Российский технологический университет МИРЭА
(Россия, г. Москва)

DOI:10.24412/2500-1000-2024-1-3-102-111

Аннотация. *Задача глобальной оптимизации играет решающую роль во многих областях. Для её решения используются меметические алгоритмы. В статье представлен обзор современных модификаций, позволяющих сократить время поиска и повысить качество решений. Рассмотренные модификации расширяют исследуемую область и применяют различные методы локального поиска.*

Ключевые слова: *меметический алгоритм, глобальный поиск, локальный поиск, кроссовер, мутация, оптимизация.*

Методы меметической оптимизации имеют структуру, в которой при помощи вероятностных операторов селекции, кроссинговера, мутации, методов локального поиска, а также механизмов обновления популяции развивается популяция пробных решений.

Вероятностный оператор селекции особей в пространстве популяций имеет аналогичное значение естественному отбору в природе. Данный оператор осуществляет выбор номера родительской особи, которая будет использоваться для создания следующего поколения потомков.

Общая схема алгоритма:

Меметический алгоритм.

Шаг 1. Генерация начальной популяции P_0, $t := 0$.

Шаг 2. Пока не выполнен критерий остановки, выполнять шаги 2.1–2.4

2.1. При помощи операторов селекции, кроссинговера, мутации построить популяцию потомков Q_t.

2.2. Выполнить локальное улучшение популяции Q_t.

2.3. Из популяций Q_t и P_t создать популяцию P_{t+1} следующего поколения, $t := t + 1$.

2.4. При условии выполнения критерия перезапуска выполнить перезапуск с новой начальной популяцией P_0, $t := 0$.

Начальную популяцию можно построить, сгенерировав особей случайно или с помощью конструктивных эвристик и/или эвристик локального поиска.

Для критерия остановки может использоваться конкретно заданное число итераций или количество вычислений целевой функции, достижение максимального числа итераций без улучшения рекорда, реализация некоторого числа перезапусков алгоритма, достижение заданного значения целевой функции и др.

Для построения потомков осуществляется отбор родительских особей при помощи функции приспособленности особей (равна целевой или ее обобщение/модификация для направленного поиска) или на основе разнообразия популяции. Потом с помощью операторов скрещивания и мутации комбинируются родительские особи. Оператор выбирается на этапе создания алгоритма или в процессе его выполнения с помощью механизмов обучения, свойств родителей, степени разнообразия популяции и др. Значения настраиваемых параметров меметического алгоритма могут быть адаптированы в процессе работы алгоритма.

В меметическом алгоритме локальная оптимизация играет значительную роль, поскольку ее целью является улучшение качества потомков. В данном случае потомок подвергается усовершенствованию, что аналогично обучению в жизненных условиях. Меметический алгоритм объединяет две методологии поиска: популяционные алгоритмы и методы локального

поиска. При этом он может рассматриваться как подход, в котором локальный поиск воздействует на множество решений, периодически применяя операции рекомбинации для сотрудничества и обмена информацией между решениями.

Современные меметические алгоритмы содержат набор методов локального поиска, которые определяют, какие операторы следует использовать для каждого решения, с какой силой и с какой частотой. Для этого используются адаптивные гиперэвристики, механизмы машинного обучения, самоадаптирующиеся и коэволюционные методы, схемы, основанные на разнообразии популяции и т. д. Как правило, эвристики локального поиска включают случайный локальный поиск и различные вариации алгоритмов локальной оптимизации с ограничениями на количество итераций и долю просматриваемых решений в окрестности. На начальном этапе и в процессе эволюции популяции используются случайный локальный поиск и локальный спуск в окрестности текущего решения, в то время как на завершающем этапе активно применяется интенсивный локальный поиск, при котором рассматривается большая часть или всё решение в окрестности на каждой итерации работы алгоритма.

