УДК 681.3.681.5

Обзорная статья

https://doi.org/10.32603/2071-8985-2026-19-3-21-42

Алгоритмы решения задачи покрытия множества

П. И. Васькин[1], **А. А. Лисс**[2]✉

[1] ЗАО «Морские компьютерные системы», Санкт-Петербург, Россия

[2] Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина), Санкт-Петербург, Россия

✉ aaliss@etu.ru

Аннотация. Приведен обзор алгоритмов решения задачи покрытия множества. Рассматриваются две альтернативные формулировки задачи покрытия множества, используемые авторами алгоритмов. Дается классификация алгоритмов решения задачи покрытия множества. Рассматриваются особенности наиболее известных алгоритмов решения задачи покрытия множества. Основное внимание уделено алгоритмам, использующим метаэвристики. Среди метаэвристик различают основанные на популяции и на единичных данных. К метаэвристикам, основанным на популяции, относятся: основанные на рое; на физике; на основе эволюции; на основе человека. Метаэвристики на основе единичных данных: имитация отжига; поиск табу; управляемый локальный поиск; жадная рандомизированная процедура адаптивного поиска.

Ключевые слова: задача покрытия множества, таблица покрытия, правила преобразования таблиц покрытия, методы решения таблиц покрытия, метаэвристические методы

Для цитирования: Васькин П. И., Лисс А. А. Алгоритмы решения задачи покрытия множества // Изв. СПбГЭТУ «ЛЭТИ». 2026. Т. 19, № 3. С. 21–42. doi: 10.32603/2071-8985-2026-19-3-21-42.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Review article

Algorithms for Solving the Set Coverage Problem

P. I. Vaskin[1], **A. A. Liss**[2]✉

[1] Marine Computer Systems CJSC, Saint Petersburg, Russia

[2] Saint Petersburg Electrotechnical University, Saint Petersburg, Russia

✉ aaliss@etu.ru

Abstract. Provides an overview of algorithms for solving the set coverage problem. Two alternative formulations of the set coverage problem, which are used by the authors of the algorithms, are considered. A classification of algorithms for solving the set coverage problem is given. The features of the most well-known algorithms for solving the set coverage problem are considered. The main focus is on algorithms using metaheuristics. Among the metaheuristics, there are population-based metaheuristics and single-data-based metaheuristics. Population-based metaheuristics include: swarm-based; based on physics; based on evolution; based on humans. Metaheuristics based on single data: simulated annealing; Taboo search; guided local search; greedy randomized adaptive search procedure.

Keywords: the set coverage problem, the coverage table, the rules for converting coverage tables, methods for solving coverage tables, metaheuristic methods

For citation: Vaskin P. I., Liss A. A. Algorithms for Solving the Set Coverage Problem // LETI Transactions on Electrical Engineering & Computer Science. 2026. Vol. 19, no. 3. P. 21–42. doi: 10.32603/2071-8985-2026-19-3-21-42.

Conflict of interest. The authors declare no conflicts of interest.

Введение. Задача покрытия множества Set Cover Problem (SCP) – это классическая задача комбинаторного анализа. Она относится к одной из 21 NP-полных (non-deterministic) вычислительных задач, перечисленных Р. М. Карпом в [1]. Точные методы решения в основном представлены алгоритмами полного перебора и методами ветвей и границ. Эти алгоритмы требуют очень большого времени для получения результата и могут решать только экземпляры ограниченного размера [2]. Поэтому в большинстве публикаций усилия авторов сосредоточены на получении приближенного решения за разумное время. Для уменьшения перебора возможных вариантов решения используются различного рода эвристики.

В последнее время наиболее часто для решения задачи покрытия множества предлагаются метаэвристические методы. Их суть заключается в том, что решение задачи получается в результате выполнения довольно сложного алгоритма, обычно моделирующего поведение популяции животных (птиц, насекомых) или физический процесс. Как правило, метаэвристический метод требует задания значений параметров, конкретизирующих алгоритм. Фактически метаэвристический метод определяет семейство алгоритмов, каждому из которых соответствует набор значений параметров метаэвристики.

Задачи оптимизации встречаются в огромном количестве областей – от маршрутизации в сети Интернет до планирования производственных очередей и оптимизации запаса товаров на складе. В различных видах деятельности проводятся попытки достичь определенных целей или оптимизировать что-либо: качество, время, расстояние и т. п. В реальных задачах ресурсы ограничены, поэтому всегда будет актуальным нахождение решения для оптимального использования тех или иных ценных ресурсов при различных ограничениях.

Классификация алгоритмов решения задачи покрытия множества может быть проведена по многим признакам. Если взглянуть на природу алгоритмов, то их можно поделить на две категории: детерминированные и стохастические алгоритмы. Детерминированные алгоритмы следуют строгой процедуре. Такие алгоритмы достигают одного и того же решения, если начинают с одной начальной точки. Если в алгоритме есть некоторая случайность, то такой алгоритм – стохастический. Случайность приводит к тому, что алгоритм обычно достигает иного решения каждый раз при запуске, даже если начинает в одной и той же точке.

Стохастические алгоритмы можно разделить на два типа: эвристические и метаэвристические. Однако разницу между ними сложно сформулировать, так как в литературе нет согласованных определений эвристики и метаэвристики, в некоторых источниках оба термина используют взаимозаменяемо. Под эвристическим подходом часто понимают то, что для поставленной задачи оптимизации не гарантировано нахождение оптимального решения, но за разумное время будет найдено достаточно близкое к нему [3]. Дальнейшим развитием эвристических алгоритмов принято считать метаэвристические. Полагается, что они работают лучше, чем простые эвристики. Для метаэвристических алгоритмов в некоторых источниках выделяют следующую характерную черту: использование компромисса между рандомизацией и локальным поиском [4].

В последних публикациях существует тенденция называть все стохастические алгоритмы с рандомизацией и глобальным поиском метаэвристическими. Рандомизация обеспечивает хороший способ перемещения от локального поиска к глобальному. Большинство метаэвристик предназначены для глобальной оптимизации.

Основные компоненты любого метаэвристического алгоритма – это интенсификация и диверсификация, или эксплуатация и исследование. Диверсификация означает генерирование разнообразных решений, чтобы исследовать пространство поиска в глобальном масштабе. Интенсификация же означает сосредоточение внимания на локальном поиске, используя информацию о том, что найдено хорошее решение в этой области. Выбор лучших решений позволяет сходиться к оптимальному значению, а диверсификация с помощью рандомизации позволяет выходить из локальных минимумов и увеличить разнообразие решений.

Агентом называют кандидата, используемого в качестве решения задачи. В зависимости от источника или алгоритма агенты могут называться по-разному: частицы, муравьи, светлячки, пчелы и т. п. В более общем случае агентов в источниках называют индивидуумами или особями [5].

Метаэвристические алгоритмы можно разделить на два вида [6]: траекторные и популяционные. В траекторных алгоритмах проводится обновление на каждой итерации положения только одного агента. При этом общее количество индивидуумов может быть больше единицы, и на разных итерациях разные агенты могут перемещаться. В популяционных алгоритмах число агентов больше единицы. Кроме того, на каждой итерации перемещаются либо все индивидуумы, либо определенное количество, превышающее единицу. В разных источниках популяционные алгоритмы могут называть поведенческими, роевыми и т. д.

Целью данной стать служит систематический обзор и анализ современных алгоритмов решения задачи покрытия множества. Далее приводятся понятия, связанные с таблицами покрытия. Предлагается перед применением алгоритма решения задачи покрытия множества приводить таблицу покрытия к циклическому виду. Отмечается, что в процессе работы приближенных алгоритмов могут появляться недопустимые или избыточные решения. Приводится псевдокод оператора осуществимости, делающий решение допустимым и безызбыточным. Рассматривается ряд приближенных алгоритмов решения задачи покрытия множества. В заключении представлены выводы и обозначены направления для будущих исследований.

Постановка задачи. Задача покрытия множества (Set Covering Problem, SCP) – это однокритериальная многомерная задача оптимизации, которая формулируется следующим образом: дано конечное множество $X = \{e_1, e_2, ..., e_k\}$ и набор подмножеств $F = \{S_1, S_2, ..., S_n\}$, где $S_i \subseteq X$, $i = 1, ..., n$. Каждому подмножеству S_i соответствует некоторая стоимость. Цель заключается в нахождении набора подмножеств F^*, который покрывает все множество X (т. е. $\bigcup_{S_j \in F^*} S_j = X$).

В случае взвешенной задачи покрытия множества F^* должен иметь минимально возможный вес, а в случае невзвешенной – минимальное число подмножеств.

Рассмотрим пример: пусть $X = \{1,2,3,4,5,6,7,8,9,10\}$ и дано семейство множеств $F = \{S_1 = \{1,3,5,8,9\}, S_2 = \{2,3,6,10\}, S_3 = \{4,7,9\}, S_4 = \{1,2,8,10\}, S_5 = \{2,3,4\}, S_6 = \{1,3,6,9\}, S_7 = \{7,10\}\}$. Минимальное покрытие F^* будет состоять из множеств S_1, S_2 и S_3.

Часто SCP представляют в виде бинарной матрицы покрытия A размером $m \times n$, где $a_{ij} = 1$, если элемент i принадлежит множеству S_j, и $a_{ij} = 0$ в противном случае. Обычно ее называют таблицей покрытия. Для примера:

$$A = \begin{Bmatrix} 1010100110 \\ 0110010001 \\ 0001001010 \\ 1100000101 \\ 0101100000 \\ 1010010010 \\ 0000001001 \end{Bmatrix}.$$

При этом возможны две метафоры такого представления, а именно: «строки покрывают столбцы» и «столбцы покрывают строки».

Рассмотрим метафору «строки покрывают столбцы». Обозначим через $A = (a_{ij})$ – произвольную матрицу размера $m \times n$ с элементами $a_{ij} \in \{0,1\}$ без нулевых строк и столбцов. Пусть каждому столбцу поставлено в соответствие положительное число c_j, называемое весом столбца. Требуется найти покрытие минимального суммарного веса (взвешенная задача покрытия) или покрытие минимальной мощности (невзвешенная задача покрытия, при которой все $c_j = 1$). Введя переменные x_j, равные 1, если столбец j входит в искомое покрытие, и равные 0 в противном случае, приходим к следующей формулировке задачи о покрытии: минимизировать сумму $\sum_{j=1}^{n} c_j x_j$ при ограничениях

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1,$$

где $i = 1, ..., m;\ x_j \in \{0,1\};\ j = 1, ..., n.$

При использовании метафоры «строки покрывают столбцы» каждой строке поставлено в соответствие положительное число c_i, называе-

мое весом строки. Переменная x_i равна 1, если строка входит в покрытие, в противном случае она равна 0. Сумма, которую требуется минимизировать, будет выглядеть следующим образом: $\sum_{i=1}^{m} c_i x_i$. Ограничения в таком случае имеют вид

$$\sum_{i=1}^{m} a_{ij} x_i \geq 1,$$

где $j = 1, ..., n;\ x_i \in \{0,1\};\ i = 1, ..., m.$

Основные понятия. Рассмотрим ряд понятий, связанных с таблицами покрытия, используя метафору «строки покрывают столбцы».

Решение таблицы покрытия (ТП) – это множество строк, в котором есть отметки в каждом столбце. Решение, из которого нельзя удалить ни одну строку, не нарушив условия покрытия всех столбцов, называется безызбыточным. Для невзвешенной задачи покрытия интересны безызбыточные решения с минимальным числом строк.

Существуют правила преобразования ТП, выполнение которых не приводит к потере минимального решения. Для невзвешенной задачи покрытия эти правила формулируются следующим образом. Первое правило использует понятия существенного столбца и существенной строки. Столбец называется существенным, если он содержит отметку только в одной строке ТП. Строка называется существенной, если она содержит отметку в существенном столбце.

