УДК 519.854.2 + 519.83
ББК 22.18

ДИНАМИЧЕСКАЯ АДАПТАЦИЯ ГЕНЕТИЧЕСКОГО АЛГОРИТМА МАРШРУТИЗАЦИИ ТРАНСПОРТА НА БОЛЬШИХ СЕТЯХ

Захаров В. В.[1], **Мугайских А. В.**[2]
(*Санкт-Петербургский государственный университет, Санкт-Петербург*)

Описывается процедура динамической адаптации генетического алгоритма для тестовых задач коммивояжёра на больших сетях, позволяющая получать более экономичные маршруты за то же время вычислений. Эффективность предложенной процедуры подтверждается результатами вычислительных экспериментов получения численного решения набора тестовых задач из библиотеки TSPLib и устойчивого уменьшения средней длины генерируемых решений по сравнению с решениями, предоставляемыми исходной эвристикой. Тем самым демонстрируется методика использования свойства временной несостоятельности эвристических алгоритмов для целей повышений уровня их эффективности. Оценка временной состоятельности эвристического алгоритма проводится с помощью нового критерия – экспериментального уровня временной состоятельности.

Ключевые слова: временная состоятельность, генетический алгоритм, задачи маршрутизации.

Введение

Одной из важнейших задач транспортной логистики является оптимизация затрат, которая позволяет при том же объё-

[1] *Виктор Васильевич Захаров, доктор физико-математических наук, профессор* (*v.zaharov@spbu.ru*).
[2] *Александр Всеволодович Мугайских, аспирант* (*alexander.mugaiskih@gmail.com*).

ме используемых ресурсов увеличить прибыль. Для транспортных компаний существенная минимизация затрат на транспортировку может быть достигнута за счёт построения эффективных маршрутных планов транспортных средств. Особенно существенный эффект снижения затрат может быть достигнут на больших транспортных сетях. Именно по этой причине эффективным алгоритмам решения задачи маршрутизации транспорта, позволяющим генерировать менее затратные маршруты, уделяется пристальное внимание со стороны исследователей. Одним из важных объектов применения математических методов в транспортной логистике является задача маршрутизации транспортных средств. Классическим примером задачи маршрутизации является задача коммивояжёра (англ. travelling salesman problem), которую мы будем подробно рассматривать.

Основной мотивацией написания данной статьи являлась проверка гипотезы о временной несостоятельности (динамической неустойчивости) эвристических алгоритмов маршрутизации и разработка метода повышения уровня динамической устойчивости и эффективности используемых алгоритмов на примере генетического алгоритма. Статья структурирована следующим образом. Кроме введения, она включает семь разделов. Первый посвящен описанию задачи коммивояжёра и обзору методов ее решения. Во втором разделе приводятся основные понятия и термины генетического алгоритма, используемые в тексте. Третий раздел включает в себя описание базовой схемы генетического алгоритма и одного из наиболее эффективных инструментов построения решения – кроссовера Греффенстетта. В четвертом разделе обсуждается проблема динамической неустойчивости (временной несостоятельности) эвристических алгоритмов маршрутизации и новый критерий – экспериментальный уровень временной состоятельности алгоритма. Пятый раздел посвящен описанию процедуры динамической адаптации эвристических алгоритмов транспортной маршрутизации применительно к генетическому алгоритму. В разделе 6 приводятся результаты численных экспериментов по оценке уровня временной состоятельности ге-

нетического алгоритма и сравнительный анализ результатов применения генетического алгоритма с кроссовером Греффенстетта и алгоритма его динамической адаптации при решении ряда тестовых задач из широко известной среди специалистов библиотеки TSPLib. Седьмой раздел содержит краткие выводы и планы дальнейших исследований.

1. Задача коммивояжёра (TSP) и методы её решения

1.1. Постановка задачи коммивояжёра

В классической постановке задача коммивояжёра заключается в поиске самого выгодного маршрута, проходящего через все указанные в условиях задачи вершины хотя бы по одному разу с возвратом в исходную вершину. Задача коммивояжёра может быть представлена в виде математической модели – графа $G = (V, E)$. Множество вершин $V = \{v_1, v_2, \ldots, v_n\}$ в графе соответствуют городам в задаче коммивояжёра, множество рёбер $E = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ – коммуникациям, соединяющим пары городов. Каждому ребру из множества E можно сопоставить критерий выгодности маршрута.

В зависимости от постановки задачи вес ребра (v_i, v_j) графа G определяет время перехода, стоимость поездки или расстояние между вершинами v_i и v_j соответственно. Задача коммивояжёра может быть сформулирована как задача поиска на графе G гамильтонова контура с наименьшим общим весом.

Пусть $I = \{1, \ldots, n\}$ – множество индексов вершин из тестовой задачи. Определим целевую функцию f как суммарную длину маршрута, включающего в себя все вершины рассматриваемой задачи.

Параметрами задачи являются элементы матрицы расстояний $C = \{c_{ij}\}$, $i, j \in I$. В случае асимметричной матрицы C задача коммивояжера моделируется ориентированным графом. В симметричном случае количество возможных маршрутов вдвое меньше асимметричного случая и верно равенство $c_{ij} = c_{ji}$. Далее будем рассматривать задачи только с симметричной матрицей расстояний, в которых длина маршрута между двумя вер-

шинами v_i и v_j определяется как евклидово расстояние: $c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, где (x_i, y_i),(x_j, y_j) – координаты вершин v_i, v_j соответственно.

Переменными задачи являются элементы бинарной матрицы переходов между вершинами $Z = \{z_{ij}\}$, $i, j \in I$, которые равны 1, если в построенном маршруте для тестовой задачи присутствует ребро (v_i, v_j), 0 – иначе.