При отборе особей для следующего поколения учитывается их качество, разнообразие и возраст. Важно поддерживать разнообразие в популяции, чтобы избежать достижения локального оптимума и продолжить исследование новых областей поиска. Существуют две основные схемы воспроизведения: популяционная, где число потомков на каждой итерации близко или равно числу особей предыдущего поколения, и стратегия с частичным воспроизведением, где замещается только часть популяции. Известны меметические алгоритмы, где популяция хранится и обновляется с использованием определенной структуры, такой как тернарное дерево или кольцо.

Если популяция начинает вырождаться, что проявляется, например, в снижении разнообразия до определенного порога или длительном отсутствии улучшений, рекомендуется перезапустить процесс. В этом случае создается новая начальная популяция, которая иногда включает некоторые особи из предыдущей популяции, возможно, подвергнутые изменениям с использованием эвристик.

Также при реализации меметического алгоритма высока роль способа представления решений и типы окрестностей, которые используются для локального поиска. Они определяют эффективность, трудоёмкость и перспективность алгоритма.

Основная часть

Существует большое количество модификаций классических алгоритмов. В статье [1] авторы предложили свою модификацию метода дифференциальной эволюции. Метод относится к эволюционным алгоритмам. И используется для решения задач оптимизации большой размерности (однокритериальных или многокритериальных). Основных параметров алгоритма дифференциальной эволюции всего три: фактор масштабирования мутации, размер популяции и crossover rate. Этот алгоритм создает популяцию из векторов $x_{i,G}, i = 1, 2, \dots, N$, где $N -$ размер популяции, которые изначально находятся в случайных местах. Особи нового поколения создаются путем комбинации элементов с предыдущего поколения. На каждом поколении G создается мутантный вектор $v_{i,G}$, который создается для каждого целевого вектора $x_{i,G}, i = 1, 2, \dots, N$ в текущем поколении. Так наиболее часто используемые операторы мутации представлены ниже:

Модификация случайного вектора:

$$v_{i,G} = x_{r_0,G} + F\left(x_{r_1,G} - x_{r_2,G}\right)$$

Модификация текущего вектора с помощью лучшего:

$$v_{i,G} = x_{i,G} + F\left(x_{best,G} - x_{i,G}\right) + F\left(x_{r_0,G} - x_{r_1,G}\right)$$

Модификация лучшего вектора 1:

$$v_{i,G} = x_{best,G} + F\left(x_{r_0,G} - x_{r_1,G}\right)$$

Модификация лучшего вектора 2:

$$v_{i,G} = x_{best,G} + F\left(x_{r_0,G} - x_{r_1,G}\right) + F\left(x_{r_2,G} - x_{r_3,G}\right)$$

где $F \in \mathbb{R}-$ фактор мутации, генерируемый в интервале $[0,1]$, $x_{best,G}-$ лучший вектор-индивид в популяции на текущем шаге, $r_0, \dots, r_3-$ случайно выбранные не повторяющиеся индексы и отличные от $i, x_{best,G}$.

Оператор кроссовера создает новый пробный вектор $u_{i,G} = \left(u_{i,1,G}, u_{i,2,G}, \dots, u_{i,D,G}\right)$ на с помощью смешивания элементов из векторов $v_{i,G}$ и $x_{i,G}$. Операцию выбора элемента можно записать следующим образом:

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, \text{если } R_j(0,1) \leq C_r \text{ или } j = j_{rand} \\ x_{i,j,G}, \text{если } R_j(0,1) > C_r \end{cases},$$

где $j = 1, 2, \dots, D$, $j_{rand}-$ случайное целое число из интервала $[1, D]$. $R_j(0,1)-$ случайное число, генерируемое с равномерным распределением для каждого индекса j, в промежутке от 0 до 1, $C_r \in [0,1]$ – оценочный параметр смешивания.

После создания нового пробного вектора реализуется процесс выбора, направленный на выбор лучшего вектора $x_{i,G}$ или $u_{i,G}$. Для задачи минимизации выбор вектора для следующего поколения $x_{i,G+1}$ показан в выражении:

$$x_{i,G+1} = \begin{cases} u_{i,G}, \text{если } f\left(u_{i,G}\right) < f\left(x_{i,G}\right) \\ x_{i,G}, \text{иначе} \end{cases},$$

где $f(.)-$ минимизируемая функция. То есть если новый вектор показывает лучший результат, чем исходный то исходный вектор заменяется новым, в противном случае вектор остается без изменений. В классическом алгоритме дифференциального поиска оптимизируется каждое решение в популяции и на каждой последующей итерации алгоритма остаются только лучшие решения с предыдущего шага.