Правило 1. Существенная строка выбирается в решение ТП. После этого исходная ТП преобразуется следующим образом: исключается существенная строка и все столбцы, в которых она имеет отметки.

Правило 2. Строка исключается из ТП, если она полностью совпадает с другой строкой или множество ее отметок является подмножеством отметок другой строки ТП.

Правило 3. Столбец исключается из ТП, если он полностью совпадает с другим столбцом или имеется столбец ТП, множество отметок которого является подмножеством исключаемого столбца.

Для взвешенной задачи покрытия правило 1 верно, а правила 2 и 3 должны быть дополнены учетом весов.

Таблица покрытия называется циклической, если к ней не применимо ни одно из точных правил преобразования ТП. Практическая рекомендация при нахождении решения ТП следующая: прежде чем применять эвристический (метаэвристический) алгоритм, целесообразно упростить ТП, используя правила 1–3.

Каждая оптимизационная задача имеет свои ограничения и свойства. На этапе генерации решений или в процессе работы метаэвристики, могут появляться недопустимые решения. В рамках задачи покрытия недопустимыми считаются решения, которые не покрывают все столбцы. Для исправления подобных решений используется оператор осуществимости [7], [8], который дополнительно устраняет избыточные строки. Пусть: I – множество всех столбцов; J – множество всех строк; α_i – множество всех строк, которые покрывают столбец $i \in I$; β_j – множество всех столбцов, которые покрывает строка $j \in J$; S – множество всех строк решения; U – множество непокрытых столбцов; ω_i – число строк, которые покрывают столбец $i \in I$. Тогда псевдокод оператора осуществимости выглядит следующим образом:

$init\ \omega_i := | S \cap \alpha_i |, \forall i \in I,$
$init\ U := \{ i \mid \omega_i = 0, \forall i \in I \}.$
for each $column\ i \in U$ (в возрастающем порядке) do
    find row $j \in \alpha_i$ with $\min\left(c_j / \left|U \cap \beta_j\right|\right)$
    add j to S
    set $\omega_i \leftarrow \omega_i + 1, \forall i \in \beta_j$
    set $U \leftarrow U - \beta_j$
end for
for each row $j \in S$ (в убывающем порядке) do
    if $\omega_i \geq 2, \forall i \in \beta_j$ then
        set $S \leftarrow S - j$
        set $\omega_i \leftarrow \omega_i - 1, i \in \beta_j$
    end if
end for

Обзор алгоритмов решения. Самый простой точный метод нахождения решения ТП – алгоритм полного перебора (brute-force). Любая задача из класса NP может быть решена с его помощью [2]. Из-за высокой сложности алгоритма полного перебора точные алгоритмы в основном построены на методе ветвей и границ [9] и методе ветвей и сечений, которые представляют собой более эффективную модификацию алгоритма brute-force. Алгоритм основан на полном переборе вариантов, который оптимизируется за счет отсечения ветвей поиска, заведомо не содержащих оптимального решения. Отсечение происхо-

дит на основе вычисления нижних и верхних границ целевой функции. Однако все равно данные точные алгоритмы требуют большого количества времени и могут решать экземпляры задач ограниченного размера [2], [9]. Классические жадные алгоритмы имеют высокую скорость [9] и просты в реализации, однако из-за своей детерминированной природы редко дают качественные решения, особенно на больших экземплярах [9], [10].

Поэтому большое количество работ посвящено исследованию и созданию эффективных метаэвристических алгоритмов для решения больших экземпляров задач покрытия за ограниченное время. Широко представлены для решения SCP генетические алгоритмы [11], [12]. Кроме того, в последние годы большое количество исследований направлено на решение данной проблемы с помощью биоинспирированных алгоритмов [13], которые вдохновляются происходящими в природе процессами:

– алгоритм пчелиной колонии [8], [14];
– алгоритм оптимизации стаи кошек [15];
– алгоритм поиска кукушки [8];
– алгоритм муравьиной колонии [16], [17];
– алгоритм светлячков [18], [19].

Природа послужила основой для алгоритма черной дыры [20], [21].

Очень подробная классификация метаэвристических алгоритмов и обширная библиография приведена в [22]. Несмотря на то, что в ней рассматриваются методы оптимизации без привязки к задаче покрытия множества, классификация может быть полезной в исследовании алгоритмов решения SCP.

Рассмотрим характерные черты метаэвристик, которые могут использоваться для решения задачи покрытия множества.

Метаэвристики, основанные на популяции.

Метаэвристики, основанные на рое. *Оптимизация с использованием роя частиц (PSO).* Оптимизация методом роя частиц, предложенная в [23], [24] – это стохастический алгоритм на основе роя, инспирированный природой и первоначально разработанный в 1995 г. Р. Эберхартом и Дж. Кеннеди, опираясь на упрощенную модель поведения птиц в стае и рыб в косяке. В PSO группа частиц (например, стая птиц) выполняет поиск в пространстве решений. Каждая частица определяет свой следующий шаг в этом пространстве, основываясь на пройденном пути всех частиц. Как только частицы обновят свои позиции на текущей итерации, начинается следующая итерация процесса поиска. В конце концов весь рой приближается к оптимуму целевой функции со скоростью сходимости, которая сильно зависит от выбранного варианта PSO и значений параметров. В [25] проведен краткий обзор существующих исследований по этому направлению за последние годы.

Как и в других метаэвристических алгоритмах, в PSO существует несколько основных этапов, включая инициализацию роя, вычисление фитнесс-функции для каждой частицы, стратегию обновления и оценку приспособленности. Последние два шага повторяются в последующих итерациях до тех пор, пока не будет удовлетворено условие завершения (например, достижение заданного числа итераций).

Часто при инициализации роя частицы разбрасываются случайным образом в пространстве возможных решений без определенного критерия. Следует отметить, что различные стратегии инициализации могут в некоторых случаях значительно улучшить скорость сходимости. Частицы роя проверяются в конце каждой итерации на индивидуальное и глобальное лучшие положения. Предполагается, что каждая частица имеет уникальное значение пригодности на каждой итерации, которое вычисляется посредством оценки целевой функции. PSO запоминает индивидуальное лучшее решение (кандидат на лучшее глобальное решение), которое когда-либо встречалось каждой частице до текущей итерации.

Оптимизация с помощью муравьиной колонии. Имитация поведения муравьев, колония которых находит кратчайший путь к пище, используя механизм феромонов [16], [17]. В контексте задачи покрытия множества «путь» муравья – это подобранное им подмножество множеств, образующее покрытие.

Алгоритм начинается с создания графа, где вершины – это множество подмножеств $F = \{S_1, S_2, ..., S_n\}$. Каждому ребру S_j (выбору подмножества) присваивается начальное (малое) количество феромона τ_j. Определяется эвристическая информация η_j («жадность») для каждого подмножества S_j. Чаще всего $\eta_j = |S_j| / c_j$ – чем больше элементов покрывает подмножество и чем оно дешевле, тем оно привлекательнее.

На каждой итерации t работает несколько «муравьев» (обычно от 10 до 50). Муравей начинает с пустого решения $F^* = \varnothing$. Все элементы универсума считаются непокрытыми. На каждом шаге муравей выбирает следующее множество S_j для добавления в F^* с вероятностью

$$p_j(t) = \tau_j(t)^{\alpha}\,\eta_j^{\beta} \Big/ \left(\sum_{l \in \text{доп}} \tau_l(t)^{\alpha}\,\eta_l^{\beta} \right),$$

где $\tau_j(t)$ – количество феромона на множестве S_j на итерации t; η_j – привлекательность подмножества S_j; α и β – параметры, определяющие влияние феромона и эвристики; допустимые подмножества – те, которые покрывают хотя бы один еще не покрытый элемент, l – порядковый номер допустимого подмножества.

Локальное обновление феромонов опционально. Сразу после добавления подмножества S_j в решение количество феромона на нем можно немного уменьшить, например: $\tau_j = (1-\phi)\tau_j$, где ϕ – коэффициент локального испарения $(0 < \phi < 1)$. Это поощряет разнообразие и предотвращает следование по одному пути.

Выбор подмножества и локальное обновление повторяются, пока муравей не покроет все элементы универсума X.

После того как все муравьи построили свои решения, происходит глобальное обновление феромонов. Сначала феромон на всех подмножествах «испаряется» (чтобы избежать застоя): $\tau_j(t+1) = (1-\rho)\tau_j(t), \forall j \in [1,n]$, где ρ – коэффициент глобального испарения ($0 < \rho < 1$). Затем только муравьи, нашедшие лучшие решения, добавляют феромон на подмножества, входящие в их решение: $\tau_j(t+1) = \tau_j(t) + \Delta\tau k_j$, где k_j – количество появлений подмножества S_j в решениях «хороших» муравьев; $\Delta\tau$ – константа. Если $\Delta\tau$ – не константа, а функция от сложности решения (чем лучше решение, тем больше феромона она оставляет), то формула вычисления нового значения феромона должна быть преобразована.

Алгоритм повторяет основной цикл, пока не будет выполнен один из критериев:

– найдено решение удовлетворительного качества;

– превышено максимальное число итераций;

– решение перестало улучшаться в течение многих итераций.

Алгоритм искусственной пчелиной колонии. Алгоритм имитирует поведение пчелиной колонии при сборе нектара [8], [14]. Его работа строится на нескольких ключевых принципах:

– каждое возможное решение задачи (кандидат на роль покрывающего множества) рассматривается как «источник пищи». Качество этого решения определяется «количеством нектара»;

– в алгоритме задействованы три типа программных «пчел», что обеспечивает баланс между интенсивным исследованием перспективных участков и поиском новых решений;

– рабочие пчелы отвечают за локальный поиск в окрестностях уже известных хороших решений, пытаясь их улучшить;

– пчелы-наблюдатели выбирают один из источников пищи, опираясь на информацию от рабочих пчел, и также занимаются его локальным улучшением. Вероятность выбора источника пропорциональна его качеству – работает принцип «чем лучше решение, тем больше ему внимания»;

– пчелы-разведчики занимаются случайным поиском в пространстве решений, чтобы находить новые, потенциально перспективные области и не допускать «застревания» алгоритма в локальном оптимуме.

Случайным образом создается начальная популяция решений (источников пищи). Каждая рабочая пчела проводит локальный поиск вокруг своего текущего решения, создавая и оценивая его модифицированную версию. На основе полученной информации о качестве всех решений пчелы-наблюдатели выбирают одно из них (с вероятностью, зависящей от его качества) и также пытаются его улучшить с помощью локального поиска. Работа пчел продолжается до тех пор, пока не будет достигнуто максимальное число итераций или не будет найдено решение удовлетворительного качества.

Алгоритм поиска кукушки. Алгоритм имитирует поведение некоторых видов кукушек, подбрасывающих свои яйца в гнезда других птиц [8].

Каждое «гнездо» в алгоритме представляет собой одно возможное решение задачи покрытия множества. «Подбросить яйцо» – значит создать новое, немного измененное решение на основе существующего. Специальный механизм случайного поиска (полеты Леви) позволяет алгоритму эффективно исследовать все пространство решений, сочетая небольшие локальные изменения и редкие, но большие «прыжки». Каждая кукушка подбрасывает одно яйцо (новое решение) в слу-

чайно выбранное гнездо. Хозяева охраняют свои гнезда. Гнезда с худшими решениями (яйцами низкого качества) могут быть обнаружены и уничтожены с некоторой вероятностью. Лучшие решения (гнезда с качественными яйцами) сохраняются для следующего поколения.

Полет Леви – это особый тип случайного блуждания (в нем используется распределение Леви), при котором короткие шаги часто прерываются редкими, но очень большими «прыжками». В отличие от обычного случайного блуждания, где длина шага предсказуема, в полете Леви длина каждого следующего шага определяется по степенному закону, что делает появление большого скачка хоть и маловероятным, но всегда возможным.