Таким образом, задача коммивояжёра может быть сформулирована и решена в рамках линейного целочисленного программирования [22, 40, 41]. Требуется найти

$$f = \sum_{i \in I} \sum_{j \neq i, j \in J} c_{ij} z_{ij} \to \min$$

при следующем наборе ограничений:

$$0 \leqslant z_{ij} \leqslant 1, z_{ij} \in \mathbb{Z}$$

$$\sum_{j \in I, j \neq i} z_{ij} = 1, \forall i \in I, \tag{1}$$

$$\sum_{i \in I, i \neq j} z_{ij} = 1, \forall j \in I, \tag{2}$$

$$\sum_{i \in S, j \in S} z_{ij} = |S| - 1 \quad (S \subset V, |S| > 1). \tag{3}$$

Ограничения (1) гарантируют, что в каждую вершину можно попасть по одному ребру, исходящему из другой вершины. Аналогично (2) показывают, что из каждой вершины существует только один путь, ведущий к другой вершине. Группа неравенств (3) обеспечивает связность маршрута обхода городов в допустимом решении, т.е. он не может состоять из двух и более несвязных частей.

Задача коммивояжёра занимает центральное место среди задач комбинаторной оптимизации [6, 10, 12]. Имеет широкое практическое применение в различных областях: транспортная логистика [15], робототехника [19, 21], теория расписаний. Примерами данной задачи могут служить построение маршрутных планов для парка машин логистической компании; наблюдение группы роботов за объектами [44]; размещение датчиков в сенсорной

сети [48]; поставка запчастей на производстве [39]; планирование авиаперелётов [25]. Несмотря на кажущуюся простоту постановки задачи коммивояжёра, ей по-прежнему уделяется пристальное внимание со стороны специалистов при рассмотрении задачи в новых формулировках и при различных ограничениях [2, 4, 13, 14].

1.2. Алгоритмы решения задачи коммивояжёра

Все методы решения задачи TSP, как правило, относят к одной из двух категорий: точные и эвристические алгоритмы [7]. Среди точных алгоритмов решения данной задачи выделим метод полного перебора, метод ветвей и границ, метод ветвей и отсечений и метод динамического программирования.

Метод ветвей и границ и метод ветвей и отсечений являются развитием алгоритма полного перебора. Основная идея заключается в проверке критерия ограничивающей функции, по которому можно приостановить построение ветви дерева перестановок на определенном уровне. Известные модификации данных подходов позволяют решать задачу коммивояжёра с несколькими сотнями вершин, однако они требует высокой вычислительной мощности компьютера.

Идея динамического программирования, предложенная Р. Беллманом [17], М. Хелдом и Р. Карпом [32], применима к задаче TSP и заключается в многошаговом процессе принятия решений, на каждой шаге которого необходимо определить функцию Беллмана и найти оптимальный маршрут обхода оставшихся вершин. Число операций, требуемых для вычисления итогового маршрута, растет экспоненциально по мере увеличения количества вершин тестовой задачи.

Ввиду NP-сложности задачи TSP точные методы не всегда могут эффективно применяться для задач большой размерности. По этой причине используют эвристические методы, которые генерируют решения, близкие к оптимальному, но за приемлемое по сравнению с точными алгоритмами время. Экспериментальный анализ эвристических алгоритмов решения задачи TSP и её подклассов проведен в работе [30].

По способу формирования итогового маршрута эвристические алгоритмы можно разделить на два класса: конструктивные (tour construction heuristics) и итерационные эвристики (tour improvement heuristics).

Конструктивные эвристики генерируют один уникальный маршрут для каждой тестовой задачи без последующего его улучшения [37]. К данному классу относят алгоритм ближайшего соседа [36], жадный алгоритм [31], эвристики вставок [27], алгоритм Кристофидеса [20]. Перечисленные алгоритмы последовательно строят допустимое решение, добавляя в него вершины, пока полный маршрут не будет сформирован. Часть решения, построенная к текущему моменту, останется неизменной до конца работы алгоритма. Как правило, с помощью данных эвристик можно получить решения, уступающие в длине оптимальному около 10–15%.

Другой класс эвристических алгоритмов решения задачи TSP – итерационные эвристики – начинает работать с уже готовым маршрутом, построенным одним из конструктивных методов, последовательно улучшая его на каждой итерации. Существует несколько способов сделать это: широко используемыми являются операции 2-замена и 3-замена [3], интегрированные в алгоритмы локального поиска и алгоритм Лина–Кернигана [33, 34].

Среди итерационных методов можно выделить отдельный класс алгоритмов, который носит название метаэвристик (metaheuristics) [18, 26]. Это достаточно общие итерационные процедуры, использующие рандомизацию и элементы самообучения, интенсификацию и диверсификацию поиска, адаптивные механизмы управления, а также конструктивные эвристики и методы локального поиска. Метаэвристики принято делить на траекторные методы, когда каждой итерации соответствует одно допустимое решение, и на алгоритмы, которые работают сразу с популяцией решений. К первой группе относят методы имитации отжига (simulated annealing) [5, 11, 28], поиск с запретами (tabu search) [49], поиск с чередующимися окрестностя-

ми (variable neighbourhood search). Ко второй – генетические алгоритмы (genetic algorithms), муравьиные колонии (ant colony optimization) [23], вероятностные жадные алгоритмы (greedy randomized adaptive search procedure) и др. [47].

2. Термины и понятия

Далее в статье используются следующие термины и понятия.

Эвристический алгоритм – алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он даёт достаточно хорошее решение в большинстве случаев.

Генетический алгоритм – эвристический алгоритм для решения задач оптимизации путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе, генерирующий различные решения при каждом запуске в виду механизмов рандомизации на основных этапах метода.

Особь – допустимое решение задачи в генетическом алгоритме.

Популяция особей – конечный набор допустимых решений задачи, с которым впоследствии будет работать генетический алгоритм.

Хромосома – совокупность параметров-генов, описывающих особь. Применительно к задаче коммивояжёра в хромосоме особи записан порядок обхода вершин тестовой задачи, соответствующий одному допустимому решению; гены в хромосоме соответствуют вершинам тестовой задачи.

Ген – это атомарный элемент хромосомы.