В модификации этого алгоритма, названной μDSDE, используется только шесть векторов кандидатов, один из них сохраняет наилучшее найденное решение, второй находится в окрестности первого, а оставшиеся четыре случайно ищут решение в других областях, что помогает не выбирать локальный минимум преждевременно и осуществлять поиск глобального оптимума. В качестве оператора локального поиска используется оператор направленного локального поиска, который в случае улучшения решения шагает в сторону локального минимума со скоростью:

$$velocity = F\left(x_{r_0,G} - x_{r_1,G}\right),$$

где $x_{r_0,G}$ – вектор лучшего решения, $x_{r_1,G}$ – второй лучший вектор соответственно. Если же следующая итерация направленного локального поиска не дает результатов, то он прекращается.

Вторая модификация стандартного метода Shuffled frog leaping algorithm (алгоритм перемешанных лягушек), который является эвристикой, имитирующей поведение лягушек. По принципу работы алгоритм похож на алгоритм роя частиц. В алгоритме различные особи популяции (лягушки) распределяются на группы, которые формируются из отсортированного в порядке убывания качества решения, то есть в каждой последующей группе все решения хуже, чем худшее решение из предыдущей группы. В каждой группе лягушки разделены в соответствии со значением пригодности от лучшей к худшей. Оператор перемешивания, который разде-

ляет особи на группы, выражен следующим образом:

$$P = \{Q_k | Q_k = Q_{i+m(j-1)}, i = 1, 2, \dots, m, j = 1, 2, \dots, n, k = 1,2, \dots, popsize\},$$

где $m -$ количество субпопуляций, $n -$ количество лягушек в каждой субпопуляции, $popsize = m \times n -$ количество лягушек в популяции.

После процесса перемешивания худшая лягушка может быть обновлена с помощью информации от лучшей лягушки (оператор прыжка лягушки), что описывается выражением:

$$Q' = Q_w + rand * (Q_{best} - Q_w),$$

где Q_w и $Q_{best} -$ представляют позиции локально худшей и лучшей лягушек соответственно, $rand -$ случайное число с равномерным распределением в интервале от 0 до 1. В случае же невозможности обновления худшей особи лучшим значением, значение лучшей локальной особи Q_{best} может быть заменено глобально лучшей особью Q_g. В случае же если на значение не может быть обновлено генерируется новое решение, случайно выбранное из всего пространства, где оптимизируемая функция определена.

Так как оператор перемешивания решений не является оператором глобального поиска в полной мере, так как все изменения основаны на лучших решениях в каждой группе, то данный не модифицированный алгоритм больше подходит для локального поиска минимума.

Для адаптации SFLA к глобальному поиску минимума и решения задач оптимизации авторы [2] модифицировали алгоритм с помощью процесса меметической диффузии, меметической эволюции и меметического обучения особей.

Процесс меметической диффузии может быть описан оператором перемешивания особей, но в модификации особи распределяются по группам в другом порядке. Сначала все особи сортируются по убыванию качества решения, после чего в каждую группу добавляется по одной особи из начала списка. То есть сначала в каждую группу добавляется по одной особи с наилучшим решением, а затем группы пополняются остальными решениями.

