САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ ЭНЕРГЕТИЧЕСКИХ СИСТЕМ

Гагаринский Андрей Алексеевич

Выпускная квалификационная работа бакалавра

Муравьиный алгоритм в задаче маршрутизации вывоза и доставки товаров с временными окнами

Направление 010400

Прикладная математика, фундаментальная информатика и программирование

Научный руководитель,

кандидат физ.-мат. наук,

доцент

Власова Т. В.

Санкт-Петербург

2017

Содержание

Введение

Проблема мартрутизации транспорта (Vehicle Routing Problem — VRP) является одним из классов задач логистики. Целью решения VRP является построение маршрутов для одного или нескольких транспортных средств, которые будут считаться оптимальными по отношению к заданным критериям.

Впервые проблема маршрутизации была сформулирована Г. Дангицом и Дж. Рамсером в 1959 году как обобщенный случай задачи коммивояжера. Тогда ее постановка заключалась в поиске оптимального маршрута, по которому конкретно заданное количество однородных товаров должно было быть доставлено со склада заказчикам с использованием транспортных средств одинаковой грузоподъемности. Предполагалось, что стоимость транспортировки определяется только общим числом пройденного пути и не зависит от стоимости самого груза; также все транспортные средства должны были вернуться на склад.

В настоящее время вопрос оптимизации транспортной логистики является очень важным для большинства предприятий. От решения этой задачи напрямую зависит цена на товар, а в некоторых областях рынка расходы на доставку товара соизмеримы с его стоимостью. При правильной организации транспортировки можно существенно уменьшить экономические за-

траты. Именно поэтому сейчас стало важным иметь автоматизированное оптимальное решение данной проблемы.

В данной работе будет представлен алгоритм, решающий одну из постановок задачи маршрутизации автотранспорта.

Постановка задачи

Целью данной работы автоматизация построения кратчайших маршрутов для задачи 1-1 маршрутизации доставки и вывоза товара с временными окнами. Для достижения поставленной цели были поставлены следующие задачи:

- Освоение принципов работы эвристических и метаэвристических методов решения оптимизационных задач, в том числе и муравьиного алгоритма.
- Программная реализация различных вариаций алгоритма муравьиной колонии.
- Сравнение результатов реализации между собой.
- Модернизация алгоритма с целью его улучшения.

Глава 1. Краткий обзор задач маршрутизации с вывозом и доставкой товаров

В данной работе рассматривается задача маршрутизации с доставкой и вывозом товаров (Vehicle Route Problem with Pickup and Delivery, VRPPD) [1]. Предполагается 2 типа клиентов: те, которым товар доставляют (pickup), и те, у которых товар забирают и отвозят на склад или другому потребителю (delivery). Методы решения таких задач варьируются в зависимости от различных ограничений, которые вводятся для приведения VRPPD к более узкому виду. Ими могут быть, например, количество транспортных средств в автопарке, количество отправных точек, разная грузоподъемность у автомашин и т.п.

Очень часто VRPPD отличают по типу взаимоотношений между клиентами в сети. Так, выделяют 3 основных вида задач: *One-to-One* (сокращенно *1-1*) [2], *One-to-Many-to-One* (сокращенно *1-M-1*) [3] и *Many-to-Many* (сокращенно *M-M*) [4, 5].

§1.1. Задача маршрутизации 1-1 вывоза и доставки товаров

В задаче *One-to-One* маршрутизации (Рис. 1) главная особенность заключается в том, что обслуживание происходит между определенными парами клиентов: от конкретного производителя необходимо доставить товар одному потребителю [6].

Данный вид задач на практике часто встречается при перевозке людей, и формируется задача адресной перевозки *(Dial-A-Ride Problem)* [7, 8]. Например, вызов такси, машин скорой помощи и т.д. Вызванное транспортное средство забирает человека в пункте А и привозит в пункт Б, затем производит аналогичные действия со следующими пассажирами. Стоит отметить, что при решении данного типа задач целевой функцией, которую необходимо оптимизировать, будет являться время.

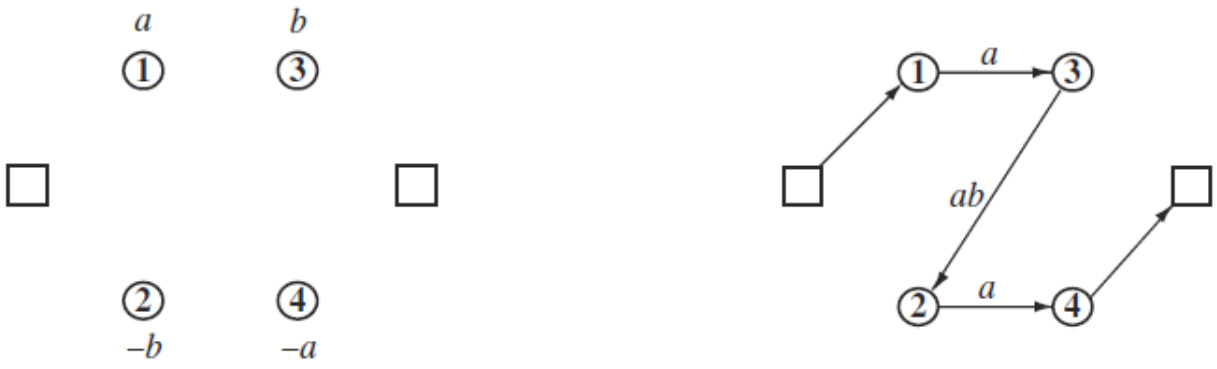


Рис. 1: One-to-One Pickup and Delivery Problem [2]

§1.2. Задача маршрутизации 1-M-1 вывоза и доставки товаров

Особенностью задачи *1-M-1* (Рис. 2) является то, что поставщиком здесь является склад, с которого потребителям развозят необходимое количество товара. Запрос клиента может быть как *смешанным* [9], когда требуется привезти и вывезти определенное количество товара, так и *простым* [10], когда необходимо либо только привезти, либо только вывезти товар.