Скрещивание, кроссовер – основной этап генетического алгоритма, в котором принимают участие две особи одной популяции – родительские особи. Путем комбинирования и наследования особенностей обоих родительских особей (далее – родителей) генерируются потомственные особи (далее – потомки), происходит улучшение качественного признака одной потомственной особи в частности и всей популяции в целом.

Мутация – один из этапов генетического алгоритма, в котором принимает участие одна особь из популяции, проводится с целью восстановления особей, выпавших из популяции, и формирования генов, которые не были представлены в исходной популяции.

Селекция – заключительный этап генетического алгоритма при формировании новой популяции особей, заключается в отборе наиболее приспособленных особей для их включение в дальнейший процесс эволюции.

3. Генетический алгоритм и кроссовер Греффенстетта

Генетический алгоритм – метаэвристический алгоритм, в основе которого лежат операции, свойственные естественному отбору, происходящему в природе. Впервые был представлен в 1975 г. Джоном Холландом как одно из направлений эволюционных алгоритмов [35]. Не детализируя описание генетического алгоритма, коротко приведем схему классического генетического алгоритма, которая состоит из следующих этапов:

1. Инициализация начальной популяции, s особей.
2. **Пока** не выполнен критерий останова, **выполнять:**
3. Производится s/2 операций скрещивания между 2 случайными особями.
4. Случайно выбранные особи из популяции мутируют.
5. Осуществляется оценка функции приспособленности и селекция.

Решением задачи коммивояжёра является маршрут, соответствующий хромосоме самой приспособленной особи популяции последнего поколения. Генетический алгоритм является универсальной эвристической процедурой, которая может быть имплементирована в программный комплекс для решения задач транспортной маршрутизации в различных постановках таких как TSPTW (travelling salesman problem with time windows), IRP (inventory routing problem), VRP (vehicle routing problem), MDVRP (multi-depot vehicle routing problem), PDP (pickup and

delivery problem) и др. Кроме того, генетическая эвристика способна решать как саму задачу, так и её подзадачи, что является необходимым условием для проведения процедуры динамической адаптации алгоритма.

Как известно, основным этапом генетического алгоритма, о котором речь пойдёт в разделе 5, является операция скрещивания. Предложенные в литературе методы скрещивания являются допустимыми для задачи коммивояжёра и дают в целом приемлемые результаты, однако они не используют информацию о расстоянии между вершинами тестовых задач TSP. Разрешил эту проблему Джон Греффенстетт, предложив в своей работе [29] новый вид скрещивания, который использует информацию о расстоянии между вершинами.

В применении к задачам TSP с симметричной матрицей переходов с 70 и 100 вершинами тестовой библиотеки TSPLib [43] использование эвристики Греффенстетта на этапе скрещивания генетического алгоритма позволило получить более короткие маршруты в сравнении с частичным скрещиванием (partially mapped crossover), циклическим скрещиванием (cycle crossover), кроссовером перестановки рёбер (edge recombination crossover), двухточечным скрещиванием (two-point crossover), равномерным скрещиванием (uniform order-based crossover), модифицированным одноточечным скрещиванием (shuffle crossover) и кроссовером Ямамура (sub-tour exchange crossover). Подробнее о перечисленных видах скрещивания можно прочитать в работе [38]. Для тестовой задачи с 100 вершинами среднее отклонение длины маршрута, сгенерированного с помощью кроссовера Греффенстетта, от длины точного оптимального решения является наименьшим по сравнению перечисленными кроссоверами и составляет 11,9% [38].

В работе [42] также было произведено сравнение 8 видов кроссовера для решения задачи VRP, которая является обобщением задачи TSP на случай нескольких транспортных средств. Были описаны следующие виды скрещивания: упорядоченное скрещивание (order crossover), частичное скрещивание, кроссовер пере-

становки рёбер, циклическое скрещивание, кроссовер альтернативных рёбер (alternating edges crossover), эвристический кроссовер Греффенстетта и две его модификации [42]. Лучшие результаты среди рассмотренных алгоритмов скрещивания на тестовых задачах VRP стандартной библиотеки Кристофидеса [24] показал эвристический кроссовер Греффенстетта и две его модификации.

Таким образом, рассмотренный оператор скрещивания генетического алгоритма превосходит другие альтернативы в генерации маршрутов меньшей длины, и он был имплементирован в программную реализацию генетического алгоритма решения задачи коммивояжёра для дальнейших исследований, результаты которых представлены в следующих разделах статьи.

Ниже приведен классический алгоритм кроссовера Греффенстетта:

1. Случайную вершину v необходимо записать в первый ген потомка

2. Пока маршрут полностью не сформирован, **выполнять:**

3. Сравнить два ребра, выходящие из вершины v у родителей, выбрать самое короткое. Вторую вершину данного ребра обозначить через v_{min}.

4. **Если** вершина v_{min} уже присутствует в хромосоме потомка, **то** за v_{min} принять случайную вершину, не принадлежащую потомку.

5. Записать v_{min} в хромосому потомственной особи, $v = v_{min}$.

4. Динамическая неустойчивость эвристических алгоритмов маршрутизации

В работе [1] авторы отмечают, что процесс поиска оптимального решения в генетических алгоритмах направляется исключительно полученными значениями целевой функции в предыдущих точках пространства решений. При этом никак не используются предположения о таких свойствах целевой функции и ограничений как выпуклость, дифференцируемость, выполнение соотношения оптимальности Беллмана.

Сформулированный в 1957 г. принцип оптимальности Беллмана применительно к задачам маршрутизации утверждает, что любая часть оптимального маршрутного плана сохраняет свойство оптимальности в каждой подзадаче, рассматриваемой вдоль сгенерированной траектории движения, в процессе реализации начального маршрута [16]. Однако данное свойство не выполняется для решений задач маршрутизации, полученных с использованием большинства итерационных эвристик. Конструктивные эвристики почти всегда генерируют динамически устойчивые маршруты, однако не обеспечиваю достаточный уровень эффективности по сравнению с итерационными.