Алгоритм начинается с инициализации популяции. Случайным образом создается n гнезд (начальных решений). Каждое решение представляется в виде двоичного вектора, где 1 означает, что множество включено в покрытие, а 0 – не включено. Для каждого гнезда i в популяции создается новое решение-кандидат: Новое_решение_i_1 = Текущее_решение_i_1 + $\alpha_1 \times$ Леви_полет(), где $\alpha_1 > 0$ – шаг масштабирования. Поскольку задача дискретная результат операции округляется до 0 или 1. Вычисляется функция пригодности (Fitness) для нового решения. Для задачи покрытия это может быть: Fitness = Число_покрытых_элементов – ω × Стоимость_выбранных_множеств, где ω – штрафной коэффициент за стоимость. Новое решение заменяет текущее, только если его пригодность лучше. После обработки всех гнезд наименее качественные решения отбрасываются с определенной вероятностью (обычно 0.25). Вместо них создаются новые совершенно случайные решения. Это помогает избежать застревания в локальных оптимумах. На каждой итерации запоминается лучшее найденное решение. Алгоритм заканчивает свою работу, если достигнет максимального числа итераций или найдет решение удовлетворительного качества.

Поскольку цель – найти покрытие минимальной стоимости, то фитнес-функция должна поощрять покрытие всех элементов. Поэтому в нее часто добавляют штраф за непокрытие всех элементов:

($-\beta_1$ × количество_непокрытых_элементов),

где β_1 – большой штрафной коэффициент.

Алгоритм светлячка (*FA*). Еще одна из распространенных метаэвристик – это FA.

Существует около двух тысяч видов светлячков, многие из которых могут производить короткие и чередующиеся вспышки. Характер вспышек часто уникален для конкретного вида. Мигающий свет создается с помощью процесса биолюминесценции. Истинное назначение данных сигнальных возможностей все еще обсуждается, однако на данный момент считается, что две основные функции таких вспышек – привлечение особей противоположного пола и привлечение потенциальной добычи. Кроме того, мигание может также служить защитным механизмом от хищников, предупреждая о горьком вкусе. Некоторые виды тропических светлячков могут согласовывать свои вспышки, что служит примером природного самоорганизованного поведения.

Известно, что интенсивность на определенном расстоянии от источника подчиняется закону обратных квадратов, т. е. интенсивность света Light уменьшается по мере увеличения расстояния r по правилу $\text{Light} \sim 1/r^2$. Вдобавок среда распространения поглощает свет, который становится слабее по мере увеличения дистанции. Поэтому светлячки видимы на ограниченном расстоянии, достаточном для общения.

Мигающий свет можно сформулировать таким образом, чтобы он был связан с целевой функцией, которая подлежит оптимизации. Существует два варианта метаэвристических алгоритмов, вдохновленных поведением светлячков, – алгоритм светлячков и алгоритм оптимизации роем светлячков. Рассмотрим первый вариант алгоритма светлячков.

FA впервые представлен в 2009 г. Синь-Ше Яном (Xin-She Yang). Данный алгоритм идеализирует некоторые характеристики мигания светлячков, поэтому модель имеет следующие правила поведения [3]:

– все светлячки однополые, поэтому они могут привлекать друг друга независимо от пола;

– привлекательность пропорциональна яркости, поэтому для любых двух светлячков менее яркий будет двигаться к более яркому;

– привлекательность уменьшается с увеличением расстояния, аналогично яркости;

– если нет более яркого светлячка, чем конкретный, он двигается случайным образом;

– яркость светлячка определяется целевой функцией.

Для задачи максимизации яркость может быть просто пропорциональной значению целевой функции, а для задачи минимизации – обратно пропорциональной.

Привлекательность светлячков определяется с помощью следующего соотношения [19]: $\beta_2 = \beta_0 \exp\left(-\gamma r^2\right)$, где r – расстояние между двумя светлячками, β_0 – привлекательность при $r = 0$; γ – параметр, выполняющий роль коэффициента поглощения света. Поскольку часто быстрее вычислить $1/\left(1+r^2\right)$, чем экспоненциальную функцию $\tau_j = (1-\phi)\tau_j + \phi\tau_0$, при вычислении привлекательности можно использовать выражением: $\beta_2 = \dfrac{\beta_0}{1+\gamma r^2}$. В ряде реализаций алгоритма предлагают использовать монотонно убывающую функцию со степенью ≥ 1.

Движение светлячка i_1, которого притягивает более привлекательный светлячок j_1, определяется с помощью следующим образом: $x_i = x_i + \beta_0 \times \times \exp\left(-\gamma r^2\right)\left(x_j - x_i\right) + \alpha_2 \varepsilon_i$, где $\alpha_2 \in [0,1]$ – параметр рандомизации, а ε_i – вектор случайных чисел, полученных из равномерного распределения. Например, распространенной формой ε_i является $\left(\text{rand} - \dfrac{1}{2}\right)$ [19], [23], где rand – случайное число из диапазона [0,1].

В некоторых работах для улучшения процесса сходимости алгоритма параметр α рекомендуют уменьшать с каждой следующей итерацией t с помощью следующего выражения: $\alpha = \alpha_\infty + + (\alpha_0 - \alpha_\infty)e^t$, где α_0 – начальное значение, α_∞ – конечное.

Схема работы алгоритма FA выглядит следующим образом:

– инициализация начальной популяции светлячков S (случайное создание и вычисление значений целевой функции $f\left(x_i^0\right)$ в начальных точках x_i^0, $i \in [1,|S|]$);

– циклическое сравнение пар светлячков (если $f(x_i) < f\left(x_j\right)$, то происходит перемещение i к j);

– обновляются значения целевых функций для измененных агентов;

– если выполнено условие окончания, то возвращение лучшего агента из популяции, иначе – переход к циклическому сравнению пар светлячков.

Стоит отметить два частных случая алгоритма FA, когда коэффициент поглощения $\gamma \to 0$ и $\gamma \to \infty$. В случае $\gamma \to 0$ среда становится прозрачной, а коэффициент привлекательности становится постоянным $\beta \approx \beta_0$. Это эквивалентно тому, что интенсивность света не уменьшается в среде и каждый светлячок виден в любой точке. В случае $\gamma \to \infty$ привлекательность стремится к дельта-функции Дирака, и привлекательность светлячков близка к нулю. Это соответствует случайному блужданию агентов, и FA преобразуется в алгоритм случайного поиска.

Параметр γ рекомендуется определять в зависимости от размерности пространства поиска, а именно: $\gamma = \dfrac{\gamma_0}{r_{\max}}$ или $\gamma = \dfrac{\gamma_0}{r_{\max}^2}$ ($\gamma_0 \in [0,1]$, $r_{\max}$ – максимальное расстояние между двумя светлячками).

Алгоритм FA имеет два внутренних цикла при сравнении всех пар агентов популяции и один внешний цикл для итераций. Поэтому в худшем случае временную сложность можно оценить как O(popSize × T × N), где T – количество итераций, N – размерность задачи.

Алгоритм летучей мыши. Алгоритм, предложенный Синь-Шэ Яном в 2010 г., имитирует эхолокационное поведение летучих мышей [25], [26]. Летучие мыши излучают звуковые импульсы и анализируют эхо для локации добычи. В алгоритме это соответствует поиску оптимального решения. Каждая «мышь» в популяции представляет собой отдельное решение задачи (кандидат на покрывающее множество). Решение кодируется битовым вектором $\mathbf{V} = [v_1, v_2, ..., v_n]$, где $v_i = 1$ – множество S_i включено в решение, в противном случае $v_i = 0$. Функция пригодности (Fitness) такая же, как в алгоритме поиска кукушки.

Алгоритм начинается с генерации N начальных решений. Для каждой «мыши» инициализируются

– частота $\lambda_i \in [\lambda_{\min}, \lambda_{\max}]$, где $\lambda_{\min}, \lambda_{\max}$ – минимальная и максимальная частоты;

– громкость $\psi_i \in [\psi_{\min}, \psi_{\max}]$, $\psi_{\min}, \psi_{\max}$ – минимальная и максимальная громкости;

– скорость импульсов $\gamma_i \in [0,1]$.

Для генерации новых решений используются формулы

– $\lambda_i(t+1) = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \cdot \text{rand}(\ )$, где $\text{rand}(\ ) \in [0,1]$ – случайная величина;

– $\gamma_i(t+1) = \gamma_i(t) + \left(v_i(t) - v_{\text{best}}\right) \cdot \lambda_i$, где v_{best} – лучшее на данный момент решение;

– $v_i(t+1) = v_i(t) + \gamma_i(t+1)$.

С вероятностью γ_i выполняется локальная оптимизация вокруг текущего лучшего решения. Если новое решение лучше текущего, то с вероятностью ψ_i решение принимается. Также увеличивается скорость импульсов (ускоряется поиск) и снижается громкость:

– $\psi_i(t+1) = \alpha_3 \psi_i(t)$, где α_3 – коэффициент затухания громкости;

– $\gamma_i(t+1) = \gamma_i(0)\left(1 - e^{-\sigma t}\right)$, где σ – коэффициент увеличения скорости импульсов.

Отличительной чертой данного алгоритма служит автоматическая балансировка параметров, позволяющая сочетать интенсивный локальный поиск с возможностью выхода из локальных оптимумов.

Алгоритм стада криля. Метаэвристический алгоритм, вдохновленный поведением стай мелких рачков-криля в океане [27]. Выживание отдельной особи зависит от двух ключевых факторов: движение к пище (глобальный оптимум) и взаимодействие с соседями (локальный поиск). В контексте задачи покрытия множества каждая особь криля представляет собой одно возможное решение – кандидата на покрывающее множество. Кодировка решения и функция пригодности такие же, как в алгоритме летучей мыши.

Случайным образом создается стадо из N особей (начальных решений). Для каждой особи i в стаде вычисляется ее новое положение по трем компонентам:

– движение, индуцированное соседями (Local Search): $R_i = \sum_{j \in \text{coc}} \left(V_j - V\right)_i \Big/ \left\|V_j - V_i\right\|$ (особи отталкиваются и притягиваются друг к другу, стремясь занять перспективные позиции в пространстве состояний), где V_i, V_j – решения, совпадающие с положением особей;

– движение в направлении пищи (Global Search): $E_i = E_{\text{best}} V_{\text{best}} - E_i V_i$, где V_{best} – лучшее найденное на данный момент решение во всем стаде (это движение направляет всех особей к глобально лучшей области), E_{best} – лучшее значение целевой функции;

– случайная диффузия (Diversification): $D_i = \delta\left(1 - t/t_{\max}\right) \cdot \text{SV}$, SV – случайный вектор, – этот компонент добавляет случайное возмущение, чтобы избежать преждевременного схода в локальный оптимум (его влияние уменьшается со временем), $t_{\max}$ – время окончания работы алгоритма.

Новое положение особи вычисляется как $V_i(t+1) = V_i(t) + \Delta t\left(R_i + E_i + D_i\right)$, где $\Delta t = 1$ – квант времени для перерасчета положения особей. Поскольку задача дискретная, полученные непрерывные значения координат $V_i(t+1)$ затем преобразуются в 0 или 1 с помощью пороговой функции (например, сигмоиды).

Оптимизатор серого волка. Метаэвристический алгоритм, вдохновленный социальной иерархией и охотничьей тактикой стаи серых волков [28]. Волки в стае делятся на четыре ранга:

– альфа (α_4) – наилучшее решение в популяции;

– бета (β_3) – второе и третье по качеству решения;

– дельта (δ_1) – остальные возможные решения;

– омега (ω_1) – все остальные возможные решения.