Также маршрут транспортного средства в данной задаче можно задать по-разному: вывоз и доставка товара может осуществляться одновременно, тогда маршрут называют *объединенным* [11], или вывоз товара будет производиться только по завершению всех доставок, тогда это маршрут *с обратным грузом* [12]. Ярким примером задачи *1-M-1* является любая доставка товаров с последующей возможностью возврата тары или изношенных частей.

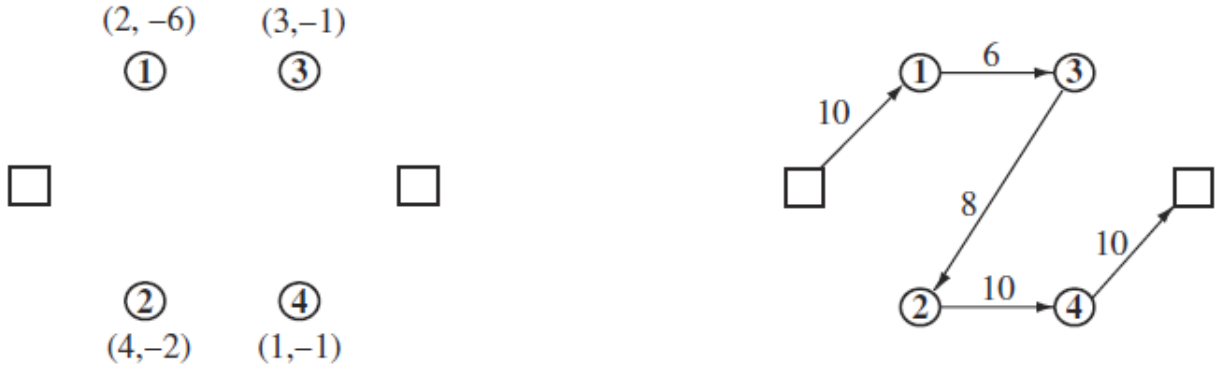


Рис. 2: One-to-Many-to-One Pickup and Delivery Problem [2]

§1.3. Задача маршрутизации M-M вывоза и доставки товаров

Задача *M-M* (Рис. 3) характерна тем, что в ней любая точка сети может быть как потребителем, так и производителем, причем даже не для одного транспортного средства. Нет необходимости доставлять товар от конкретного «продавца» конкретному «покупателю», как это было в задаче 1-1. Транспортному средству необходимо пройти всех клиентов, при этом, желательно, контролируя количество товара, имеющееся у него в наличии, для оптимизации суммарного пройденного расстояния путем уменьшения числа возвращений на склад для загрузки/разгрузки товара.

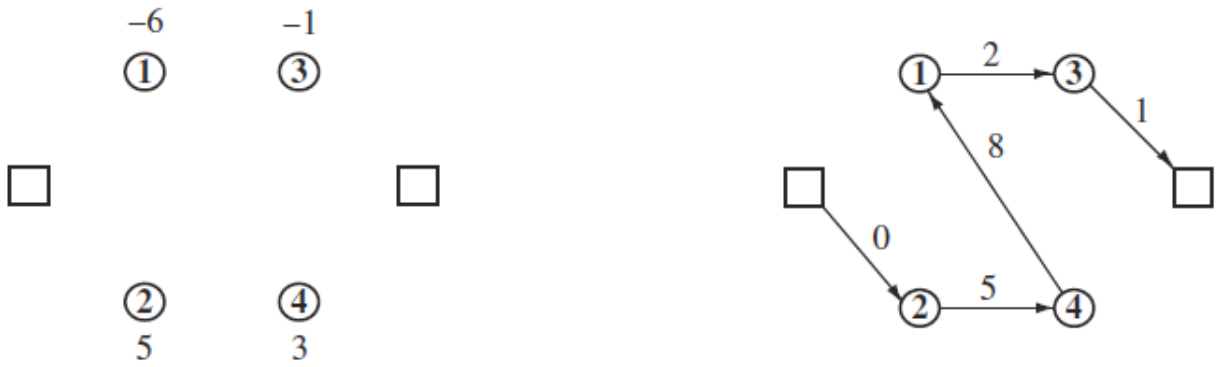


Рис. 3: Many-to-Many Pickup and Delivery Problem [2]

§1.4. Основные виды задачи 1-1 VRPPD

В данной работе представлен один из способов решения задачи 1-1 вывоза и доставки товаров. Как правило, при решении данной проблемы необходимо построить маршрут транспортного средства минимальной стоимости, который будет начинаться и заканчиваться в депо.

В зависимости от введённых ограничений выделяют следующие основные виды 1-1 VRPPD:

- с возможностью возврата (VRPPD with BackHauls — VRPPDB) [13–15];
- с ограничением на грузоподъёмность (Capasicated VRPPD — CVRPPD) [16, 17];
- с ограничением на временные окна (VRPPD with Time Windows — VRPPDTW) [17–22];
- периодическая задача (Periodic VRPPD — VRPPDP) [23];
- с промежуточными понижениями (VRPPD with Intermediate Drops — VRPPDID) [24].

§1.5. Математическая постановка задачи 1-1 доставки и вывоза товаров с временными окнами

Общую формулировку задачи 1-1 маршрутизации автотранспорта с одним транспортным средством впервые предложили Руланд и Родин в 1997 году [25], представив её в виде задачи линейного программирования. В 2002 году Лау и Лианг в статье [26], усложнили модель, введя ограничения на временные окна и изменив количество транспортных средств.

Постановка выглядит следующим образом.

В нашей модели предполагается, что существует неограниченное количество транспортных средств, и все они имеют одинаковую грузоподъёмность.