Используя принципы динамической устойчивости решений, описанные в [9], введём следующие обозначения:

Пусть P – множество тестовых задач, $S(p)$ – конечное множество решений, сгенерированных эвристическим алгоритмом для тестовой задачи $p \in P$. Каждому решению $s(p)$ соответствует маршрут, представляющий собой последовательность обхода вершин тестовой задачи. Рассмотрим последовательность обхода вершин одного маршрута и разобьём её на T частей так, что количество вершин, которые были пройдены за первые t частей маршрута, вычисляется по формуле $n(t, s(p)) = \lfloor n_0 t/T \rfloor$, где n_0 – количество вершин в тестовой задаче p, а T – параметр алгоритма, определяемый в начале эксперимента. Таким образом, каждая часть маршрута (кроме последней) содержит одинаковое число вершин. Под периодом t, где $t = 0, 1, \ldots, T-1$, будем понимать промежуток времени в пути, которому соответствует часть t маршрута исходного решения $s(p)$.

Пусть $s^+(t, p)$ – часть маршрута $s(p)$, соответствующая порядку обхода городов после периода t. Часть $s^-(t, p)$ исходного решения включает города маршрута $s(p)$, посещённые за периоды $\tau = 0, 1, \ldots, t$. Тогда каждое решение можно представить в виде объединения двух частей $s(p) = s^-(t, p) \cup s^+(t, p)$. При этом верно следующее: $s^+(0, p) = s(p)$.

Рассмотрим подзадачу $p(s^-(t, p))$, которая отличается от начальной тестовой задачи p тем, что множество городов сокра-

щено за счёт исключения тех из них, которые входят в часть маршрута $s^-(t,p)$. Точка отправления в подзадаче $p(s^-(t,p))$ находится в последней вершине маршрута $s^-(t,p)$. Обозначим через $s(p(s^-(t,p)))$ некоторое решение подзадачи $p(s^-(t,p))$, генерируемое эвристическим алгоритмом.

Определение 1. *Маршрут (решение) $s(p)$, сгенерированный определенной эвристикой, будем называть состоятельным во времени (динамически устойчивым) относительно этой эвристики, если для каждого $t = 1, \ldots, T-1$ и любого $s(p(s^-(t,p)))$, сгенерированного той же эвристикой, верно неравенство*

$$f(s^+(t,p)) \leqslant f(s(p(s^-(t,p)))), \tag{4}$$

где f – минимизируемая целевая функция для данной задачи маршрутизации транспорта.

Определение 2. *Маршрут (решение) $s(p)$, сгенерированный некоторой эвристикой, будем называть несостоятельным во времени (динамически неустойчивым) относительно этой эвристики, если хотя бы для одного $t^{'}$ найдется сгенерированное этой же эвристикой решение $s(p(s^-(t^{'},p)))$, для которого выполняется неравенство*

$$f(s^+(t,p)) > f(s(p(s^-(t^{'},p)))). \tag{5}$$

Свойство временной состоятельности решения в применении к задачам транспортной маршрутизации означает, что при движении согласно начальному маршрутному плану в подзадачах, генерируемых по методу описанному выше, не будет найдено решение выгоднее текущего. И наоборот, временная несостоятельность решения означает, что хотя бы в одном из периодов будет сгенерировано решение выгоднее текущего.

Опишем процедуру оценки уровня временной состоятельности эвристического алгоритма H для конкретного класса задач транспортной маршрутизации.

1. Рассмотрим множество P тестовых задач из данного класса задач транспортной маршрутизации.

2. Для каждой тестовой задачи $p \in P$ сформируем множество N различных решений, полученных эвристическим алгорит-

мом H. Решения будут различны, так как операторы скрещивания, мутации используют рандомизацию в своих алгоритмах.

3. Для каждого решения проведём M экспериментов его проверки на временную состоятельность. Параметр M служит для получения средней оценки показателя временной состоятельности решения. Обычно полагаем $M = 5$. Начиная с $t = 1$ сформулируем текущую задачу $p(s^-(t,p))$, найдём её решение $s(p(s^-(t,p)))$ с помощью алгоритма H. Далее проверим начальное решение $s(p)$ на состоятельность после первого периода , т.е. справедливость неравенства (4). Если неравенство выполнено, то переходим к следующему периоду. Иначе фиксируем номер периода, на котором свойство временной состоятельности нарушилось и запускаем следующий тест.

Количество экспериментов, в которых начальное решение $s(p) \in N$ перестало быть состоятельным во времени после периода t обозначим через $b(s(p),t)$. Из принципа оптимальности Беллмана следует, что для оптимального решения тестовой задачи, т.е. полученного точными методами, выполнено условие (5).

$$\sum_{t=1}^{T-1} b(s(p),t) = 0. \tag{6}$$

Определение 3. *Экспериментальным уровнем временной состоятельности эвристического алгоритма H будем называть величину, определяемую по следующей формуле:*

$$conH = 1 - \frac{1}{M|P|} \sum_{p \in P} \frac{1}{|N|} \sum_{s(p) \in N} \sum_{t}^{T-1} b(s(p),t). \tag{7}$$

Уровень временной состоятельности для эвристического алгоритма может меняться в зависимости от тестовых задач, на которых он был рассчитан, однако можно заметить, что $0 \leqslant conH \leqslant 1$. Чем ближе значение величины $conH$ к 0, тем чаще решение полученное данной эвристикой теряет оптимальность в ходе своей реализации. Для маршрутов, сгенерированных точными методами, значение данной оценки равно 1.

5. Динамическая адаптация генетического алгоритма

Понятие динамической устойчивости решений было широко исследовано в последние годы для задач теории игр. Используя основную идею, изложенную в статьях [50, 51, 52], мы предложим процедуру (алгоритм) динамической адаптации эвристических алгоритмов в применении к различным классам задач транспортной маршрутизации на примере задачи TSP и генетического алгоритма.