Основные этапы охоты:

– поиск и окружение добычи (случайный поиск в пространстве решений);

– загон добычи (поиск сужается вокруг лучших решений: $\alpha_4, \beta_3, \delta_1$);

– атака (локальный поиск и окончательное приближение к оптимуму).

К сожалению, не удалось найти конкретного описания применения алгоритма GWO именно для задачи покрытия множества. При желании это нетрудно сделать на основе формализации из других алгоритмов.

Алгоритм оптимизации «мотылек–пламя». Метаэвристический алгоритм оптимизации, основанный на природном поведении мотыльков при навигации [29]. Алгоритм имитирует поведение мотыльков, которые:

– используют источники света для навигации;

– поддерживают постоянный угол к источнику света;

– движутся по спиральным траекториям вокруг ярких объектов.

Для решения задачи покрытия этот алгоритм не применялся, но нет принципиальных трудностей это сделать.

Алгоритм стрекозы. Метаэвристический алгоритм, вдохновленный стайным поведением стрекоз в природе [30]. Его работа построена на моделировании пяти основных принципов поведения в стае:

– разделение – избегание столкновений с соседними особями в стае;

– выравнивание – соответствие скорости соседним особям;

– сцепление – стремление к центру массы стаи;

– притяжение к пище – движение в сторону источников пищи;

– отталкивание от врага – движение от внешних угроз.

В алгоритме эти принципы переводятся в математические операции, которые обновляют позиции «стрекоз» (решений-кандидатов) в пространстве поиска, балансируя между исследованием (глобальным поиском) и использованием (локальным поиском). Чтобы применить алгоритм к решению задачи покрытия множества, необходимо выполнить следующие шаги адаптации: кодирование решения и определение функции пригодности. Процесс оптимизации традиционен для метаэвристических алгоритмов: инициализация, оценка, обновление позиций, завершение.

Алгоритм оптимизации кита. Алгоритм имитирует охотничье поведение горбатых китов [31]. Киты создают спиралевидные пузырьковые сети, поднимаясь к поверхности воды, что позволяет им эффективно ловить добычу.

Ключевые механизмы имитации поведения китов:

– киты «знают» положение лучшей добычи (оптимального решения);

– другие особи обновляют свои позиции в направлении лучшего кита (окружение добычи);

– киты движутся по спирали вокруг добычи (метод пузырьковой сети);

– выбирается случайный кит вместо лучшего (поиск добычи).

При окружении добычи позиции китов пересчитываются по формулам: $D = C\left|X_{\text{best}}(t) - X(t)\right|$ и $X(t+1) = X_{\text{best}} - NVD$, где X_{best} – позиция лучшего решения; NV – направляющий вектор; C – коэффициент влияния лучшего решения; D – расстояние до лучшего решения.

При движении по спирали вокруг добычи позиции китов пересчитываются по формуле: $X(t+1) = De^{b\pi}\cos(b\pi) + X_{\text{best}}(t)$, где b – параметр, определяющий форму спирали.

При $NV < 1$ киты ищут добычу случайным образом: $D = C\left|X_{\text{rand}}(t) - X(t)\right|$ и $X(t+1) = X_{\text{rand}} - NVD$, где X_{rand} – позиция случайного кита.

Направляющий вектор определяется как $NV = 2\alpha_5 r_1 - \alpha_5$, где α_5 – управляющий коэффициент, который линейно убывает от 2 до 0 по ходу итераций. Коэффициент C рассчитывается по формуле: $C = 2r_2$, r_1, r_2 – случайные числа в диапазоне [0,1].

Алгоритм роя сальпа. Алгоритм имитирует поведение роя сальп (морских беспозвоночных) в природе, где они выстраиваются в цепочки для коллективного перемещения и поиска пищи. Алгоритм имитирует поведение лидера и последователей, механизмы поиска пищи и групповую динамику.

В рое сальп имеется лидер – сальп с наилучшим решением. Остальные члены роя следуют за лидером. Характеристики позиции лидера: является результатом предыдущих итераций; обновляется при нахождении лучшего решения; служит ориентиром для всего роя и может меняться в процессе оптимизации. Кроме позиции лидера в алгоритме используется позиция пищи. Характеристики позиции пищи: представляет собой идеальное или целевое состояние; может быть статичной или динамической; служит ориентиром для поиска и не всегда совпадает с текущим лучшим решением. Чтобы избежать предвзятости начальная позиция пищи может быть выбрана случайным образом. Затем позиция пищи может быть определена на основе математических критериев, например как центр тяжести лучших решений.

Метаэвристики, основанные на физике. *Оптимизация центральной силы* – метаэвристический алгоритм, вдохновленный законами небесной механики. Этот алгоритм подробно рассмотрен в литературе. Отличительной его чертой служит использование притяжения к одной точке, называемой центром силы, т. е. использует упрощенную модель гравитационного взаимодействия. Так как взаимодействие небесных тел понятно всем, кто имеет не слишком глубокие знания по физике, то в данном обзоре его детали не рассматриваются.

Алгоритм гравитационного поиска. Алгоритм гравитационного поиска моделирует взаимодействие всех частиц между собой (учитывает взаимные гравитационные силы) [32]. Он имеет более сложную математическую модель и требует больше вычислительных ресурсов по сравнению с алгоритмом оптимизации центральной силы, но за счет этого получает более качественные решения.

Поиск системы магнитного заряда. Метаэвристический алгоритм оптимизации, основанный на принципах электромагнетизма и взаимодействии заряженных частиц. Его физические основы следующие: закон Кулона для моделирования взаимодействия частиц; электромагнитные силы для направления поиска; законы сохранения энергии и заряда. Основные компоненты алгоритма: частицы системы (представляют потенциальные решения, обладают зарядом и положением, взаимодействуют между собой) и силовые поля (кулоновские силы, потенциальные поля, гравитационные компоненты). В задаче покрытия множества потенциальные поля могут использоваться для: привлечения агентов к неохваченным областям; отталкивания от уже покрытых зон; балансировки нагрузки между агентами. Гравитационные компоненты помогают в: оптимизации размещения объектов; балансировки нагрузки; минимизации перекрытий.

Оптимизация электромагнитного поля. Метаэвристический метод, основанный на принципах электромагнетизма и взаимодействии заряженных частиц. В отличие от алгоритма поиска системы магнитного заряда, он использует полный набор электромагнитных взаимодействий, а также векторные поля и поляризацию частиц. За счет этого он дает более качественные решения, но требует больше вычислительных ресурсов.

Оптимизация процесса испарения воды. Метаэвристический метод, вдохновленный природным процессом испарения воды и перехода ее в парообразное состояние. Молекулы жидкости представляют потенциальные решения, обладают энергией и взаимодействуют между собой. Параметры данного процесса – это температура системы, энергия активации и скорость испарения. Процесс оптимизации состоит: из моделирования теплового движения; расчета энергии молекул; перехода в газообразное состояние и формирования новых решений.

Оптимальная оптимизация инерции. Метаэвристический алгоритм оптимизации, основанный на концепции оптимальной инерции и динамики движения частиц в пространстве поиска. Он базируется на моделировании частиц с переменной инерцией и адаптивном изменении параметров инерции. Компоненты алгоритма – это частицы системы и ее параметры. Частицы системы представляют потенциальные решения. Они обладают характеристиками скорости и инерции и взаимодействуют через обмен информацией. К параметрам системы относятся: коэффициент инерции; скорость движения; локальная и глобальная память; адаптивные факторы.

Многовершинная оптимизация. Метаэвристический алгоритм оптимизации, вдохновленный космологическими концепциями мультивселенной и черных дыр [33]. Алгоритм базируется: на моделировании взаимодействия между вселенными; использовании концепции черных дыр; механизмах изменения объема и формы. Компоненты алгоритма – вселенные и космологические параметры. Вселенные представляют потенциальные решения, обладают уникальными характеристиками и взаимодействуют между собой. К параметрам алгоритма относятся: плотность материи; гравитационные силы; квантовые эффекты и черные дыры.

Оптимизация теплообмена. Метаэвристический алгоритм оптимизации, основанный на физических принципах теплового расширения материалов. Алгоритм базируется на: моделировании теплового расширения; использовании температурных градиентов; механизмах изменения объема и формы. Компонентами алгоритма служат элементы системы и физические параметры. Элементы системы представляют потенциальные решения, обладают температурными характеристиками и взаимодействуют через теплопередачу. К параметрам алгоритма относятся: температура элементов; коэффициент расширения; теплопроводность и плотность материала.

Алгоритм черной дыры. Метаэвристический алгоритм оптимизации, вдохновленный феноменом черной дыры. Черная дыра (ЧД) – область пространства, в которой сосредоточено такое количество массы, что ближайший объект не может избежать ее гравитационной силы. Любое тело, которое попадает в черную дыру, в том числе и свет, не может ее покинуть и будет поглощено.

Алгоритм BHA впервые был представлен в [20] в качестве нового оптимизационного алгоритма для кластеризации данных. Подобно другим алгоритмам на основе популяции, алгоритм начинает свою работу с создания популяции потенциальных решений, которые в рамках алгоритма называются «звездами». Кроме того, среди звезд выбирается «черная дыра». ЧД – фиксированная звезда в пространстве поиска, которая имеет наилучшую пригодность (значение целевой функции).

После инициализации в алгоритме выполняется вращение звезд вокруг ЧД (т. е. вокруг наиболее приспособленного агента). Это вращение осуществляется с помощью оператора, который перемещает все звезды на каждой итерации (ЧД при этом не меняет своего положения):

$$x_i^j(t+1) = x_i^j(t) + \text{rand}\left[x_{\text{BH}}^j - x_i^j(t)\right], \quad \forall i \in \{1, ..., N_1\},$$

где $x_i^j(t)$ и $x_i^j(t+1)$ – переменные решения i на итерациях t и $t+1$; N_1 – количество звезд в популяции; x_{BH}^j – переменная лучшего решения (ЧД) в пространстве поиска; rand – случайное число в интервале [0,1].

После перемещения к ЧД звезда может достичь точки, при которой ее пригодность (т. е. значение целевой функции) становится лучше, чем у ЧД. В таком случае ЧД и данная звезда меняются местами, после чего алгоритм продолжит работу уже с новой ЧД.

Кроме того, в процессе перемещения звезда может пересечь горизонт событий и будет поглощена черной дырой. При этом она заменяется на новую ЧД случайным образом. Следовательно, размер популяции остается неизменным.

Радиус горизонта события рассчитывается следующим образом: $R = f_{\text{BH}} \Big/ \sum_{i=1}^{N} f_i$, где f_{BH} – пригодность черной дыры; f_i – пригодность i-го решения.

Звезда поглощается черной дырой, когда расстояние между ними меньше, чем радиус горизонта события. В публикациях расстояние определяется по-разному – например, евклидово расстояние либо же иная разновидность векторного расстояния. Однако в рамках задачи покрытия множества звезды имеют вид двоичных векторов, т. е. векторные расстояния > 1. При этом радиус горизонта событий ограничен в пределах $\left(0, \frac{1}{N}\right]$. Иногда используют другое соотношение для определения факта пересечения горизонта события: $\left|f_{\text{BH}} - f_i\right| < sR$, $s \in (0,1]$, где s – дополнительный параметр для настройки [34].

Временная сложность алгоритма BHA такая же, как в алгоритме светлячка.

Метаэвристики на основе эволюции. *Генетический алгоритм* (*GA*). Генетические алгоритмы вероятностного характера, основанные на правилах естественного отбора и наследования, были предложены Джоном Холландом (Jonn Holland) в 1975 г. Новый алгоритм получил название «репродуктивный план Холланда» [35]. Идеи Холланда развили его ученики Кеннет Де Йонг (Kennet De Jong) и Дэвид Е. Голдберг (David E. Goldberg). Благодаря им был создан классический генетический алгоритм (именно алгоритм Голдберга и получил название «генетический алгоритм») [36].