Пусть N — набор запросов на транспортировку. Для каждого запроса $i \in N$ груз в количестве q_i должен быть доставлен из источника N_i^+ в место назначения N_i^- (положительное q_i для вывоза товара и отрицательное для доставки). Определим $N^+ = \bigcup_{i \in N} N_i^+$ и $N^- = \bigcup_{i \in N} N_i^-$ как множества точек отправления и мест назначения. Для удобства, предположим что N_i^+ и N_i^- не пересекаются. Пусть $V = N_i^+ \bigcup N_i^-$ и $n = |V|$. Обозначим M и m как множество и количество транспортных средств соответственно. Каждое транспортное средство имеет грузоподъёмность Q, начинает и заканчивает свой маршрут в

депо O без груза.

Пусть d_{ij} — расстояние от точки i до точки j для любых $i, j \in V \bigcup O$, а t_{ij} — время на преодоление этого пути. Также пусть $[e_i, l_i]$ обозначает временное окно, т.е. временной интервал, в течение которого вершина i должна быть обслужена. Следует подчеркнуть, что длительность обслуживания каждого клиента может быть легко включена во время поездки, поэтому для простоты примем ее равной *0*.

Определение 1. *Транспортным маршрутом* R_k для транспортного средства k называется направленный маршрут через подмножество вершин $V_k \subset V$, для которого выполняются следующие условия:

1. R_k начинается и заканчивается в депо O.
2. Если N_i^+ принадлежит V_k, то и N_i^- также принадлежит V_k для всех $i \in N$, и наоборот.
3. Если N_i^+ и N_i^- принадлежат V_k, то N_i^+ должен быть посещен раньше, чем N_i^-.
4. Транспортное средство k посещает каждую точку V_k лишь однажды.
5. Загруженность транспортного средства в любой момент времени не превышает Q.

6. Время прибытия A_i и время выезда D_i из любой точки i удовлетворяет условию $D_i \in [e_i, l_i]$, где $D_i = \max(A_i, e_i)$ (т.е. если $A_i < e_i$, то транспортное средство обязано ждать в точке i).

Определение 2. *Транспортным планом* R является набор транспортных маршрутов $R = \{R_k \mid k \in M\}$, для которого:

1. R_k является транспортным маршрутом для транспортного средства k для всех $k \in M$.
2. $\{V_k \mid k \in M\}$ — подмножество V.

В задачах VRPPDTW существует широкий круг целевых функций, но в данной работе рассмотрены следующие:

1. Минимизировать количество используемых транспортных средств, что является доминирующим фактором при расчете затрат на транспортировку.
2. Минимизировать общую длину плана, т.е. сумму длин всех транспортных маршрутов.

Чтобы сформулировать VRPPDTW в виде задачи линейного программирования, необходимо ввести еще несколько переменных. Пусть $z_{ik}(i \in V, k \in M) = 1$, если запрос i назначен транспортному средству k, иначе $z_{ik} = 0$; пусть $x_{ijk}(i \in V, j \in V, k \in M) = 1$, если транспортное средство k перемещается из точки i в точку j, иначе $x_{ijk} = 0$. Обозначим через y_j промежуточную переменную, которая хранит общую загруженность транспортного средства при посещении точки j.

При моделировании целевой функции для нашей задачи необходимо помнить, что минимизация общего количества задействованных транспортных средств более важна, чем минимизация общего пройденного пути. Поэтому обозначим через P стоимость задействования каждого транспортного средства. P зафиксирован как величина намного большая, чем максимальная общая длина пути. Таким образом, задача линейного программирования будет состоять в минимизации целевой функции $f(x)$ при следующих ограничениях:

$$min(P \times m + \sum_{k \in M} \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ijk}) \tag{1}$$

$$\sum_{k \in M} z_{ik} = 1, \forall i \in N \tag{2}$$

$$\sum_{k \in M} \sum_{j \in V} x_{ijk} = 1, \forall i \in V \tag{3}$$

$$\sum_{i \in V} x_{iOk} = 1, \forall k \in M \tag{4}$$

$$\sum_{j \in V} x_{Ojk} = 1, \forall k \in M \tag{5}$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0, (\forall h \in V)(\forall k \in M) \tag{6}$$

$$\sum_{i \in V} x_{ijk} Q \geq y_j, (\forall j \in V)(\forall k \in M) \tag{7}$$

$$x_{ijk} = 1 \Rightarrow y_i + q_i = y_j, (\forall i, j \in V \cup O)(\forall k \in V) \tag{8}$$

$$y_O = 0 \tag{9}$$

$$y_i \geq 0, \forall i \in V \tag{10}$$

$$x_{ijk} = 1 \Rightarrow D_i + t_{ij} \leq D_j, (\forall i, j \in V)(\forall k \in M) \tag{11}$$

$$p = N_i^+, q = N_i^-, D_p \leq D_q, \forall i \in N \tag{12}$$

$$D_O = 0 \tag{13}$$

(1) является целевой функцией, которую необходимо минимизировать. Ограничение (2) показывает, что каждому заказу будет соответствовать только одно транспортное средство. Ограничение (3) гарантирует, что каждая вершина будет посещена лишь один раз. Ограничения (4) и (5) говорят о том, что каждое транспортное средство отправляется со склада и прибывает на склад. (6) утверждает, что если транспортное средство прибыло в вершину, оно обязано будет оттуда уехать.

(7 – 10) в совокупности образуют ограничение на грузоподъемность. Временные окна и предшествование обеспечиваются ограничениями (11 – 13).

Глава 2. Муравьиный алгоритм в задачах маршрутизации автотранспорта с вывозом и доставкой товаров

§2.1. Разновидности алгоритмов для решения задачи маршрутизации с вывозом и доставкой товаров

Задача маршрутизации, также как и задача коммивояжера, является *NP-полной*. Поэтому для ее решения используются специальные алгоритмы. Эти алгоритмы классифицируются на точные, эвристические и метаэвристические. Рассмотрим каждый из типов и разберем, чем они отличаются:

1. *Точные* алгоритмы имеют свойство давать максимально точный результат (отсюда и название). К ним относятся

   - метод ветвей и границ [2, 3];
   - метод ветвей и отсечений [3];
   - динамическое программирование [27];
   - стохастическое программирование [28].