На начальном этапе сгенерируем множество N различных решений тестовой задачи генетическим алгоритмом. Из множества N выберем лучшее решение, с минимальным значением целевой функции. Для него проведём M экспериментов проверки на временную состоятельность. Пусть выбранный маршрут является несостоятельным во времени, начиная с периода t, т.е. после периода t будет найдено лучшее решение в текущей подзадаче $p(s^-(t,p))$. Для уменьшения общей длины обхода всех вершин текущий маршрутный план следует изменить согласно новому решению, полученному в рассмотренной подзадаче. Общая схема работы динамической адаптации состоит в следующем [8].

Алгоритм 1 (Динамическая адаптация ГА для TSP)**.**

1. Генерируем множество N решений эвристическим алгоритмом H для $p \in P$.

2. Определим $s_1(p) = \arg\min\limits_{s(p)\in N} f(s(p))$.

3. От $t = 1$ **до** $T - 1$ **выполнять**

4. Сформулируем текущую задачу $p(s_t^-(t,p))$, получим множество N решений.

5. Определим $s_{t+1}^*(p) = \arg\min\limits_{s(p(s_t^-(t,p)))\in N} f(s(p(s_t^-(t,p))))$.

6. Проверим свойство временной состоятельности для $s_t(p)$.

7. **Если** $f(s_t^+(t,p)) > f(s_{t+1}^*(p))$ **, тогда** маршрут следует изменить на $s_{t+1}(p) = s_t^-(t,p) \cup s_{t+1}^*(p)$.

6. Численные эксперименты

6.1. Расчёт экспериментального уровня временной состоятельности генетического алгоритма с эвристическим кроссовером Греффенстетта

Для проведения эксперимента были рассмотрены пять тестовых задач из стандартной библиотеки TSPLib и применён генетический алгоритм с эвристическим кроссовером Греффенстетта их решения.

В вычислительном эксперименте использовался следующий набор параметров: $|P| = 5, M = 5, T = 5, |N| = 20$. Для каждой из тестовых задач было сгенерировано по 20 различных решений. Количество тестов для одного решения $s(p)$ равнялось 5. Исходный маршрут разбивался на пять периодов. Фиксировался номер периода t, после которого решение $s(p)$ теряло свойство временной состоятельности. Данным значениям соответствуют столбцы $b(s,t)$ в таблице 1.

Таблица 1. Оценка временной состоятельности генетического алгоритма для TSP

Тест. задача	Кол-во тестов	$b(s,t)$				Кол-во сост. во врем.
		$t=1$	$t=2$	$t=3$	$t=4$	
att48	100	31	16	26	8	19
eil51	100	64	14	11	3	8
berlin52	100	19	27	4	4	46
st70	100	16	22	4	1	57
eil101	100	15	28	3	2	52
Сумма	500	145	107	48	18	182

Среднее значение экспериментального уровня временной состоятельности генетического алгоритма для решения задачи TSP: $conGA = 0{,}364$.

Данное значение является довольно низким: только треть сгенерированных на начальном этапе решений сохраняет свойство оптимальности в процессе своей реализации. Это означает,

что существуют другие маршруты, которые можно получить динамически с помощью того же эвристического алгоритма, значения целевой функции которых будет меньше, чем у текущих решений.

Вычислим значение уровня временной состоятельности для решений, полученных с помощью процедуры динамической адаптации. Проведём эксперимент по той же схеме и с параметрами, определёнными ранее. Результаты экспериментов приведены в таблице 2.

Таблица 2. Оценка временной состоятельности динамической адаптации генетического алгоритма для решения задачи TSP

Тест. задача	Кол-во тестов	$b(s,t)$				Кол-во сост. во врем.
		$t=1$	$t=2$	$t=3$	$t=4$	
att48	100	24	5	11	1	59
eil51	100	47	11	7	5	30
berlin52	100	2	12	5	3	78
st70	100	1	7	5	5	82
eil101	100	1	6	4	4	85
Сумма	500	75	41	32	18	334

Среднее значение экспериментального уровня временной состоятельности динамически адаптированного генетического алгоритма для решения задачи TSP равно 0,668, что почти в два раза выше, чем для генетического алгоритма до адаптации.

Представим зависимость количества состоятельных во времени решений, полученных в процессе эксперимента в первые t периодов, от t. Общее количество проведенных прогонов (тестов) равно $M|N||P|$. Рассмотрим функцию (8), значения которой равно количеству состоятельных во времени решений после завершения каждого периода:

$$c(s,t) = M|N||P| - \sum_{k=0}^{t} b(s,k). \tag{8}$$

На рис. 1 непрерывная линия соответствует значениям функции $c(s,t)$ при расчёте экспериментального уровня временной

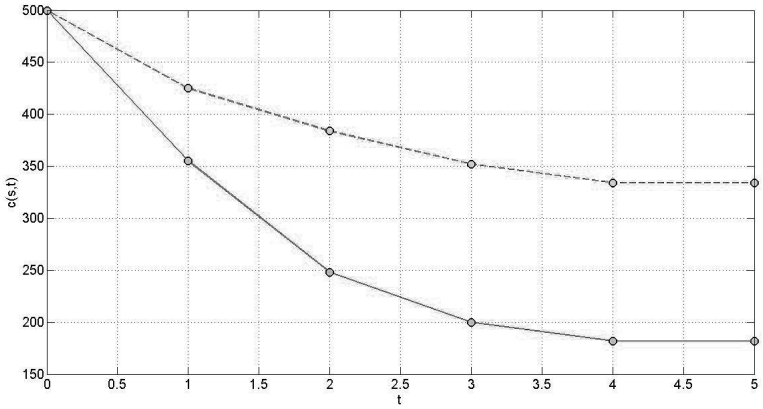

Рис. 1. Зависимость количества состоятельных во времени решений от периода для генетического алгоритма и процедуры динамической адаптации ГА

состоятельности генетического алгоритма с эвристическим кроссовером Греффенстетта, а пунктирная линия – значениям этой функции для динамической адаптации этого алгоритма.