Рассмотрим основные понятия, применяемые в генетических алгоритмах [37].

Особь (генетический код, индивидуум) – вектор (или строка) чисел (обычно бинарная строка из нулей и единиц.

Расстояние – расстояние по Хеммингу между бинарными особями.

Кроссинговер – операция, при которой две особи обмениваются своими частями.

Мутация – случайное изменение одной или нескольких позиций в особи.

Популяция – совокупность особей.

Пригодность (приспособленность) – критерий или функция, экстремум которой следует найти.

Терминология генетических алгоритмов представляет собой синтез собственно генетических и искусственных понятий.

Сущность любого генетического алгоритма заключается в выполнении следующих шагов:

– заполнение начальной популяции;

– выбор родителей;

– скрещивание (кроссинговер);

– мутация;

– замещение худшей особи;

– повторение заданного числа шагов начиная с выбора родителей (или при схождении популяции).

Схождением называется такое состояние популяции, при котором перестают появляться новые особи. В такой ситуации кроссинговер и мутация не изменяют популяции, поскольку создаваемые потомки либо совпадают с особями из популяции, либо вымирают, так как имеют меньшую приспособленность, чем минимальная приспособленность в популяции.

Даже на самом первом этапе имеется большая свобода в выборе алгоритма формирования начальной популяции, от которой зависит конечный результат генетического алгоритма.

Во-первых, нужно задать размер популяции. Если популяция маленькая, то алгоритм быстро сходится, но качество решения таблицы покрытия, скорее всего, будет далеко от минимального, и ничего не остается, как экспериментально подобрать размер популяции под максимально допустимое время схождения.

Во-вторых, попробовать различные варианты алгоритма формирования начальной популяции, экспериментально оценивая каждый из них. На взгляд авторов, при применении генетического алгоритма для решения задачи покрытия множества заслуживает внимания следующий подход к формированию начальной популяции:

– найти решение таблицы покрытия каким-либо «жадным» алгоритмом и включить это решение в начальную популяцию (первая особь);

– при формировании очередной особи $r_1/2$ строк выбирать случайным образом (с равной вероятностью) из множества строк таблицы покрытия, где r_1 – число строк в решении «жадным» алгоритмом (число единиц в первой особи);

– для определения остальных строк использовать оператор осуществимости, псевдокод которого приведен ранее (или его модификацию с эвристикой «жадного» алгоритма, который использовался для определения первой особи).

Такой подход позволяет разбросать особи начальной популяции по всему пространству возможных особей, увеличивая вероятность нахождения оптимального или близкого к оптимальному решения, в процессе отбора локальных экстремумов.

Рассмотрим следующий шаг генетического алгоритма – выбор родителей. Панмиксия – самый простой оператор отбора. В соответствии с ним каждому члену популяции сопоставляется случайное целое число на отрезке $[1; n]$, где n – количество особей в популяции. Будем рассматривать эти числа как номера второй особи, которая примет участие в скрещивании. При таком выборе какие-то из членов популяции не будут участвовать в процессе размножения, так как образуют пару сами с собой. Какие-то члены популяции примут участие в процессе воспроизводства неоднократно с различными особями популяции. Использование панмиксии предполагает, что процесс изменения популяции состоит из смены поколений.

Инбридинг такой способ отбора, когда первый родитель выбирается случайным образом, а второй родитель – член популяции, ближайший к первому (минимальное расстояние по Хеммингу).

При аутбридинге также используется понятие схожести особей. Однако теперь брачные пары формируются из максимально далеких особей.

Инбридинг и аутбридинг по-разному влияют на поведение генетического алгоритма. Инбридинг характерен поиском в локальных узлах, что приводит к разбиению популяции на отдельные локальные группы. Аутбридинг же направлен на предупреждение сходимости алгоритма к уже найденным решениям, заставляя алгоритм просматривать новые, неисследованные области.

Инбридинг и аутбридинг связаны увеличением сложности вычислений, так как нужно определять расстояние между особями. На наш взгляд, для решения задачи покрытия множества больше подходит случайный выбор родителей. При этом можно не использовать равновероятный выбор, а сделать вероятность выбора особи зависящей от значения ее функции пригодности.

Ряд авторов предлагают более изощренные способы селекции, преимущества которых не очевидны. Также кажется искусственным разбиение развития популяции на поколения. Достаточно в качестве параметра генетического алгоритма использовать максимальное количество скрещиваний. А еще лучше использовать для завершения работы алгоритма индикатор схождения популяции.

Оператор рекомбинации применяют сразу после оператора отбора родителей для получения новых особей-потомков. Смысл рекомбинации состоит в том, что созданные потомки должны наследовать генную информацию от обоих родителей. Различают дискретную рекомбинацию и кроссинговер. Так как дискретная рекомбинация применяется к особям с вещественными генами, то она не подходит к решению задачи покрытия множества.

Рассмотрим рекомбинацию бинарных строк, которую называют кроссинговером или скрещиванием. Одноточечный кроссинговер заключается в том, что случайным образом определяется точка разрыва внутри особи, в которой они делятся на две части и обмениваются ими. В двухточечном кроссинговере (многоточечном) особи рассматриваются как циклы, которые формируются соединением концов особи вместе. Для замены сегментов одного цикла сегментом другого цикла требуется выбор двух точек разреза. При использовании однородного кроссинговера создается маска кроссинговера. Обычно это строка из 0 и 1, размерность которой совпадает с размерностью особей. Гены маски разыгрываются как случайные двоичные числа (равновероятные). Согласно

значениям маски, геном потомка становится ген первой (если ген маски равен 0) или второй (если ген маски равен 1) особи-родителя. В триадном кроссинговере после отбора пары родителей из остальных членов популяции случайным образом выбирается особь, которую используют в качестве маски. Причем определенный процент генов маски мутируют. Существует целый ряд более сложных кроссинговеров: перестановочный кроссинговер, кроссинговер с уменьшением замены и т. п.

Нетрудно заметить, что список возможных кроссинговеров настолько велик, что порождает целое множество разновидностей генетических алгоритмов. На наш взгляд, не стоит отвлекаться на выбор типа кроссинговера при использовании генетического алгоритма для решения задачи покрытия множества. Наиболее подходящим кроссинговером для этого видится однородный.

После процесса воспроизводства, как правило, выполняют мутацию. Данный оператор необходим для «выбивания» популяции из локального экстремума с целью предупреждения преждевременной сходимости. Это обеспечивается тем, что изменяется случайно выбранный ген особи. Как и кроссинговер, мутации могут проводиться не по одной, а по нескольким случайным точкам. Вероятность мутации может быть или фиксированным случайным числом на отрезке [0, 1] или функцией от какой-либо характеристики решаемой задачи или текущей популяции особей.

Для создания новой популяции можно использовать различные методы отбора особей: отбор усечением, элитарный отбор (с использованием промежуточной популяции), отбор вытеснением, метод Больцмана (метод отжига) и т. п.

Мы рекомендуем для решения задачи покрытия множества отказаться от использования метафоры «поколения». После мутации и выполнения оператора осуществимости вычислить значение функции пригодности для особей-потомков, которые не совпадают с особями текущей популяции (все особи в популяции должны отличаться друг от друга). Особь-потомок вытесняет особь с худшим значением функции пригодности, если ее функция пригодности лучше. При решении невзвешенной задачи покрытия функция пригодности – это число единиц в особи (число строк в решении ТП), поэтому почти всегда будет не одна худшая особь, а подмножество худших особей. Для вытеснения равновероятно выбирается любая из таких особей.

Следует отметить, что генетические алгоритмы – это целый класс алгоритмов, предназначенных для решения разнообразных задач, а именно:

– канонический генетический алгоритм;

– гибридный алгоритм;

– алгоритмы с нефиксированным размером популяции;

– параллельные алгоритмы и т. д.

Канонический алгоритм – это алгоритм Д. Холланда, развитый его учениками.

Идея гибридных алгоритмов заключается в сочетании генетического алгоритма с другим, не генетическим методом поиска, подходящим к рассматриваемой задаче. Сгенерированные потомки оптимизируются выбранным методом и только затем проверяется возможность их включения в популяцию. Известно, что генетический алгоритм способен быстро найти во всей области поиска хорошие решения, но имеются трудности в получении наилучших из них. Обычный оптимизационный метод может быстро достичь локального максимума, но не может найти глобальный. Сочетание двух алгоритмов позволяет использовать преимущества обоих.

В генетическом алгоритме с нефиксированным размером популяции каждой особи приписывается максимальный возраст (число поколений, через которые она погибает). Возраст каждой особи индивидуален и зависит от ее приспособленности. В каждом поколении создается дополнительная популяция из потомков. Из основной популяции удаляются те особи, срок жизни которых истек, и добавляются потомки из промежуточной популяции.

Генетические алгоритмы применяются и при параллельных вычислениях. При этом формируется несколько живущих отдельно популяций. На этапе формирования нового поколения по определенному правилу отбираются особи из разных популяций.

Одна из серьезных проблем, возникающих при использовании генетических алгоритмов, – преждевременная сходимость. Обычно используют следующие пути устранения преждевременной сходимости: увеличение размера популяции, применение самоадаптирующихся генетических алгоритмов и создание «банка» заменяемых особей.

В первом случае с увеличением размера популяции в ней растет многообразие генотипа, однако, с другой стороны, увеличивается размер требуемой памяти и время работы алгоритма.

Более эффективен второй способ. Самоадаптация заключается в изменении вероятности мутаций в зависимости от свойств текущей популяции.

В третьем подходе создается массив для сохранения особей, генотип которых был утерян при формировании новых поколений, и временами эти особи добавляются в популяцию.

Дифференциальная эволюция. Популяционный алгоритм оптимизации, основанный на принципах эволюции и на работе с вещественными числами. Алгоритм базируется: на популяционном подходе; механизмах мутации; операциях кроссовера; отбора лучших решений. Компоненты алгоритма: популяции и операторы эволюции. Популяция – это набор потенциальных решений. Дня нее используется случайная инициализация с вещественными значениями параметров. К операторам эволюции относятся: мутация; кроссовер; селекция. В отличие от генетических алгоритмов, дифференциальная эволюция работает не с дискретными структурами (хромосомами), а с векторными операциями и вещественными числами.

Процесс оптимизации состоит из генерации пробных векторов; оценки целевой функции; обновления популяции и проверки условий покрытия.

Мутация выполняется по формуле: $\mathbf{v}_i = x_{r_1} + F\left(x_{r_2} - x_{r_3}\right)$. Вектор $\mathbf{v}_i$ – это промежуточное решение, которое генерируется в процессе мутации и служит основой для последующего кроссовера в алгоритме дифференциальной эволюции. В процессе создания промежуточного решения берется базовый вектор, к нему добавляется разность между двумя другими векторами популяции и результат масштабируется коэффициентом *F*.

Пусть $\mathbf{V} = \left[v_1, v_2, ..., v_n\right]$, где $v_i = 1$ – множество S_i включено в решение, в противном случае $v_i = 0$.

Кроссовер – это ключевой этап в алгоритме DE, который определяет, как будет формироваться пробный вектор на основе базового и мутированного векторов. Базовая формула кроссовера: $\begin{cases} u_{ij} = v_{ij}, \text{ если } \text{rand}_j \le CR \text{ или } j_{\text{rnd}} \\ x_{ij}, \text{ иначе} \end{cases}$, где u_{ij}, v_{ij}, x_{ij} – элементы пробного, мутированного, базового векторов; *CR* – коэффициент кроссовера; j_{rnd} – случайно выбранный индекс. j_{rnd} предотвращает преждевременную сходимость (гарантирует попадание хотя бы одного элемента из *v*).