   Минус данных алгоритмов заключается в том, что они имеют очень большую зависимость от размерности задачи. Время работы таких алгоритмов при применении на практике слишком велико. Поэтому чаще в реальных задачах

больших размерностей применяют эвристические и метаэвристические алгоритмы.

2. *Эвристический* алгоритм — это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он дает достаточно хорошее решение в большинстве случаев. В действительности может быть даже известно (то есть доказано) то, что эвристический алгоритм формально неверен. Его все равно можно применять, если при этом он дает неверный результат только в отдельных, достаточно редких и хорошо выделяемых случаях, или же дает неточный, но все же приемлемый результат.

   Проще говоря, *эвристика* — это не полностью математически обоснованный (или даже «не совсем корректный»), но при этом практически полезный алгоритм. Выбор такого вида алгоритмов связан, в первую очередь, с высокой вычислительной сложностью NP-полных задач. Среди эвристических методов можно выделить

   - метод сбережений [29];
   - метод, основанный на совпадениях [30].

3. *Метаэвристические* алгоритмы являются, по сути, объеди-

нением основных эвристических методов в рамках алгоритмических схем более высокого уровня. Они позволяют работать с задачами больших размерностей, при этом решая их точнее, чем эвристики, и без сильных временных затрат:

- семейство генетических алгоритмов [31, 32];
- муравьиные алгоритмы [17–22];
- поиски с запретами [33–35];
- метод имитации отжига [36, 37].

§2.2. Биологическая интерпретация алгоритма муравьиной колонии

Муравьиный алгоритм (алгоритм оптимизации подражанием муравьиной колонии, ant colony optimization, ACO) — один из эффективных метаэвристических алгоритмов для нахождения приближенных решений задачи коммивояжера, а также решения аналогичных задач поиска маршрутов на графах.

В начале 90-х годов 20-го века доктор Марко Дориго впервые сумел формализовать поведение муравьев и применить стратегию их поведения для решения задачи о кратчайших путях [38]. Позже, Гамбарделла, Тайллард и Ди Каро разработали другие подходы к решению сложных оптимизационных задач с помощью муравьиных алгоритмов. На сегодняшний день ме-

тоды, использующие алгоритм муравьиной колонии, являются весьма конкурентоспособными и для некоторых задач дают наилучшие результаты.

Муравьи принадлежат к классу «социальных насекомых». Они обитают в пределах определеннного коллектива — семьи или колонии. Коллективная система способна разрешать сложные динамические задачи, которые не могли бы быть выполнены элементами этой системы по отдельности. Ее основной чертой является роевой интеллект (Swarm intelligence) — разновидность кооперативного поведения, используемая муравьями как стратегии выживания в мире дикой природы.

Для передачи информации друг другу муравьи используют феромоны — специальное вещество, которое откладывается, как след, во время движения муравья. Чем выше концентрация феромона на тропе, тем больше муравьев впоследствии пойдет по ней. Феромон со временем выветривается. Это позволяет муравьям лучше приспосабливаться к изменению окружающей среды.

На рис. 4 показаны результаты одного из наблюдений ученых. В нем муравьи ищут кратчайший путь от колонии до источника пищи. Действия колонии можно разделить на три этапа:

1. Первый муравей находит источник пищи (F) любым способом (a), а затем возвращается к гнезду (N), оставив за собой тропу из феромона (b).

2. Следующие муравьи выбирают один из четырех возможных путей, затем укрепляют его и делают привлекательным, откладывая свой феромон.

3. На более длинных маршрутах феромон испаряется сильнее, чем на коротких, поэтому муравьи выбирают оптимальную тропу.