Сравнив значения таблиц 1 и 2 отметим, что после процедуры динамической адаптации значение экспериментального уровня временной состоятельности решений увеличилось как для каждой тестовой задачи, так и в среднем для всех.

6.2. Сравнение генетического алгоритма с эвристическим кроссовером Греффенстетта и его динамической адаптации

На примере задачи TSP и генетического алгоритма было показано, что динамический подход позволяет получать решения с более высоким уровнем временной состоятельности.

Проверим, как изменится длина маршрутных планов при данной процедуре. В таблице 3 представлено сравнение длин маршрутов (решений), сгенерированных генетическим алгоритмом и динамической адаптацией генетического алгоритма. Средние значения целевой функции приведены для 500 прогонов алго-

ритмов. Критерием останова для каждого запуска генетического алгоритма с эвристическим кроссовером Греффенстетта выступало заданное число поколений. Оно равнялось 100. Процедура динамической адаптации проводилась по схеме, приведённой в алгоритме 3, при следующем наборе параметров: $T = 5, |N| = 20$.

Введём следующие обозначения: значение $l_{GA}(p)$ равно длине маршрута, полученного генетическим алгоритмом для тестовой задачи p, $l_{DAGA}(p)$ – длина маршрута, полученного при использовании динамически адаптированного генетического алгоритма. В последних двух столбцах таблицы 3 приведены значения целевой функции, соответствующие эффективному запуску. Эффективным запуском процедуры динамической адаптации для тестовой задачи p будем называть запуск, для которого значение величины $l_{GA}(p) - l_{DAGA}(p)$ максимально в проведённом эксперименте.

Таблица 3. Сравнение длины маршрутных планов, сгенерированных генетической эвристикой с эвристическим кроссовером Греффенстетта и её динамической адаптацией

Тестовая задача	Среднее значение f		Эффективный запуск	
	Генетич. алгоритм	Динами. адаптация	Генетич. алгоритм	Динам. адаптация
att48	12270,72	11723,79	11657	10928
eil51	493,72	464,64	493	448
berlin52	8836,31	8257,36	8570	7893
st70	803,25	758,61	765	708
eil101	779,30	720,338	725	665

Из таблицы 3 видно, что для каждой из рассмотренных тестовых задач среднее значение сгенерированных решений алгоритмом динамической адаптации меньше, чем классическим генетическим алгоритмом с эвристическим кроссовером Греффенстетта. Статистику улучшения проведём по формуле

$$k = \frac{l_{GA}(p) - l_{DAGA}(p)}{l_{GA}(p)} 100\%. \tag{9}$$

Средний процент улучшения равен 6,01%. Максимальное значение улучшения получено для тестовой задачи eil51 и составляет 9,12%.

6.3. Время работы динамической адаптации генетического алгоритма

Процедура динамической адаптации генетического алгоритма с эвристическим кроссовером Греффенстетта требует дополнительных запусков эвристики на каждом периоде, тем самым увеличивая время получения одного маршрутного плана по сравнению с классическим подходом. Проведём следующий эксперимент. Пусть t_{dyn} – средняя продолжительность работы (в секундах) алгоритма динамической адаптации при генерировании маршрута. . Ограничим время работы классической генетической эвристики величиной t_{dyn}. Для каждой тестовой задачи проведём по 100 запусков генетического алгоритма и его динамической адаптации и сравним результаты двух методов при фиксированной продолжительности их работы, равной t_{dyn}.

Таблица 4. Сравнение среднего значения длины генерируемых маршрутов при работе алгоритмов при различных фиксированных значениях t_{dyn}

Тестовая задача	Среднее значение f		t_{dyn}
	Генетический алгоритм	Динамическая адаптация	
att48	12273,51	11723,79	120,1
eil51	492,44	464,64	165,5
berlin52	8820, 25	8257,36	206,35
st70	807,37	758,61	1300,4
eil101	786,54	720,338	3403,7

Как видно из таблицы 4, при фиксированном времени работы метод динамической адаптации генетической эвристики способен генерировать лучшие решения по сравнению с генетическим алгоритмом с эвристическим кроссовером Греффенстетта. Данное замечание стоит учитывать в задачах большой размер-

ности, когда время генерации одного маршрута может занимать десятки минут. Как показал проведенный эксперимент, в случае заданного в условии задачи промежутка времени для генерации маршрутного плана выгоднее использовать процедуру динамической адаптации вместо классической эвристики, распределив время между соответствующими периодами работы алгоритма.

7. Заключение

В ходе исследования был подробно описан и реализован на языке программирования Java генетический алгоритм с эвристическим кроссовером Греффенстетта решений задачи TSP. Произведена оценка уровня временной состоятельности решений, генерируемых данной эвристикой. Проведенные расчеты для набора тестовых задач из библиотеки TSPLib продемонстрировали, что генетический алгоритм с эвристическим кроссовером Греффенстетта генерирует маршруты с невысоким уровнем временной состоятельности.

Для генерации решений с более высоким уровнем временной состоятельности предложена процедура динамической адаптации генетического алгоритма с эвристическим кроссовером Греффенстетта, в результате применения которой уровень временной состоятельности решений, генерируемых полученной эвристикой, увеличен в два раза. При этом среднее значение длины маршрутов уменьшилось на 6,01%. Таким образом, использование на практике предложенного динамического подхода, как показал численный эксперимент, позволяет генерировать маршруты меньшей длины.

Было показано, что при фиксированном ограничении на время работы процедура динамической адаптации генетического алгоритма с кроссовером Греффенстетта позволяет получать маршруты меньшей длины.

Стоит отметить, что предложенный метод динамической адаптации может быть применён и к другим эвристическим алгоритмам решения задачи TSP. Использование данного метода при решении задач транспортной маршрутизации таких как TSPTW,

IRP, MDVRP, PDP возможно, если эвристический алгоритм способен решать как саму задачу, так и её подзадачи [45, 46].