Эволюционное программирование. Метод эволюционных вычислений, основанный на принципах естественного отбора и адаптации, применяемый для решения сложных оптимизационных задач [38]. Компоненты алгоритма: представление решения и операторов эволюции.

В представлении решения используются: векторы действительных чисел, бинарное кодирование, специальные структуры данных для покрытия. Векторы действительных чисел предназначены для кодирования вероятностей включения элементов; представления весов подмножеств; оценки качества покрытия. Специальные структуры данных – это способы организации информации, оптимизированные для эффективного решения задачи покрытия множества (матрицы покрытия, списки смежности, деревья покрытий). Матрица покрытия позволяет эффективно проверять полноту покрытия. Списки смежности эффективны при динамических изменениях. Деревья покрытия поддерживают операции объединения и пересечения.

Различают следующие операторы эволюции: мутация, селекция, адаптация параметров. Это очень похоже на алгоритм дифференциальной эволюции, но эволюционное программирование сосредоточено на принципах естественного отбора, фокусируется на эволюции программ (стратегий) и использует более общие эволюционные механизмы.

Эволюционная стратегия. Метод эволюционных вычислений, основанный на принципах естественного отбора и адаптации, специально адаптированный для решения задачи покрытия множества. Компоненты алгоритма: представление решения и параметры стратегии.

В представлении решения используются: вещественные числа для кодирования; вероятности включения элементов; векторное представление подмножеств. Параметры стратегии: размер популяции; параметры мутации и отбора. В алгоритме применяется бинаризация решений для получения окончательного покрытия и адаптивное кодирование параметров. Отличается от алгоритма эволюционного программирования тем, что фокусируется на оптимизации непрерывных параметров и использует самоадаптацию параметров мутации.

Генетическое программирование (*ГП*). Метод эволюционных вычислений, который использует принципы естественного отбора для автоматиче-

ского создания компьютерных программ или выражений, способных решать задачу покрытия множества [39]. Компоненты алгоритма: представление решения и операторы ГП.

В представлении решения используются: деревья выражений; программы как иерархические структуры и кодирование стратегий покрытия. Операторы ГП это: кроссовер и мутация деревьев; репликация успешных решений. Кодирование решений заключается в представлении подмножеств как узлов дерева, использовании логических операций и условий покрытия.

Инкрементальное обучение на основе вероятности. Для задачи покрытия множества предполагает постепенное построение решения через обновление вероятностей выбора элементов в покрытии на основе накопленного опыта. Ключевые механизмы алгоритма следующие: вероятностная модель элементов (каждому элементу универсума присваивается вероятность включения в покрытие, вероятности инициализируются равномерно или эвристически); инкрементальное обновление вероятностей и элементы выбираются вероятностно на каждом шаге.

Формула инкрементального обновления: $P_i(t+1) = P_i(t) + \alpha_6(\text{reward} - \text{baseline}) \times \text{indicator}$, где α_6 – скорость обучения α_5; reward – оценка полезности элемента; baseline – средний уровень полезности; indicator – индикатор использования элемента.

К сожалению, не удалось найти научные работы, которые бы непосредственно описывали применение инкрементального обучения на основе вероятности для решения задачи покрытия множества. На наш взгляд, идеи метода перспективны для применения в решении задачи покрытия множества.

Метаэвристики, основанные на человеке. *Империалистический алгоритм конкуренции.* Метаэвристический алгоритм оптимизации, вдохновленный процессами конкуренции между империями [40]. Каждая империя представляет собой возможное решение задачи. Колонии – это варианты решений, которые могут быть улучшены. Империи конкурируют друг с другом, поглощая слабые колонии и улучшая свои решения. Процесс продолжается до достижения оптимального или близкого к оптимальному решения.

Применение алгоритма к покрытию множества состоит в следующем:

– каждая империя представляет собой набор подмножеств, покрывающих элементы множества;

– алгоритм оценивает качество покрытия (например, по количеству использованных подмножеств);

– слабые империи и колонии устраняются, а сильные – развиваются, что позволяет находить оптимальное покрытие.

Компоненты алгоритма: начальные страны (случайные решения в пространстве поиска задачи оптимизации) и империи, состоящие из империалистического государства (лучшее решение) и колоний (остальные решения группы).

Инициализация алгоритма заключается в генерации случайных начальных решений, оценке их с помощью целевой функции и формировании начальных империй. Основной цикл работы алгоритма содержит:

– ассимиляцию (движение колоний к империалистическому государству);

– революцию (случайные изменения характеристик некоторых стран);

– конкуренцию (борьба империй за колонии) и обмен позициями между колонией и империалистическим государством.

Ассимиляция приближает колонии к империалистическому государству, улучшая решения «заимствованием» характеристик. Революция – это внезапные случайные изменения стран, позволяющие избежать локальных оптимумов. Империалистическая конкуренция заключается в захвате сильными империями колоний слабых империй, что обеспечивает глобальный поиск решений.

Алгоритм формирования человеческих групп. Метаэвристический алгоритм оптимизации, вдохновленный процессами формирования социальных групп. Компоненты алгоритма:

– индивиды (потенциальные решения задачи);

– группы (объединение индивидов со схожими характеристиками);

– лидеры групп (лучшие решения в каждой группе);

– социальные связи (взаимодействие между индивидами).

Инициализация заключается в создании начальной популяции решений, формировании случайных групп и определении лидеров групп

(на основе оценки качества решения). Процесс оптимизации состоит из обмена информацией между членами группы; обучения у лидеров групп; перемещения между группами и формирования новых групп.

Алгоритм оптимизации, основанный на процессе преподавания и обучения [41]. Компоненты алгоритма: индивиды (потенциальные решения задачи); группы (объединение индивидов со схожими характеристиками); лидеры групп (лучшие решения в каждой группе); социальные связи (взаимодействие между индивидами). Компоненты алгоритма: учителя (лучшие решения в популяции); студенты (остальные решения); процесс обучения (обмен знаниями между участниками); фазы алгоритма (фаза преподавания, фаза обучения). В процессе оптимизации выполняется периодическая переоценка учителей.

Алгоритм оптимизации, вдохновленный стратегией и тактикой игры в футбол. Основные компоненты алгоритма: игроки (потенциальные решения задачи); команды (группы решений со схожими характеристиками); поле (пространство поиска решений); мяч (целевая функция оптимизации). При инициализации алгоритма создается начальная популяция «игроков», формируются «команды» решений и распределяются роли между участниками. В процессе оптимизации используются: атака (поиск новых решений); защита (сохранение лучших решений); передачи (обмен информации между игроками); тактические маневры (улучшение решений). Стратегия атаки направлена на расширение покрытия. Стратегия защиты обеспечивает качество покрытия. Командная работа – синхронизация решений. Роли игроков: форварды (поиск новых решений); полузащитники (оптимизация существующих решений); защитники (сохранение лучших решений). Тактические элементы: пасы (обмен информацией между решениями); дриблинг (локальная оптимизация); удары по воротам (проверка целевых функций).

Метаэвристики на основе единичных данных. ***Имитация отжига.*** Это стохастический алгоритм оптимизации, основанный на физическом процессе отжига металлов [42]. Физический аналог – процесс кристаллизации металлов при постепенном охлаждении. Математическая модель – вероятностный поиск оптимума с возможностью выхода из локальных минимумов. Компоненты алгоритма: начальное решение (произвольное покрытие множества); температура системы (управляющий параметр алгоритма); оператор изменения (механизм генерации новых покрытий).

При инициализации алгоритма выполняется генерация начального покрытия, установка начальной температуры и определение параметров охлаждения. Начальная температура – критически важный параметр алгоритма. Она определяет степень исследовательской способности алгоритма в начале работы, вероятность принятия худших решений и эффективность выхода из локальных оптимумов. Для определения начальной температуры используют эмпирический или аналитический подход. Формула охлаждения влияет на скорость сходимости алгоритма, качество получаемого решения и время работы алгоритма. Применяют охлаждения: экспоненциальное, линейное и логарифмическое.

Генерация нового состояния происходит следующим образом: новое решение генерируется на основе текущего; процесс носит случайный характер; изменение зависит от температуры.

Поиск табу. Метаэвристический метод оптимизации, использующий список табу (память о недавних изменениях) [43]. При инициализации алгоритма поиска с запретами выполняются следующие действия: создается пустое множество табу-списка () $\mathrm{Tabu}(i_0) = \varnothing$; выбирается начальное решение i_0; устанавливаются основные параметры алгоритма. Процесс инициализации включает: определение начальной длины списка запретов $(\lambda_1 \geq 0)$; установку правил формирования запретов; задание критериев обновления списка. Запрет накладывается на элементы, которые менялись в последних λ_1 шагах. При $\lambda_1 = 0$ получается стандартный локальный спуск.

Основной процесс алгоритма состоит из генерации соседних решений (добавление, удаление, замена подмножеств); оценки качества новых покрытий; выбор лучшего допустимого решения и обновления списка табу. Соседнее решение – это альтернативное решение, получаемое из текущего внесением определенных изменений, при этом оставаясь в пределах допустимого пространства поиска. Основные типы генерации: изменение отдельных элементов; перестановка элементов; замена компонентов; добавление/удале-

ние элементов. Процесс генерации включает: выбор базового решения (определение текущего состояния, анализ допустимых изменений, учет ограничений задачи); формирование окрестности (создание множества возможных изменений, проверка на соответствие ограничениям, оценка допустимости преобразований); отбор кандидатов (исключение запрещенных решений, учет списка табу, проверка на уникальность).

Управляемый локальный поиск. Метаэвристический метод, который модифицирует стандартный локальный поиск посредством введения системы штрафов для выхода из локальных оптимумов [44]. Компоненты алгоритма: целевая функция с модифицированным штрафом $(g(x) = \lambda_2 \alpha_6 \sum_{i=1}^{i=m} (I_i(x)\, p_i))$ и параметры управления (λ_2 – интенсивность поиска, α_6 – коэффициент балансировки штрафов, p_i – штраф за свойство i).

$I_i(x)$ – это индикаторная функция, которая отражает наличие определенных свойств или характеристик в решении. В контексте задачи покрытия множеств она может отражать: наличие избыточных подмножеств; недостаточное покрытие определенных элементов; часто встречающиеся конфигурации. Система штрафов позволяет избегать попадания в локальные оптимумы; исследовать новые области поиска; контролировать качество получаемых решений; адаптивно настраивать процесс оптимизации.

Для простых задач можно использовать итерационный локальный или стохастический локальный поиск. Существенным недостатком алгоритма итерационного локального поиска является отсутствие выхода из тупиковых ситуаций, что приводит к риску застревания в локальных оптимумах. В алгоритме стохастического локального поиска решения, как правило, имеют низкое качество.

Жадная рандомизированная процедура адаптивного поиска. Метаэвристический алгоритм для решения задач комбинаторной оптимизации, включая задачу покрытия множества [45]. Компоненты алгоритма: итеративный процесс построения решений; рандомизированное жадное построение; локальный поиск для улучшения решений; ограниченный список кандидатов.

Суть рандомизированного жадного построения заключается в гибридном методе создания решений, который сочетает принципы жадной эвристики (выбор наилучших элементов на каждом шаге, локально-оптимальные решения, последовательное улучшение) с элементами случайности (случайный выбор среди хороших решений, ограниченный список кандидатов). Ограниченный список кандидатов обеспечивает баланс жадности и случайности, а также контролирует разнообразие решений.

Заключение. В данной статье был проведен обзор метаэвристических алгоритмов решения NP-трудной задачи покрытия множества. Анализ показал, что существует большое разнообразие таких алгоритмов, причем каждая метаэвристика представляет собой не конкретный алгоритм, а порождает целое множество алгоритмов в зависимости от значений параметров алгоритма и характеристик его частей.