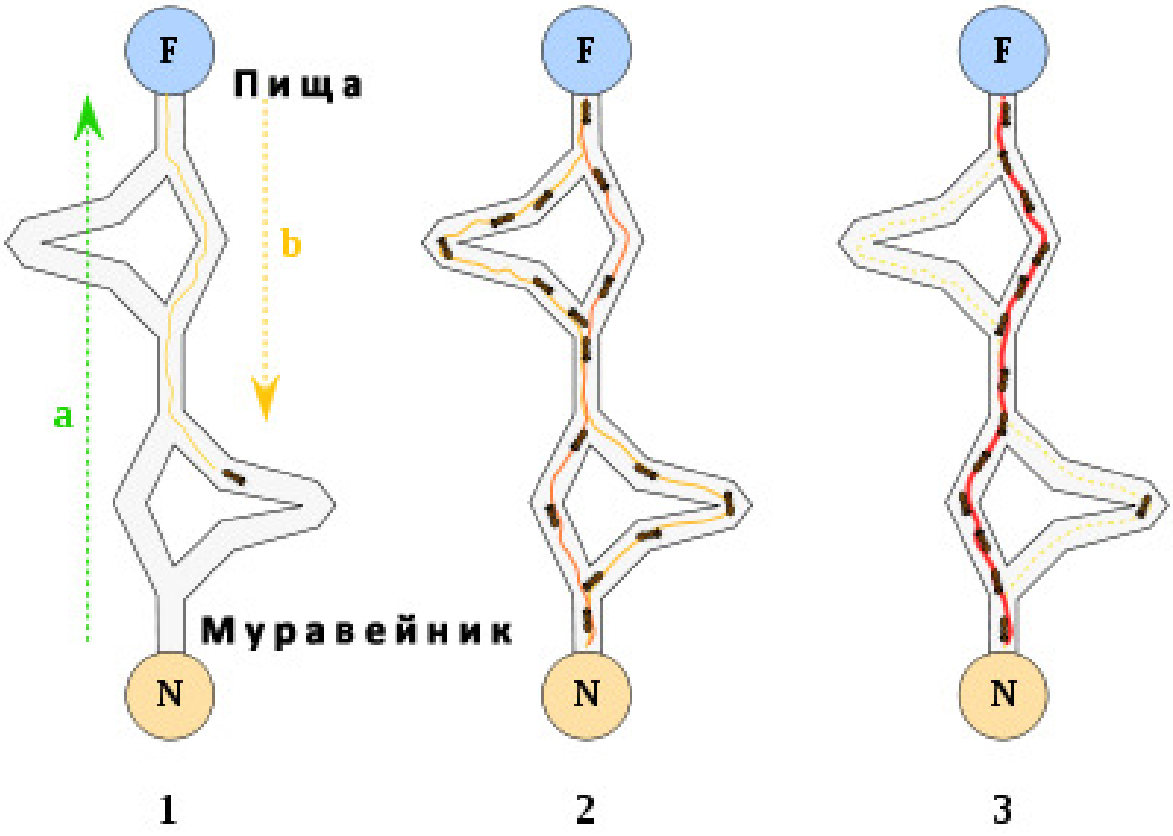


Рис. 4: Выбор муравьями оптимальных маршрутов

§2.3. Классический алгоритм муравьиной колонии при решении задачи коммивояжера

Задача коммивояжера заключается в нахождении самого выгодного маршрута, проходящего через указанные вершины хотя бы по одному разу с последующим возвращением в исходную точку. Более формально: дан граф $G = (X, U)$, где $|X| = n$ — множество вершин (города), $|U| = m$ — множество ребер (возможные пути между городами). Дана матрица чисел D_{ij}, где $i, j \in 1, 2, \ldots, n$, представляющих собой стоимость переезда из вершины x_i в x_j.

Необходимо найти такую перестановку $x \in X$, что целевая функция

$$f(x) = min(D_{x_1 x_n} + \sum_{i=1}^{n-1} D_{x_i x_{i+1}}) \tag{14}$$

С учетом особенностей задачи коммивояжера можно описать локальные правила поведения муравьев при выборе пути:

1. Каждый муравей обладает собственной «памятью» J_{ik} — список городов (*Tabu list*), которые необходимо посетить муравью k, находящемуся в городе i.

2. Муравьи обладают «зрением», обратно пропорциональным длине ребра

$$\eta_{ij} = 1/D_{ij} \tag{15}$$

«Зрение» определяет «жадность» выбора муравья: чем ближе к нему находится вершина, тем она лучше «видна», и тем больше агент стремится попасть в нее.

3. Любой муравей способен улавливать след феромона. Это пробуждает желание агента пройти по конкретному ребру. Уровень феромона в момент времени t на ребре D_{ij} будет соответствовать $\tau_{ij}(t)$.

4. Вероятность перехода муравья k из точки i в точку j устанавливается следующим соотношением:

$$\begin{cases} P_{ij,k}(t) = \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{l \in J_{i,k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}}, \\ P_{ij,k}(t) = 0, j \notin J_{i,k}, \end{cases} \tag{16}$$

где α, β — параметры, задающие веса следа феромона, коэффициенты эвристики. α и β определяют «жадность» муравья. При $\alpha = 0$ муравей стремится выбрать кратчайшее ребро, а при $\beta = 0$ — ребро с наибольшим количеством феромона.

5. Пройдя ребро (i, j), муравей откладывает на нем некоторое количество феромона, которое должно быть связано с оптимальностью сделанного выбора. Обозначим за $T_k(t)$ список вершин, пройденных муравьем k к моменту времени t,

$L_k(t)$ — длина этого маршрута, а Q — параметр, имеющий значение порядка длины оптимального пути. Тогда откладываемое количество феромона может быть задано в виде:

$$\Delta\tau_{ij,k}(t) = \begin{cases} \dfrac{Q}{L_k(t)}, (i,j) \in T_k(t), \\ 0, (i,j) \notin T_k(t), \end{cases} \tag{17}$$

В каждый последующий момент времени происходит испарение феромона на тропах. Пусть $p \in [0,1]$ — коэффициент выветривания, тогда правило испарения феромона имеет вид:

$$\tau_{ij}(t+1) = (1-p)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij,k}(t), \tag{18}$$

где m — число муравьев в одной колонии.

Перед стартом работы алгоритма количество феромона на ребрах принимается за малую положительную константу τ_0.

Общее количество муравьев в колонии является постоянным в течение всего времени решения задачи. Слишком большое число муравьев приводит к быстрому выделению субоптимальных маршрутов. Когда же муравьев очень мало, возникает опасность потери коллективного поведения из-за чрезмерно

быстрого испарения феромона. Обычно количество муравьев принимают равным числу городов — каждый муравей начинает маршрут из отдельного города.

Псевдокод простого алгоритма муравьиной колонии для задачи коммивояжера будет выглядеть следующим образом:

1. **Инициализация матрицы расстояний D.**
2. **Инициализация параметров** α, β **и** Q**.**
3. **Инициализация ребер графа — расчет видимости** η_{ij} **и присвоение ребрам начального феромона.**
4. **Размещение муравьев в начальные точки — случайные города без совпадений.**
5. **Выбрать длину первоначального кратчайшего маршрута**
   $L_{opt} = \infty$**.**
6. **Начать цикл по количеству колоний** $t = 1$, $t = t_{max}$**.**
7. **Начать цикл по всем муравьям** $k = 1, m$**.**
8. **Построить маршрут** $T_k(t)$**, используя распределение вероятности (16); рассчитать длину** $L_k(t)$**.**
9. **Завершить цикл по муравьям.**

10. Проверить все $L_k(t)$ на лучшее решение по сравнению с L_{opt}.

11. Если $L_k(t)$ предпочтительнее, обновить L_{opt} и T_{opt}.