Литература

1. БОРИСОВСКИЙ П.А., ЕРЕМЕЕВ А.В. *О сравнении некоторых эволюционных алгоритмов* // Автоматика и телемеханика. – 2004. – №3. – С. 3–9.
2. БРОНШТЕЙН Е.М., ЗАИКО Т.А. *Детерминированные оптимизационные задачи транспортной логистики* // Автоматика и телемеханика. – 2010. – №10. – С. 133–147.
3. ЕРЗИН А.И. *Задачи маршрутизации.* – Новосибирск: РИЦ НГУ, 2014. – 95 с.
4. ИВАНКО Е.Е. *Метод масштабирования в приближенном решении задачи коммивояжёра* // Автоматика и телемеханика. – 2011. – №12. – С. 115–129.
5. ИПАТОВ А.В. *Модифицированный метод отжига в задача маршрутизации транспорта* // Труды института математики и механики УрО РАН. – 2011. – Т. 17. – №4. – С. 121–125.
6. МЕЛАМЕД И.И., СЕРГЕЕВ С.И., СИГАЛ И.Х. *Задача коммивояжёра. Вопросы теории* // Автоматика и телемеханика. – 1989. – №9. – С. 3–33.
7. МЕЛАМЕД И.И., СЕРГЕЕВ С.И., СИГАЛ И.Х. *Задача коммивояжёра. Приближенные алгоритмы* // Автоматика и телемеханика. – 1989. – №11. – С. 3–26.
8. МУГАЙСКИХ А.В. *Динамическая адаптация генетического алгоритма для задачи коммивояжёра* // Процессы управления и устойчивость. – 2015. – Т. 2. – №1. – С. 665–670.
9. ПЕТРОСЯН Л.А., ЗЕНКЕВИЧ Н.А. *Принципы устойчивой кооперации* // Управление большими системами. – 2009. – Вып. 3. – С. 100–120.
10. САЗОНОВ В.В., СКОБЕЛЕВ П.О., ЛАДА А.Н. И ДР. *Применение мультиагентных технологий в транспортной задаче с временными окнами и несколькими пунктами погрузки* // Управление большими системами. – 2016. – Вып. 64. – С. 65–80.

11. ТОВСТИК Т.М, ЖУКОВА Е.В. *Алгоритм приближенного решения задачи коммивояжёра* // Вестник С.-Петерб. ун-та. – 2013. – Сер. 1, вып. 1. – С. 101–109.
12. УРАКОВ А.Р., ТИМЕРЯЕВ Т.В. *Алгоритм решения динамической задачи поиска кратчайших расстояний в графе* // Управление большими системами. – 2017. – Вып. 65. – С. 60–86.
13. ЧЕНЦОВ А.Г. *Задача последовательного обхода мегаполисов с условиями предшествования* // Автоматика и телемеханика. – 2014. – №4. – С. 170–190.
14. ЧЕНЦОВ А.Г. *Одна параллельная процедура построения функции Беллмана в обобщённой задаче курьера с внутренними работами* // Автоматика и телемеханика. – 2012. – №3. – С. 134–139.
15. ANBUUDAYASANKAR S.P., GANESH K., MOHAPATRA S. *Survey of Methodologies for TSP and VRP* // Models for Practical Routing Problems in Logistics. – 2014. – P. 11–42.
16. BELLMAN R. *Dynamic Programming.* – Princeton: Princeton University Press, 1957. – 392 p.
17. BELLMAN R. *Dynamic programming treatment of the travelling salesman problem* // Journal of the ACM. – 1962. – Vol. 9. – P. 61–63.
18. BLUM C., ROLI A. *Metaheuristics in combinatorial optimization: Overview and conceptual comparison* // ACM Comput. Surv. – 2003. – Vol. 35, No. 3. – P. 268–308.
19. CHIU K.-M., LIU J.-S. *Robot routing using clustering-based parallel genetic algorithm with migration* // Proc. of Merging Fields Of Computational Intelligence And Sensor Technology. – 2011. – №9. – P. 42–49.
20. CHRISTOFIDES N. *Worst-case analysis of a new heuristic for the travelling salesman problem* // Technical Report 388. – Graduate School of Industrial Administration, Carnegie Mellon University, 1976. – 11 p.

21. COMARELA G., GONCALVES K., PAPPA G.L., ALMEIDA J., ALMEIDA V. *Robot routing in sparse wireless sensor networks with continuous ant colony optimization* // Proc. of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO'11), New York, NY, USA, ACM. – 2011. – P. 599–606.
22. DANTZIG G.B. *Linear programming and extensions.* – Princeton: Princeton Univ. Press, 1963. – 219 p.
23. DORIGO M., GAMBARDELLA L.M. *Ant colonies for the travelling salesman problem* // Biosystems. – Vol. 43, No. 2. – 1997. – P. 73–81.
24. DORRONSORO B. *The VRP Web.* – Languages and Computation Sciences Department, University of Malaga. – URL: http://www.bernabe.dorronsoro.es/vrp/.
25. FURINI F., PERSIANI C.A., TOTH P. *The Time Dependent Traveling Salesman Planning Problem in Controlled Airspace* // Transportation Research Part B: Methodological. – 2016. – Vol. 90. – P. 38–55.
26. GENDREAU M., POTVIN J.-Y. *Handbook of Metaheuristics.* – Springer Publishing Company, 2010. – 649 p.
27. GENDREAU M., YEHERTZO A., LAPORTE G., STAN M. *A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows* // Oper. Res. – 1998. – Vol. 46, No. 3. – P. 330–335.
28. GRANVILLE V., KRIVANEK M., RASSON J.-P. *Simulated annealing: A proof of convergence* // IEEE Trans. Pattern Anal. Mach. Intell., 1994. – Vol. 16, No. 6. – P. 652–656.
29. GREFENSTETTE J.J., GOPAL R., ROSMAITA B.J., VAN GUCHT D. *Genetic algorithms for the traveling salesman problem* // Proc. of the 1st Int. Conference on Genetic Algorithms. – Hillsdale: Lawrence Erlbaum Associates, 1985. – P. 160–168.
30. GUTIN G., PUNNEN A.P. *The traveling salesman problem and its variations.* – Kluwer Academi, 2002. – 837 p.