Эта черта метаэвристических алгоритмов затрудняет сравнение алгоритмов между собой. Поэтому остается актуальной задача автоматической генерации таблиц покрытия, на которых можно будет проводить экспериментальное сравнение алгоритмов. Основная проблема автоматической генерации таблиц покрытий заключается в том, чтобы параметры генератора позволяли навести порядок в их бесконечном множестве.

К современным тенденциям следует отнести разработку гибридных алгоритмов, сочетающих идеи разных метаэвристик, а также применение методов машинного обучения для подбора эффективного алгоритма (включая его параметры) при решении конкретных экземпляров задачи покрытия множества.

Список литературы

1. Karp R. M. Reducibility among combinatorial problems // Complexity of Comp. computations. Boston, MA: Springer, 1972. P. 85–103. doi: 10.1007/978-1-4684-2001-2_9.

2. Коновалов И. С., Остапенко С. С., Кобак В. Г. Сравнение эффективности работы точных и приближенных алгоритмов для решения задачи о покрытии множества // Вестн. ДонГТУ. 2017. Т. 17. № 3(90). С. 137–144. doi: 10.23947/1992-5980-2017-17-3-137-144.

3. Yang X.-Sh. Nature-inspired metaheuristic algorithms. 2nd ed. Frome, UK: Luniver press, 2010. 160 p.

4. Обзор метаэвристических методов оптимизации, применяемых при решении электроэнергетических задач / Р. А. Алехин, Ю. П. Кубарьков, Д. В. Зака-

мов, Д. В. Умяров // Вестн. СамГТУ. Сер.: Технические науки. 2019. № 3(63). С. 6–19.

5. Тимофеева О. П., Неимущев С. А., Неимущева Л. И. Исследование популяционных алгоритмов в решении задач непрерывной оптимизации // Тр. НГТУ им. Р. Е. Алексеева. 2018. № 4(123). С. 48–55.

6. Карпенко А. П. Основные сущности популяционных алгоритмов глобальной оптимизации. Опыт систематизации // Интернет-журн. «Науковедение». 2017. Т. 9. № 6(43). С. 41–67.

7. Adaptive black hole algorithm for solving the set covering problem / R. Soto, B. Crawford, R. Olivares, C. Taramasco, I. Figueroa, A. Gómez, C. Castro, F. Paredes // Math. Problems in Engin. 2018. Vol. 2018. Art. 2183214. P. 1–23. doi: 10.1155/2018/2183214.

8. Comparing cuckoo search, bee colony, firefly optimization, and electromagnetism-like algorithms for solving the set covering problem / R. Soto, B. Crawford, C. Galleguillos, J. Barraza, S. Lizama, A. Muñoz, J. Vilches, S. Misra, F. Paredes // Intern. Conf. on Computational Sci. and Its Appl. (ICCSA 2015). Springer, Cham, 2015. At: Lecture Notes in Comp. Sci. (LNTCS). Vol. 9155. P. 187–202. doi: 10.1007/978-3-319-21404-7_14.

9. Balas E., Carrera M. C. A dynamic subgradient-based branch-and-bound procedure for set covering // Operations Research. 1996. Vol. 44. № 6. P. 875–890. doi: 10.1287/opre.44.6.875.

10. Akhter F. A heuristic approach for minimum set cover problem // Int. J. Adv. Res. Artif. Intell (IJARAI). 2015. Vol. 4, no. 6. P. 40–45. doi: 10.14569/IJARAI.2015.040607.

11. Еремеев А. В. Генетический алгоритм для решения задачи о покрытии // Дискретный анализ и исследование операций. Сер. 2. 2000. Т. 7, № 1. С. 47–60.

12. Beasley J. E., Chu P. C. A genetic algorithm for the set covering problem // Eur. J. of Operational Research. 1996. Vol. 94, no. 2. P. 392–404.

13. Тюхин М. В., Ломазов В. А. Применение биоинспирированных алгоритмов роевого интеллекта для решения задач оптимизации // Вопр. устойчивого развития общества. 2021. № 6. С. 811–815.

14. Application of the artificial bee colony algorithm for solving the set covering problem / B. Crawford, R. Soto, R. Cuest, F. Paredes // The Sci. World J. 2014. Vol. 2014. Art. 189164. P. 1–8. doi: 10.1155/2014/189164.

15. A binary cat swarm optimization algorithm for the non-unicost set covering problem / B. Crawford, R. Soto, N. Berríos, F. Johnson, F. Paredes, C. Castro, E. Norero // Math. Problems in Engin. 2015. Vol. 2015. Art. 578541. P. 1–8. doi: 10.1155/2015/578541.

16. A hybrid ant algorithm for the set covering problem / B. Crawford, R. Soto, F. Monfroy, F. Paredes, W. Palma // Intern. J. of Phys. Sci. 2011. Vol. 6, no. 19. P. 4667–4673.

17. Al-Shihabi S., Arafeh M., Barghash M. An improved hybrid algorithm for the set covering problem // Comp. Ind. Engin. 2015. Vol. 85. P. 328–334. doi: 10.1016/j.cie.2015.04.007.

18. Yang X.-Sh. Firefly algorithms for multimodal optimization // Proc. of the 5th Intern. Symp. on Stochastic Algorithms, Foundations and Appl. (SAGA 2009). Springer Berlin, Heidelberg, 2009. Vol. 5792. P. 169–178.

19. Binary firefly algorithm for the set covering problem / B. Crawford, R. Soto, W. Palma, C. Castro, F. Paredes // 2014 9th Iberian Conf. on Information Syst. and Technol. (CISTI). Barcelona, Spain: IEEE, 2014. P. 1–5. doi: 10.1109/CISTI.2014.6877090.

20. Hatamlou A. Black hole: A new heuristic optimization approach for data clustering // Inf. Sci. 2013. Vol. 222. P. 175–184. doi: 10.1016/j.ins.2012.08.023.

21. Solving the set covering problem with a binary black hole inspired algorithm / Á. G. Rubio, B. Crawford, R. Soto, W. Palma, C. Castro, F. Paredes, R. Olivares, C. Valenzuela // Int. Conf. on Computational Sci. and Its Appl. (ICCSA 2016). At: Lecture Notes in Comp. Sci. (LNTCS). Vol. 9786. Springer, Cham, 2016. P. 207–219. doi: 10.1007/978-3-319-42085-1_16.

22. Grasshopper optimization algorithm: Theory, variants, and Appl. / Ya. Meraihi, A. B. Gabis, S. Mirjalili, A. Ramdane-Cherif // IEEE Access. 2021. Vol. 9. P. 50001–50024. doi: 10.1109/ACCESS.2021.3067597.

23. Kennedy J., Eberhart R. Particle swarm optimization // Proc. of the IEEE Int. Conf. on Neural Networks. Perth, WA, Australia: IEEE, 1995. P. 1942–1948. doi: 10.1109/ICNN.1995.488968

24. Eberhart R., Kennedy J. A new optimizer using particle swarm theory // Proc. of the Sixth Int. Symp. on Micro Machine and Human Sci. Nagoya, Japan: IEEE, 1995. P. 39–43. doi: 10.1109/MHS.1995.494215.

25. Казакова Е. М. Краткий обзор методов оптимизации на основе роя частиц // Вестн. КРАУНЦ. Физ.-мат. науки. 2022. Т. 39, № 2. С. 150–174.

26. Yang X.-S. A new metaheuristic bat-inspired algorithm // Nature Inspired Cooperative Strategies for Optimization (NISCO 2010) / Eds. J. R. Gonzalez. Berlin, Heidelberg: Springer, 2010. Vol. 284. P. 65–74. doi: 10.1007/978-3-642-12538-6_6.

27. Gandomi A. H., Alavi A. H. Krill herd: A new bio-inspired optimization algorithm // Commun. Nonlinear Sci. Numer. Simulat. 2012. Vol. 17, no. 12. P. 4831–4845. doi: 10.1016/j.cnsns.2012.05.010.

28. Mirjalili S., Mirjalili S. M., Lewis A. Grey wolf optimizer // Advances in Engin. Software. 2014. Vol. 69. P. 46–61. doi: 10.1016/j.advengsoft.2013.12.007.

29. Mirjalili S. Moth-Flame optimization algorithm: A novel nature-inspired heuristic paradigm // Knowledge-Based Syst. 2016. Vol. 89. P. 228–249. doi: 10.1016/j.knosys.2015.07.006.

30. Mirjalili S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems // Neural Comp. & Appl. 2016. Vol. 27. P. 1053–1073. doi: 10.1007/s00521-015-1920-1.

31. Mirjalili S., Lewis A. The whale optimization algorithm // Adv. in Eng. Software. 2016. Vol. 95. P. 51–67. doi: 10.1016/j.advengsoft.2016.01.008.

32. Rashedi E., Nezamabadi-Pour H., Saryazdi S. GSA: A Gravitational Search Algorithm // Inf. Sci. 2009. Vol. 179, no. 13. P. 2232–2248. doi: 10.1016/J.INS.2009.03.004.

33. Mirjalili S. Z., Mirjalili S. M., Hatamlou A. Multi-verse optimizer: A nature-inspired algorithm for global optimization // Neural Comp. and Appl. 2015. Vol. 27, no. 2. P. 495–513. doi: 10.1007/s00521-015-1870-7.

34. Pashaei E., Aydin N. Binary black hole algorithm for feature selection and classification on biological data // Appl. Soft Comp. 2017. Vol. 56. P. 94–106. doi: 10.1016/j.asoc.2017.03.002.

35. Holland J. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, 1975. 206 p.

36. Goldberg D. E. Genetic algorithms in search, optimization and machine learning. Addison-Wesley, 1989. P. 432. doi: 10.5860/CHOICE.27-0936.

37. Панченко Т. В. Генетические алгоритмы. Астрахань: Астраханский гос. ун-т, 2007. 88 с.

38. Reeves C., Rowe J. E. Genetic algorithms: Principles and perspectives: A guide to GA theory. Springer, 2002. 655 p.

39. Cramer N. L. A representation for the adaptive generation of simple sequential programs // Proc. of Intern. Conf. on Genetic Algorithms and Their Appl. / Ed. J. J. Grefenstette. Pittsburgh, PA: Carnegie-Mellon University 1985,). 1985. P. 183–187.

40. Atashpaz-Gargari E., Lucas C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition // 2007 IEEE Congress on Evolutionary Computation. Singapore: IEEE, 2007. P. 4661–4667. doi: 10.1109/CEC.2007.4425083.

41. Rao R. V., Savsani V., Vakharia D. P. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems // Comp.-Aided Design. 2011. Vol. 43, no. 3. P. 303–315. doi: 10.1016/j.cad.2010.12.015.

42. Лопатин А. С. Метод отжига // Стохастическая оптимизация в информатике. 2005. Вып. 1. С. 133–149.

43. Glover F., Taillard E. A user's guide to tabu search // Annals of operations research. 1993. Vol. 41, no. 1. P. 1–28. doi: 10.1007/BF02078647.

44. Voudouris C., Tsang E. P. K., Alsheddy A. Guided local search // Wiley Encycl. of Operations Research and Management Sci. 2011. 41 p. doi: 10.1002/9780470400531.eorms0369.

45. Resende M. G. C., Ribeiro C. C. GRASP: Greedy randomized adaptive search procedures // Handbook of Metaheuristics. Boston. Kluwer Academic Publishers, 2003. P. 287–320.

Информация об авторах

Васькин Павел Ильич – канд. техн. наук, доцент, директор по направлению ИС и БД, ЗАО «Морские компьютерные системы», ул. Бабушкина, 80, Санкт-Петербург, 192174, Россия.
E-mail: vpi@mcs.ru

Лисс Анна Александровна – канд. техн. наук, доцент, заведующий кафедрой МОЭВМ СПбГЭТУ «ЛЭТИ».
E-mail: aaliss@etu.ru

References

1. Karp R. M. Reducibility among combinatorial problems // Complexity of Comp. Computations. Boston, MA: Springer, 1972. P. 85–103. doi: 10.1007/978-1-4684-2001-2_9.