12. Начать цикл по всем ребрам графа.

13. Обновить феромонные следы в соответствии с (17).

14. Завершить цикл по ребрам графа.

15. Завершить цикл по количеству колоний.

16. Вывести наилучший маршрут T_{opt} и его длину L_{opt}.

Вычислительная сложность муравьиного алгоритма зависит от количества колоний (итераций), количества точек в графе и количества муравьев — $O(t \times m \times n^2)$, где t — количество итераций, m — число муравьев в одной колонии и n — количество вершин в графе.

§2.4. Алгоритм муравьиной колонии для решения задачи 1-1 VRPPDTW

Формально, общие правила алгоритма муравьиной колонии для задач VRP ненамного отличаются от TSP-правил. Также, вводимые для муравьиного алгоритма ограничения очень зависят от вида VRP. Впервые применение данного алгоритма

в VRP было показано в работе Буллхеймера [40] в 1999 году. В ней в процедуру нахождения вероятности перехода в другой город добавляются оценка последовательности из двух городов и оценка использования грузовой мощности транспортного средства.

Первая величина рассчитывается как: $\mu_{ij} = d_{i0} + d_{0j} - d_{ij}$, где d_{i0} — расстояние от точки i до депо, d_{0j} — расстояние от депо до точки j и d_{ij} — расстояние от точки i до точки j. Чем больше данная величина, тем выгоднее нам после вершины i следовать в вершину j. Соответственно, чем она меньше, тем больше вероятность того, что муравей после точки i пойдет в депо дозагружаться товаром. Эта оценка очень хорошо подходит для *1-M-1* VRP, но в применении к *1-1* бесполезна.

Вторая оценка показывает загруженность транспортного средства после перехода из точки i в точку j. Пусть Q_i — количество товара в транспортном средстве при выезде из точки i, q_j — товарный запрос клиента j. Следует подчеркнуть, что в том случае, если клиент является поставщиком, величина q_j положительна, в противном же случае $q_j < 0$. За оценку эффективности использования грузовой мощности транспортного средства автор предлагает взять следующую величину: $\kappa_{ij} = (Q_i + q_j)/Q$, где Q — максимальная грузоподъемность

транспортного средства. Интерпретируя в 1-1 VRPPD, муравей будет стараться загрузить максимум товара у производителей, и лишь затем развозить его по потребителям. Эта стратегия сильно зависит от размещения потребителей и производителей на плоскости, однако имеет шанс на успех.

Приняв вторую оценку в качестве дополнительного параметра, имеем следующую формулу для расчета вероятности перехода муравья k из точки i в точку j:

$$\begin{cases} P_{ij,k}(t) = \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}[\kappa_{ij}(t)]^{\lambda}}{\sum_{l \in J_{i,k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}[\kappa_{ij}(t)]^{\lambda}}}, \\ P_{ij,k}(t) = 0, j \notin J_{i,k}, \end{cases} \tag{19}$$

где λ — параметр, наравне с α и β, определяющий степень зависимости выбора клиента от товарного запроса этого клиента.

Среди использования муравьиного алгоритма в работах по VRPPD [41, 42] более всего распространены ограничения, схожие с ограничением на последовательность из двух городов, рассмотренным в [40].

Большая работа по усовершенствованию алгоритма муравьиной колонии как для TSP, так и для VRP задач, была проделана Лукой Гамбарделла, и напечатана в его собственной книге [43] в 2014 – 2015 годах. В данной работе вводится три основ-

ных правила для улучшения производительности алгоритма:

1. Правило перехода из одной вершины в другую должно обеспечивать баланс между исследованием новых ребер и использованием накопленных априорных знаний.

2. Правило глобального обновления феромонов применяется только к ребрам, принадлежащим лучшему маршруту.

3. Правило локального обновления феромонов применяется для каждого маршрута в колонии.

Разберем все три правила подробнее:

- Баланс правила перехода от вершины i к вершине j обеспечивается следующим уравнением:

$$s = \begin{cases} argmax([\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}), q \leq q_0 \\ S, q > q_0, \end{cases} \tag{20}$$

где q — случайно сгенерированное число из промежутка $[0..1]$, q_0 — заданный параметр $(0 < q_0 < 1)$, а S — номер вершины, полученный при использовании правила (16). Данный метод называется *пропорциональным псевдорандомом (pseudo-random-proportional rule)*. Эксперементальным путем было показано, что применение данного правила увеличивает точность работы алгоритма от 5% до 20%.

Рекомендуемое значение параметра q_0 установлено равным 0.9

- Правило глобального обновления уровня феромонов применяется только на ребрах лучшего маршрута. Новое значение феромонного следа вычисляется по формуле:

$$\tau_{ij}(t+1) = (1-\gamma)\tau_{ij}(t) + \gamma\Delta\tau_{ij}(t), \tag{21}$$

где

$$\Delta\tau_{ij}(t) = \begin{cases} \frac{1}{L_{opt}}, (i,j) \in L_{opt} \\ 0, (i,j) \notin L_{opt}, \end{cases} \tag{22}$$

в котором L_{opt} — длина наилучшего найденного маршрута, γ — параметр выветривания феромона ($0 < \gamma < 1$). Процедура глобального обновления предназначена для «откладывания» большего количества феромонов на лучших (кратчайших) путях.

- Если обновлять только ребра наилучших решений, то очень скоро можно попасть в так называемый «локальный оптимум». В этой ситуации на ребрах лучших найденных решений будет расположено слишком большое количество феромона относительно остальных, и каждый последующий муравей с вероятностью 99% выберет именно это ребро. Правило локального обновления регулирует разницу феро-

монов между «лучшими» и «простыми» ребрами графа.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t), \tag{23}$$

где ρ — параметр $(0 < \rho < 1)$, и изменение

$$\Delta\tau_{ij}(t) = \tau_0. \tag{24}$$

Муравьиный алгоритм, удовлетворяющий этим трем правилам, называют алгоритмом муравьиной системы (*Ant Colony System — ACS*). Также обозначим за *ACS with capacity* алгоритм *ACS*, использующий при $q > q_0$ формулу (19) для определения расчета вероятности перехода. Это необходимо, чтобы проверить на тестовых примерах ее влияние на общую длину маршрута.