Вторая модификация реализована при помощи меметической эволюции. Для того, чтобы худшая лягушка при прыжке ориентировалась не только на лучшую лягушку, добавляется еще две точки, обозначающие геометрический и гравитационный центры. Для определения позиции геометрического центра Q_c используется формула:

$$Q_c = \frac{1}{n}\sum_{j=1}^{n} Q_j.$$

Для определения позиции гравитационного центра сначала вычисляется сила притяжения между элементами:

$$F_{ij} = G_0 \frac{M_i M_j}{R_{ij}^2 + \varepsilon},$$
$$R_{ij} = \left\| x_i - x_j \right\|_2,$$

где M_i и $M_j -$ представляют массу частиц i и j, x_i и x_j представляют положения частиц в пространстве высокой размерности, $R_{ij} -$ евклидова норма между x_i и x_j, $\varepsilon -$ небольшая константа. Качество решения частицы используется для измерения гравитационной массы частицы:

$$M_i = \frac{m_i}{\sum_{i=1}^{n} m_i},$$
$$m_i = \frac{fit_i - worstfit}{bestfit - worstfit},$$

где fit_i – представляет качество решения i-ой частицы, $worstfit$ и $bestfit$ – представляют качество решения худшей и лучшей частиц в популяции из n частиц. Отношение величин силы гравитации между двумя частицами:

$$Gf_{ij} = \frac{\frac{M_j}{R_{ij}^2 + \varepsilon}}{\sum_{j=1}^{n} \frac{M_j}{R_{ij}^2 + \varepsilon}} (j \neq i),$$

откуда вытекает формула для определения гравитационного центра

$$Q_g = \sum_{j=1, j\neq i}^{n} Gf_{ij} \cdot Q_j,$$

где Q_j – позиция j-ой лягушки, n – количество лягушек в субпопуляции, Q_g – гравитационный центр. Процесс выбора точки притяжения для худшей лягушки можно записать следующим образом:

$$Q_m = \begin{cases} Q_g, \text{если } rand < 0.5 \\ Q_c, \text{иначе} \end{cases}.$$

Тогда новое положение худшей частицы будет определяться с помощью выражения:

$$Q'_w = Q_w + rand(Q_{best} - Q_w) + rand(Q_m - Q_w),$$

где Q_w – худшая лягушка, Q_{best} – лучшая лягушка. Таким образом, худшая лягушка может избежать попадания в локальный оптимум, но так как это не гарантированно, то применяется полет Леви, который действует, как оператор мутации для двух центров притяжения: гравитационного и геометрического и определяется выражением:

$$LevyFlight = rand(0,1) \cdot normal(0,1) \cdot Levy,$$

где $rand(0,1)$ – число, полученное с помощью равномерного распределения, $normal(0,1)$ – число, полученное с помощью нормального распределения, $Levy$ – число, полученное с помощью распределения Леви в пространстве высокой размерности. Определение нового положения худшей лягушки с применением полета Леви выглядит следующим образом:

$$Q'_w = Q_w + rand(Q_{best} - Q_w) + rand(LevyFlight \otimes Q_m - Q_w),$$

где $\otimes$ – поэлементное умножение.

Последняя модификация – это процесс меметического обучения особей основан-

ный на том, что особи должны учиться у других случайных особей, а также необходимости более широкого изучения возможных решений, чтобы не попасть в локальный минимум.

Для этого авторы решили использовать аппроксимацию дискретного равномерного распределения для выбора особей и измерения.

$$u_i = randperm(D),$$
$$popA = randperm(popSize),$$
$$popB = randperm(popSize),$$

где u_i – вектор строка с различными случайными целыми числами в диапазоне [1, D]. $popA$ и $popB$ – векторы строки со случайными целыми числами в диапазоне [1, popSize] соответственно.

Выбор измерения, основанный на аппроксимации равномерного дискретного распределения, можно записать как:

$$U_i = u_j(1{:}\, ceil(rand * D))$$

где $rand * D$ – равномерное случайное распределение в диапазоне [0, D], $ceil$ – оператор округления в сторону большего целого. В итоге операцию обновления особи можно записать в виде:

$$Q'_{ij} = \begin{cases} Q_{ij} + rand\left(Q_{popA(u),j} - Q_{popB(v),j}\right), j \in U_i \\ Q_{ij}, j \notin U_i \end{cases},$$

где $Q_{popA(u)}$ и $Q_{popB(v)}$ – осуществляют выбор случайной особи на основе аппроксимации равномерного дискретного распределения, j – целое число в диапазоне [1, D], u, v, i – целые числа в диапазоне [1, popSize] соответственно.