2. Konovalov I. S., Ostapenko S. S., Kobak V. G. Sravnenie jeffektivnosti raboty tochnyh i priblizhennyh algoritmov dlja reshenija zadachi o pokrytii mnozhestva // Vestn. DonGTU. 2017. T. 17. № 3(90). S. 137–144. doi: 10.23947/1992-5980-2017-17-3-137-144. (In Russ.).

3. Yang X.-Sh. Nature-inspired metaheuristic algorithms. 2nd ed. Frome, UK: Luniver press, 2010. 160 p.

4. Obzor metajevristicheskih metodov optimizacii, primenjaemyh pri reshenii jelektrojenergeticheskih zadach / R. A. Alehin, Ju. P. Kubar'kov, D. V. Zakamov, D. V. Umjarov // Vestn. SamGTU. Ser.: Tehnicheskie nauki. 2019. № 3(63). S. 6–19. (In Russ.).

5. Timofeeva O. P., Neimushhev S. A., Neimushheva L. I. Issledovanie populjacionnyh algoritmov v reshenii zadach nepreryvnoj optimizacii // Tr. NGTU im. R. E. Alekseeva. 2018. № 4(123). S. 48–55. (In Russ.).

6. Karpenko A. P. Osnovnye sushhnosti populjacionnyh algoritmov global'noj optimizacii. Opyt sistematizacii // Internet-zhurn. «Naukovedenie». 2017. T. 9. № 6(43). S. 41–67. (In Russ.).

7. Adaptive black hole algorithm for solving the set covering problem / R. Soto, B. Crawford, R. Olivares, C. Taramasco, I. Figueroa, A. Gómez, C. Castro, F. Paredes // Math. Problems in Engin. 2018. Vol. 2018. Art. 2183214. P. 1–23. doi: 10.1155/2018/2183214.

8. Comparing cuckoo search, bee colony, firefly optimization, and electromagnetism like algorithms for solving the set covering problem / R. Soto, B. Crawford, C. Galleguillos, J. Barraza, S. Lizama, A. Muñoz, J. Vilches, S. Misra, F. Paredes // Intern. Conf. on Comp. Sci. and Its Appl. (ICCSA 2015). Springer, Cham, 2015. At: Lecture Notes in Comp. Sci. (LNTCS). Vol. 9155. P. 187–202. doi: 10.1007/978-3-319-21404-7_14.

9. Balas E., Carrera M. C. A dynamic subgradient-based branch-and-bound procedure for set covering // Operations Research. 1996. Vol. 44. № 6. P. 875–890. doi: 10.1287/opre.44.6.875.

10. Akhter F. A heuristic approach for minimum set cover problem // Int. J. Adv. Res. Artif. Intell (IJARAI). 2015. Vol. 4, no. 6. P. 40–45. doi: 10.14569/IJARAI.2015.040607.

11. Eremeev A. V. Geneticheskij algoritm dlja reshenija zadachi o pokrytii // Diskretnyj analiz i issledovanie operacij. Ser. 2. 2000. T. 7, № 1. S. 47–60. (In Russ.).

12. Beasley J. E., Chu P. C. A genetic algorithm for the set covering problem // Eur. J. of Operational Research. 1996. Vol. 94, no. 2. P. 392–404.

13. Tjuhin M. V., Lomazov V. A. Primenenie bioinspirirovannyh algoritmov roevogo intellekta dlja reshenija zadach optimizacii // Vopr. ustojchivogo razvitija obshhestva. 2021. № 6. S. 811–815. (In Russ.).

14. Application of the artificial bee colony algorithm for solving the set covering problem / B. Crawford, R. Soto, R. Cuest, F. Paredes // The Sci. World J. 2014. Vol. 2014. Art. 189164. P. 1–8. doi: 10.1155/2014/189164.

15. A binary cat swarm optimization algorithm for the non-unicost set covering problem / B. Crawford, R. Soto, N. Berríos, F. Johnson, F. Paredes, C. Castro, E. Norero // Math. Problems in Engin. 2015. Vol. 2015. Art. 578541. P. 1–8. doi: 10.1155/2015/578541.

16. A hybrid ant algorithm for the set covering problem / B. Crawford, R. Soto, F. Monfroy, F. Paredes, W. Palma // Intern. J. of Phys. Sci. 2011. Vol. 6, no. 19. P. 4667–4673.

17. Al-Shihabi S., Arafeh M., Barghash M. An improved hybrid algorithm for the set covering problem // Comp. Ind. Engin. 2015. Vol. 85. P. 328–334. doi: 10.1016/j.cie.2015.04.007.

18. Yang X.-Sh. Firefly algorithms for multimodal optimization // Proc. of the 5th Intern. Symp. on Stochastic Algorithms, Foundations and Appl. (SAGA 2009). Springer Berlin, Heidelberg, 2009. Vol. 5792. P. 169–178.

19. Binary firefly algorithm for the set covering problem / B. Crawford, R. Soto, W. Palma, C. Castro, F. Paredes // 2014 9th Iberian Conf. on Information Syst. and Technol. (CISTI). Barcelona, Spain: IEEE, 2014. P. 1–5. doi: 10.1109/CISTI.2014.6877090.

20. Hatamlou A. Black hole: A new heuristic optimization approach for data clustering // Inf. Sci. 2013. Vol. 222. P. 175–184. doi: 10.1016/j.ins.2012.08.023.

21. Solving the set covering problem with a binary black hole inspired algorithm / Á. G. Rubio, B. Crawford, R. Soto, W. Palma, C. Castro, F. Paredes, R. Olivares, C. Valenzuela // Int. Conf. on Computational Sci. and Its Appl. (ICCSA 2016). At: Lecture Notes in Comp. Sci. (LNTCS). Vol. 9786. Springer, Cham, 2016. P. 207–219. doi: 10.1007/978-3-319-42085-1_16.

22. Grasshopper optimization algorithm: Theory, variants, and Appl. / Ya. Meraihi, A. B. Gabis, S. Mirjalili, A. Ramdane-Cherif // IEEE Access. 2021. Vol. 9. P. 50001–50024. doi: 10.1109/ACCESS.2021.3067597.

23. Kennedy J., Eberhart R. Particle swarm optimization // Proc. of the IEEE Int. Conf. on Neural Networks. Perth, WA, Australia: IEEE, 1995. P. 1942–1948. doi: 10.1109/ICNN.1995.488968

24. Eberhart R., Kennedy J. A new optimizer using particle swarm theory // Proc. of the Sixth Int. Symp. on Micro Machine and Human Sci. Nagoya, Japan: IEEE, 1995. P. 39–43. doi: 10.1109/MHS.1995.494215.

25. Kazakova E. M. Kratkij obzor metodov optimizacii na osnove roja chastic // Vestn. KRAUNC. Fiz.-mat. nauki. 2022. T. 39, № 2. C. 150–174. (In Russ.).

26. Yang X.-S. A new metaheuristic bat-inspired algorithm // Nature Inspired Cooperative Strategies for Optimization (NISCO 2010) / Eds. J. R. Gonzalez. Berlin, Heidelberg: Springer, 2010. Vol. 284. P. 65–74. doi: 10.1007/978-3-642-12538-6_6.

27. Gandomi A. H., Alavi A. H. Krill herd: A new bio-inspired optimization algorithm // Commun. Nonlinear Sci. Numer. Simulat. 2012. Vol. 17, no. 12. P. 4831–4845. doi: 10.1016/j.cnsns.2012.05.010.

28. Mirjalili S., Mirjalili S. M., Lewis A. Grey wolf optimizer // Advances in Engin. Software. 2014. Vol. 69. P. 46–61. doi: 10.1016/j.advengsoft.2013.12.007.

29. Mirjalili S. Moth-Flame optimization algorithm: A novel nature-inspired heuristic paradigm // Knowledge-Based Syst. 2016. Vol. 89. P. 228–249. doi: 10.1016/j.knosys.2015.07.006.

30. Mirjalili S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems // Neural Comp. & Appl. 2016. Vol. 27. P. 1053–1073. doi: 10.1007/s00521-015-1920-1.

31. Mirjalili S., Lewis A. The whale optimization algorithm // Adv. in Eng. Software. 2016. Vol. 95. P. 51–67. doi: 10.1016/j.advengsoft.2016.01.008.

32. Rashedi E., Nezamabadi-Pour H., Saryazdi S. GSA: A Gravitational Search Algorithm // Inf. Sci. 2009. Vol. 179, no. 13. P. 2232–2248. doi: 10.1016/J.INS.2009.03.004.

33. Mirjalili S. Z., Mirjalili S. M., Hatamlou A. Multi-verse optimizer: A nature-inspired algorithm for global optimization // Neural Comp. and Appl. 2015. Vol. 27, no. 2. P. 495–513. doi: 10.1007/s00521-015-1870-7.

34. Pashaei E., Aydin N. Binary black hole algorithm for feature selection and classification on biological data // Appl. Soft Comp. 2017. Vol. 56. P. 94–106. doi: 10.1016/j.asoc.2017.03.002.

35. Holland J. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, 1975. 206 p.

36. Goldberg D. E. Genetic algorithms in search, optimization and machine learning. Addison-Wesley, 1989. P. 432. doi: 10.5860/CHOICE.27-0936.

37. Panchenko T. V. Geneticheskie algoritmy. Astrahan': Astrahanskij gos. un-t, 2007. 88 s. (In Russ.).

38. Reeves C., Rowe J. E. Genetic algorithms: Principles and perspectives: A guide to GA Theory. Springer, 2002. 655 p. (In Russ.).

39. Cramer N. L. A representation for the adaptive generation of simple sequential programs // Proc. of Intern. Conf. on Genetic Algorithms and Their Appl. / Ed. J. J. Grefenstette. Pittsburgh, PA: Carnegie-Mellon University 1985,). 1985. P. 183–187.

40. Atashpaz-Gargari E., Lucas C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition // 2007 IEEE Congress on Evolutionary Computation. Singapore: IEEE, 2007. P. 4661–4667. doi: 10.1109/CEC.2007.4425083.

41. Rao R. V., Savsani V., Vakharia D. P. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems // Comp.-Aided Design. 2011. Vol. 43, no. 3. P. 303–315. doi: 10.1016/j.cad.2010.12.015.

42. Lopatin A. S. Metod otzhiga // Stohasticheskaja optimizacija v informatike. 2005. Vyp. 1. S. 133–149. (In Russ.).

43. Glover F., Taillard E. A user's guide to tabu search // Annals of operations research. 1993. Vol. 41, no. 1. R. 1–28. doi: 10.1007/BF02078647.

44. Voudouris C., Tsang E. P. K., Alsheddy A. Guided local search // Wiley Encyclopedia of Operations Research and Management Sci. 2011. 41 p. doi: 10.1002/9780470400531.eorms0369.

45. Resende M. G. C., Ribeiro C. C. GRASP: Greedy randomized adaptive search procedures // Handbook of Metaheuristics. Boston. Kluwer Academic Publishers, 2003. P. 287–320.

Information about the authors

Pavel I. Vaskin – Cand. Sci. (Eng.), Associate Professor, Director of Information Technology and Database, Marine Computer Systems CJSC, 80 Babushkina St., Saint Petersburg, Russia.
E-mail: vpi@mcs.ru

Anna A. Liss – Cand. Sci. (Eng.), Associate Professor, Head of the Computer Software and Application Department, Saint Petersburg Electrotechnical University.
E-mail: aaliss@etu.ru

Статья поступила в редакцию 05.11.2025; принята к публикации после рецензирования 20.01.2026; опубликована онлайн 30.03.2026.

Submitted 05.11.2025; accepted 20.01.2026; published online 30.03.2026.