Псевдокод ACS будет выглядеть следующим образом:

1. **Инициализация матрицы расстояний D.**
2. **Инициализация параметров α, β и Q.**
3. **Инициализация ребер графа — расчет видимости η_{ij} и присвоение ребрам начального феромона.**
4. **Размещение муравьев в начальные точки — случайные города без совпадений.**
5. **Выбрать длину первоначального кратчайшего маршрута**

$L_{opt} = \infty$.

6. **Начать цикл по количеству колоний** $t = 1$, $t = t_{max}$.

7. **Начать цикл по всем муравьям** $k = 1, m$.

8. **Построить маршрут** $T_k(t)$, **используя распределение вероятности (20); рассчитать длину** $L_k(t)$.

9. **Завершить цикл по муравьям.**

10. **Начать цикл по муравьям.**

11. **Провести локальное обновление феромонов на ребрах, используя (23) и (24).**

12. **Завершить цикл по муравьям.**

13. **Проверить все** $L_k(t)$ **на лучшее решение по сравнению с** L_{opt}.

14. **Если** $L_k(t)$ **предпочтительнее, обновить** L_{opt} **и** T_{opt}, **провести глобальное обновлениес помощью (21) и (22).**

15. **Завершить цикл по количеству колоний.**

16. **Вывести наилучший маршрут** T_{opt} **и его длину** L_{opt}.

Глава 3. Решение задачи маршрутизации 1-1 доставки и вывоза товаров с одним транспортным средством

Было решено взять задачу именно с одним транспортным средством, поскольку данный тип задач лучше всего подходит для демонстрации работы алгоритмов. Также, задачи с единственным транспортным средством используются как вспомогательные при решении более сложных и объемных задач.

§3.1. Практическая постановка задачи 1-1 маршрутизации

Как уже отмечалось ранее, в работе рассматривается задача 1-1 маршрутизации вывоза и доставки товаров. Ее характерной особенностью является то, что каждому поставщику соответствует единственный конкретный потребитель. Наиболее ярким примером будет работа курьерской службы. Заказчиком услуг такого рода может быть любая *компания-производитель* товара, не имеющая собственного автопарка.

Пусть имеется одно транспортное средство и множество заказов от «производителей» товара на доставку конкретным «потребителям». У каждого клиента имеется свое временное

окно, в течение которого он может быть обслужен.

Перед выездом из депо мы имеем информацию о количестве потребителей и количестве производителей, их координатах на плоскости, а, соотвественно, и о расстояниях между ними. Количество товара, необходимое для доставки/вывоза из каждой точки сети также известно. Транспортное средство имеет конкретную грузоподъемность, которую нельзя превышать.

Необходимо построить маршрут минимальной стоимости, которая определяется расстоянием.

§3.2. Математическая формулировка

Изложенная задача курьерской службы является задачей маршрутизации 1-1 доставки и вывоза товаров с одним транспортным средством и ограничением на грузоподъёмность и временные окна. Таким образом, задача линейного программиро-

вания для нее примет вид:

$$min(\sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij}) \tag{25}$$

$$\sum_{j \in V} x_{ij} = 1, \forall i \in V \tag{26}$$

$$\sum_{i \in V} x_{iO} = 1, \tag{27}$$

$$\sum_{j \in V} x_{Oj} = 1, \tag{28}$$

$$\sum_{i \in V} x_{ih} - \sum_{j \in V} x_{hj} = 0, \forall h \in V \tag{29}$$

$$\sum_{i \in V} x_{ij} Q \geq y_j, \forall j \in V \tag{30}$$

$$x_{ij} = 1 \Rightarrow y_i + q_i = y_j, \forall i, j \in V \cup O \tag{31}$$

$$y_O = 0 \tag{32}$$

$$y_i \geq 0, \forall i \in V \tag{33}$$

$$x_{ij} = 1 \Rightarrow D_i + t_{ij} \leq D_j, \forall i, j \in V \tag{34}$$

$$p = N_i^+, q = N_i^-, D_p \leq D_q, \forall i \in N \tag{35}$$

$$D_O = 0 \tag{36}$$

Данные обозначения и неравенства отличаются от введённых в параграфе 1.5. только тем, что количество используемых транспортных средств равно единице.

§3.3. Решение задачи маршрутизации 1-1 доставки и вывоза товаров с одним транспортным средством и временными окнами с помощью муравьиного алгоритма

Решение задачи производилось на компьютере со следующими характеристиками: Процессор Intel Core i5 CPU M480 2.67 GHz, RAM 4 GB. Реализация алгоритма муравьиной колонии и его модификаций производилась в вычислительной среде MATLAB R2013b. В качестве исходных данных использовались тестовые выборки SET P1, сгенерированные для решения задачи CVRPPDTW. Формат текстового файла, содержащего начальные данные и пример самих данных размещены в *Приложении 1* и *Приложении 2* соответственно. Код программы приведен в *Приложениях 3-8.*

Для сравнения по результатам были выбраны 3 алгоритма: простейший *ACO*, *ACS* и *ACS with capacity.*

- *ACO* в данном контексте — алгоритм, использующий правила простого муравьиного алгоритма из задачи *TSP* в совокупности с ограничениями задачи VRPPDTW (25) — (36).
- *ACS* является улучшением *ACO*. Используя 3 основных правила этот алгоритм должен показывать лучшие результаты на тестовых данных по сравнению с *ACO*.

- *ACS with capacity* — еще не проверявшийся для задачи 1-1 VRPPDTW алгоритм, который задействует всего один дополнительный параметр.