В источнике [3] авторы описывают их меметический алгоритм для решения задач крупномасштабной глобальной оптимизации. Основные идеи – добавление методов многородительской эволюции и пошагового адаптивного локального поиска. Мутантный вектор генерируется по формуле:

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + \boldsymbol{r}_1\left(\boldsymbol{x}_{best,G} - \boldsymbol{x}_{i,G}\right) + \boldsymbol{r}_2\left(\boldsymbol{x}_{p-best,G} - \boldsymbol{x}_{,G}\right),$$

где $\boldsymbol{x}_{best,G}$ – лучший вектор в текущем поколении, $p - best -$ индекс случайно выбранного вектора из 10% лучших векторов G-го поколения. Затем в зависимости от значений параметров в операции кроссовера выбирается один из векторов: мутантный вектор, этот же вектор с предыдущего шага, либо один из двух лучших векторов популяции. Операцию кроссовера можно записать в виде:

$$x_{ji,G+1} = \begin{cases} v_{ji,G}, \text{если } r_{rand(ji)} \le CP1, \\ x_{ja(i),G}, \text{если } CP1 < r_{rand(ji)} \le CP1 + CP2, \\ x_{jb(i),G}, \text{если } CP1 + CP2 < r_{rand(ji)} \le CP1 + CP2 + CP3, \\ x_{ji,G}, \text{иначе}, \end{cases}$$

где $j \in \{1,2,\dots,D\}$, $a(i)$ и $b(i) \in \{1,2,\dots,NP\} -$ индексы векторов случайно выбранных из лучшей половины векторов G-го поколения. Размер популяции уменьшается в процессе оптимизации, пока не уменьшится до одной особи.

Пошаговый адаптивный алгоритм локального поиска имеет базовый размер ша-

га для каждого измерения. Затем выбираются случайные измерения, размер шага в которых умножается на случайное число. Выбор измерений, в которых ведется поиск, производится в соответствии с выражением:

$$f_{ji,G} = \begin{cases} 1, \text{если } rand(1) \le \dfrac{0.5}{D} + \left(\dfrac{0.1 * D}{iteration}\right)^2 \text{ или } j = j_{rand(i)}, \\ 0, \quad \text{иначе,} \end{cases}$$

где $f_{ji} \in \{0,1\}$ – показывает будет ли j-е измерение i-го вектора изменяться, $rand(1)$ – функция, генерирующая случайное число с равномерным распределением в интервале $(0,1)$, $iteration$ – номер итерации, $j_{rand(i)} \in \{1,2,\dots,D\}$ – случайно выбранный индекс, чтобы убедится, что хотя бы одно измерение в векторе $\boldsymbol{x}_i$ будет участвовать в поиске. Генерируется новое решение, в соответствии с формулой:

$$\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G} + \boldsymbol{s}_{i,G} * rand(D,1) * \boldsymbol{f}_{i,G},$$

где $\boldsymbol{s}_{i,G}$ – вектор, представляющий стандартный размер шага i-го индивида в G-ом поколении. Если новое решение лучше исходного, размер шага этих выбранных измерений умножается на 2. В противном случае решение восстанавливается, и каждый размер шага умножается на -0,5. Таким образом шаг обновляется в соответствие с выражением:

$$s_{ji,G+1} = \begin{cases} s_{ji,G} \times (-0.5), \text{если } \boldsymbol{x}_{i,G} \text{ лучше чем } \boldsymbol{x}_{i,G+1} \text{ и } f_{ji,G} = 1, \\ s_{ji,G} \times 2, \quad \text{если } \boldsymbol{x}_{i,G} \text{ не лучше чем } \boldsymbol{x}_{i,G+1} \text{ и } f_{ji,G} = 1, \\ s_{ji,G}, \text{иначе.} \end{cases}$$

В модифицированном алгоритме реализуется выполнение операции глобального поиска после нескольких операций выполнения локального поиска. После уменьшения количества особей до заданного значения производится только локальный поиск.