31. GUTIN G., YEO A., ZVEROVICH A. *Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp* // Discrete Applied Mathematics. – 2002. – Vol. 117, No. 1–3. – P. 81–86.
32. HELD M., KARP R.M. *A dynamic programming approach to sequencing problems* // Proc. of the 1961 16th ACM National Meeting, ACM. – 1961. – P. 201–204.
33. HELSGAUN K. *An effective implementation of the lin-kernighan traveling salesman heuristic* // European Journal of Operational Research, 2000. – Vol. 126. – P. 106–130.
34. HELSGAUN K. *General k-opt submoves for the lin-kernighan tsp heuristic* // Math. Program. Comput. – 2009. – Vol. 1, No. 2–3. – P. 119–163.
35. HOLLAND J.H. *Adaptation in natural and artificial systems.* – Ann Arbor: The University of Michigan Press, 1975. – 183 p.
36. HURKENS COR A.J., WOEGINGER G.J. *On the nearest neighbor rule for the traveling salesman problem* // Oper. Res. Lett., 2004. – Vol. 32, No. 1. – P. 1–4.
37. JOHNSON D.S., MCGEOCH L.A. *Experimental analysis of heuristics for the stsp. In Local Search in Combinatorial Optimization.* – Wiley and Sons, 2001. – 80 p.
38. KHAN I.H. *Assessing Different Crossover Operators for Travelling Salesman Problem* // I.J. Intelligent Systems and Applications. – 2015. – Vol. 1. – P. 19–25.
39. KNEPPER R.A., LAYTON T., ROMANISHIN J., RUS D. *Ikeabot: An autonomous multi-robot coordinated furniture assembly system* // Robotics and Automation. – 2013. – P. 855–862.
40. MILLER C.E., TUCKER A.W., ZEMLIN R.A. *Integer programming formulation of traveling salesman problems* // Journal of the ACM. – 1960. – Vol. 7, No. 4. – P. 326–329.
41. PAPADIMITRIOU C.H., STEIGLITZ K. *Combinatorial Optimization: Algorithms and Complexity.* – Upper Saddle River: Prentice-Hall, Inc., 1982. – 496 p.

42. PULJIC K., MANGER C.R. *Comparison of eight evolutionary crossover operators for the vehicle routing problem* // Mathematical Communications. – 2013. – Vol. 18. – P. 359–375.
43. REINELT G. *Travelling Salesman Problem Library.* – URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.
44. SIVASOUNDARI A., KALAIMANI S. *Wireless surveillance robot with motion detection and live video transmission* // Int. Journal of Emerging Science and Engineering. – 2013. – No. 1. – P. 147–165.
45. SHIROKIKH V.A., ZAKHAROV V.V. *Dynamic Adaptive Large Neighbourhood Search for Inventory Routing Problem* // Advances in Intelligent Systems and Computing. – 2015. – Vol. 359. – P. 231–241.
46. SHIROKIKH V.A., ZAKHAROV V.V. *Heuristic evaluation of the characteristic function in the Cooperative Inventory Routing Game* // Journal on Vehicle Routing Algorithms. – 2017. – P. 1–14.
47. TALBI E.-G. *Metaheuristics for Bi-level Optimization*. – Springer Publishing Company, Inc., 2013. – 288 p.
48. WANG Y., WU C.H. *Robot-assisted sensor network deployment and data collection* // Proc. of the 2007 IEEE Int. Symposium on Computational Intelligence in Robotics and Automation. – 2007. – P. 467–472.
49. ZACHARIASEN M, DAM M. *Tabu Search on the Geometric Traveling Salesman Problem* // Proc. from Metaheuristics Int. Conference, Colorado, 1996. – P. 571–587.
50. ZAKHAROV V.V., DEMENTIEVA M. *Multistage cooperative games and problem of time consistency* // Int. Game Theory Review. – 2004. – Vol. 6. – P. 157–170.
51. ZAKHAROV V.V., SHCHEGRYAEV A.N. *Multi-period cooperative vehicle routing games* // Contributions to Game Theory and Management. – 2014. – Vol. 7, No. 2. – P. 349–359.

52. ZAKHAROV V.V., SHCHEGRYAEV A.N. *Stable Cooperation in Dynamic Vehicle Routing Problems* // Automation and Remote Control. – 2015. – Vol. 76, No. 5. – P. 935–943.

DYNAMIC ADAPTATION OF GENETIC ALGORITHM FOR THE LARGE-SCALE ROUTING PROBLEMS

Victor Zakharov, Saint-Petersburg State University, Saint-Petersburg, Doctor of Science, professor (v.zaharov@spbu.ru).
Alexander Mugayskikh, Saint-Petersburg State University, Saint-Petersburg, student (alexander.mugaiskih@gmail.com).

Abstract: This paper is devoted to implementation of the dynamic adaptation procedure for genetic algorithm used for the traveling salesman problem on large-scale networks. It is shown that solutions obtained by genetic algorithm can be improved during its dynamic adaptation and allow finding the more effective routes for the equal time. To evaluate effectiveness of new approach computational experiments were performed on well-known benchmark instances from TSPLib. As a result, the experimental level of time consistency of improved solution considerably increases compare to basic one. Dynamically adapted genetic algorithm has demonstrated possibility to produce solutions with higher level of time consistency. At the same time proposed procedure reduces length of the one solution in certain experiment as well as average length of all routes in it.

Keywords: time consistency, genetic algorithm, routing problems.

Статья представлена к публикации членом редакционной коллегии О.П. Кузнецовым.

Поступила в редакцию 17.04.2017.
Дата опубликования 31.05.2018.