В источнике [4] авторы представили меметический алгоритм, основанный на использовании изменение методов квантового моделирования и соединении их с меметическими алгоритмами для улучшения компоненты локального поиска. Алгоритм квантового моделирования основан на волновой функции, описывающей квантовое состояние частиц, и методе Монте-Карло. В итоге позиция частицы при симуляции поведения квантовой частицы в одномерном пространстве определяется выражением:

$$x = p \pm g|x - p| \ln\left(\frac{1}{rand}\right),$$

где g – параметр поиска, p – потенциальная яма.

Также используется алгоритм гравитационного поиска, основанный на законе тяготения Ньютона. Формула для определения силы притяжения между элементами:

$$F_{ij} = G_0 \frac{M_i M_j}{R_{ij}^2 + \varepsilon},$$
$$R_{ij} = \left\| x_i - x_j \right\|_2,$$

где M_i и $M_j -$ представляют массу частиц i и j, x_i и x_j представляют положения частиц в пространстве высокой размерности, $R_{ij} -$ евклидова норма между x_i и x_j, $\varepsilon -$ небольшая константа. Качество решения частицы используется для измерения гравитационной массы частицы:

$$M_i = \frac{m_i}{\sum_{i=1}^{n} m_i},$$
$$m_i = \frac{fit_i - worstfit}{bestfit - worstfit},$$

где $fit_i -$ представляет качество решения i-ой частицы, $worstfit$ и $bestfit -$ представляют качество решения худшей и лучшей частиц в популяции из n частиц. Отношение величин силы гравитации между двумя частицами. Правило поиска гравитационного алгоритма может быть сформулировано следующим образом:

$$v_i^d = rand \cdot v_i^d + \frac{\sum_{j \in Kbest, j \neq i} G_0 \frac{M_i M_j}{R_{ij}^2 + \varepsilon}\left(x_j^d - x_i^d\right)}{M_i},$$
$$x_i^d = x_i^d + v_i^d,$$

где $Kbest -$ набор K лучших частиц, $d -$ измерение, $v_i^d -$ ускорение.

В алгоритме используется разбиение индивидуумов в популяции на локальные группы, которые периодически перетасовываются и обмениваются информацией. В данном методе для автоматического балансирования между глобальным и локальным поисками используется каскадная параллельная меметическая структура, в которой один уровень структуры зависит от другого. Зависимый уровень структуры активируется группами индивидуумов с независимого уровня структуры.

Механизм квантовой эволюции реализован в каждой отдельной группе. Для повышения разнообразия населения используются разные центры потенциала:

$$P = (1 - w) \cdot x_{worst} + w \cdot x_{best}, \qquad w = rand,$$
$$sgn() = \begin{cases} 1, \text{если } rand < 0.5 \\ -1, \text{иначе} \end{cases},$$
$$g = 1.5 - 0.5 \times \frac{FES}{MAX\_FES},$$
$$x_{worst} = P \pm g \cdot sgn() \cdot |x_{worst} - P'| \cdot \frac{1}{\ln(rand)},$$

где $x_{worst} -$ представляет позицию худшей частицы, $x_{best} -$ позиция лучшей частицы, $P -$ центр потенциала, который находится в прямоугольной области, где x_{worst} и x_{best} вершины диагонали в этой области. В итоге худшая частица должна сходиться к одному из центров P или P' $P \neq P'$:

$$x_{worst}(t) = P(t), t \to \infty,$$
$$x_{worst}(t) = P'(t), t \to \infty.$$

Также для улучшения разнообразия индивидуумов применяется геометрический и гравитационный центры, позволяющие избежать попадания в локальный минимум. Положение геометрического центра определяется как:

$$P_c = \frac{1}{n}\sum_{j=1}^{n} x_j,$$

где n – количество частиц в подгруппе. Отношение величин силы гравитации между двумя частицами:

$$Gf_{ij} = \frac{\frac{M_j}{R_{ij}^2 + \varepsilon}}{\sum_{j=1}^{n} \frac{M_j}{R_{ij}^2 + \varepsilon}} (j \neq i),$$

$$P_g = \sum_{j=1, j\neq i}^{n} Gf_{ij} \cdot x_j,$$

где P_j – позиция j-ой частицы, n – количество частиц в подгруппе, P_g – гравитационный центр. Затем один из центров - геометрический или гравитационный выбираются в качестве альтернативного центра квантового потенциала:

$$P' = \begin{cases} P_g, \text{если } rand < 0.5 \\ P_c, \text{иначе} \end{cases}.$$

Итоговую формулу для вычисления нового положения худшей частицы можно записать в виде:

$$x_{worst} = x_{worst} + rand(x_{best} - x_{worst}) + parameter \cdot Direction \cdot Length,$$

где $parameter = g \cdot \frac{1}{\ln(rand)}, Length = |P' - x_{worst}|, Direction = sgn()$.

Благодаря данным изменениям авторам оригинальной статьи удалось повысить скорость работы алгоритма, однако для лучших результатов требуется правильное задание параметров алгоритма.

Заключение

В заключении можно сделать вывод, что меметические алгоритмы активно применяются в реальных задачах, характеризующихся мультимодальностью, большими размерностями решений, а также накладываемыми ограничениями на время, необходимое для решения задачи, и вычислительными затратами. А для более точного результата следует расширять границы исследуемой области, чтобы повысить вероятность нахождения глобального оптимума, при этом применять локальный поиск для уточнения решения, найденного во время разведки. Следовательно, необходимо соблюдать баланс между локальным и глобальным поиском.

Библиографический список

1. Йылдыз Ю.Э., Топал А.О. Крупномасштабная непрерывная глобальная оптимизация на основе микродифференциальной эволюции с локальным направленным поиском // Информационные науки. – 2019. – № 477. – С. 533-544.

2. Тан Д., Чжэнь Л., Ян Дж., Чжао Дж. Алгоритм меметического прыжка лягушки для глобальной оптимизации // Мягкие вычисления. – 2019. –№23. – DOI: 10.1007/s00500-018-3662-3.

3. Моура О., Пауло З., Венфен Л. Новый меметический алгоритм, основанный на многопараметрической эволюции и адаптивном локальном поиске для крупномасштабной глобальной оптимизации // Вычислительный интеллект и нейронаука. – 2022. – С. 1687-5265.

4. Тан Д., Лю З., Чжао Дж., Донг С., Цай Ю. Меметический алгоритм квантовой эволюции для глобальной оптимизации // Нейронные вычисления и приложения. – 2019. – № 32. – С. 9299-9329.

5. Туан Н.К., Хоанг Т.Д., Бинь Х.Т. Управляемая дифференциальная эволюционная многозадачность с использованием метода поиска Пауэлла для решения многоцелевой непрерывной оптимизации // Конгресс IEEE по эволюционным вычислениям (CEC). 2018. doi: 10.1109/CEC.2018.8477860.

6. Сун Дж., Мяо З., Гонг Д., Цзэн Х.-Дж., Ли Дж., Ван Г. Интервальная многоцелевая оптимизация с помощью меметических алгоритмов // IEEE Операции в Кибернетике. 2020. DOI: 10.1109/TCYB.2019.2908485

7. Еремеев А.В. Эволюционные алгоритмы. – 2022. – DOI:10.48550/arXiv.1511.06987.

HUMAN DEVELOPMENT AND LABOR POTENTIAL ANALYSIS IN THE KYRGYZ REPUBLIC

S.U. Astanova
Osh Technological University named after Academician M.M. Adyshev
(Kyrgyzstan, Osh)

Abstract. *The article addresses the problem of human development and analysis of labor potential in the Kyrgyz Republic. The author points out the low level of education and the shortage of highly qualified specialists. The potential of the workforce in Kyrgyzstan is surveyed, with the identification of negative and positive trends in the labor market. The article concludes with findings within the scope of the research.*

Keywords: *human development, labor potential analysis, education, highly qualified specialists, innovation.*