Результаты работы этих алгоритмов можно увидеть в *Таблице 1* и *Таблице 2*.

Для работы всех трех алгоритмов были сформированы одинаковые инициализирующие параметры:

$t_{max} = 50$ — максимальное число итераций (колоний), $n = 100$ — количество городов (вершин графа), $k = 100$ — количество муравьев в каждой колонии, $\alpha = 1$ — коэффициент запаха, $\beta = 2$ — коэффициент видимости, $Q = 1$ — участвует в обновлении феромонов, $\tau_0 = 0.1$ — начальное количество феромона на тропах, $p = 0.5$ — коэффициент испарения феромона; $q = 0.9$ — параметр пропорционального псевдо-рандома, $\gamma = 0.5$ — параметр выветривания в глобальном обновлении феромонов ACS, $\rho = 0.5$ — параметр выветривания в локальном обновлении феромонов; $\lambda = 2$ — параметр, показывающий степень зависимости выбора клиента от его товарного запроса. По полученным данным видно, что алгоритм ACS показывает наилучшие результаты как при подсчетах средних дистанций, так и в маршрутах наименьшей стоимости. Введение трех основных правил повлекло за собой незначительное увеличение време-

Таблица 1: результат работы алгоритмов на тестовых примерах.

Название выборки	*ACO*			*ACS*		
	Среднее	t	Лучшее	Среднее	t	Лучшее
1P1.DAT	1227.2	51.4	1200.9	1150.6	53.5	1118.1
2P1.DAT	1201.2	50.3	1158.9	1199.5	54.0	1142.6
3P1.DAT	1240.7	52.5	1204.3	1204.7	53.0	1131.1
4P1.DAT	708.1	52.1	700.9	705.2	52.8	700.1
5P1.DAT	690.9	52.2	680.6	672.6	52.7	662.8
6P1.DAT	705.1	51.3	696.6	690.8	53.4	679.7
7P1.DAT	1019.2	52.8	1015.9	1014.2	52.5	1009.0
8P1.DAT	1189.2	51.0	1163.4	1159.6	53.5	1132.4
9P1.DAT	1201.2	51.4	1163.4	1165.6	53.0	1138.4
10P1.DAT	1053.6	51.5	1023.5	1029.8.6	52.9	1007.4

ни работы алгоритма, при этом хорошо повысив его точность. Алгоритм *ACS with capacity* является еще более затратным по времени: при подсчете вероятности перехода в нем возводятся в степень сразу три числа, в то время, как в двух других только два. Результаты работы алгоритма *ACS with capacity* различны относительно двух других алгоритмов для каждого набора данных. Это может быть следствием того, что данное решение имеет более сильную зависимость от расположения клиентов на плоскости.

Говоря о временных затратах на работу алгоритма, хочется упомянуть вычислительную сложность муравьиных алгоритмов. Она зависит от таких параметров как t_{max} — мак-

Таблица 2: результат работы CS with capacity на тестовых примерах.

Название выборки	*ACS with capacity*		
	Среднее	t	Лучшее
1P1.DAT	1252.2	58.9	1220.9
2P1.DAT	1215.2	60.5	1166.4
3P1.DAT	1232.2	59.4	1200.9
4P1.DAT	752.2	59.5	725.4
5P1.DAT	707.4	59.8	705.3
6P1.DAT	705.0	59.9	695.8
7P1.DAT	1050.5	51.8	1030.2
8P1.DAT	1194.3	58.4	1145.1
9P1.DAT	1176.4	58.3	1151.1
10P1.DAT	1045.2	59.8	1024.4

симального количества итераций, числа вершин графа — n и количества муравьев в колонии — m. При поиске вероятности перехода в новую вершину (16) алгоритм возводит количество феромона на ребре τ_{ij} в степень α, и видимость η_{ij} в степень β. Можно заметить, что η_{ij} будет величиной постоянной для двух конкретных i и j вне зависимости от времени. Таким образом, возведение коэффициента η_{ij} в степень β на каждом шаге является действием лишним, увеличивающим общую вычислительную сложность алгоритма. Более удобно за «зрение» муравья принимать величину:

$$\eta_{ij} = (1/D_{ij})^{\beta}, \tag{37}$$

тогда вычисление вероятности примет вид

$$\begin{cases} P_{ij,k}(t) = \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]}{\sum_{l \in J_{i,k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]}}, \\ P_{ij,k}(t) = 0, j \notin J_{i,k}, \end{cases} \tag{38}$$

Выводы

Для задачи 1-1 маршрутизации доставки и вывоза товаров с одним транспортным средством и временными окнами в среде MATLAB R2013b были программно реализованы: алгоритм муравьиной колонии (ACO), алгоритм муравьиной системы (ACS) и алгоритм муравьиной системы с учетом грузовых запросов клиентов *ACS with capacity*. Таблицы с результатами дают основания сделать следующие выводы:

1. Постановка муравьиного алгоритма в виде ACS может быть применима к задаче 1-1 доставки и вывоза товаров с одним транспортным средством и временными окнами.

2. Добавление зависимости выбора клиента от его товарных запросов не очень хорошо адаптировалось в решении задачи 1-1 VRPPDTW.

Заключение

В ходе научно-исследовательской работы была изучена литература по предметной области. Были рассмотрены различные виды задачи маршрутизации, множество методов их решения. Был полностью освоен принцип работы метаэвристических алгоритмов в *NP-полных* задачах. С помощью таких программных продуктов как MATLAB R2013b было реализовано три разновидности муравьиного алгоритма, результаты работы которых были впоследствии сравнены между собой. Получены таблицы, хранящие в себе среднее время работы алгоритмов, их лучшие и усредненные ответы. Уменьшена вычислительная стоимость одного из алгоритмов.

Список литературы

[1] Власова Т. В., Скуднева А. О. Использование встроенных функций прикладных программ для решения задачи маршрутизации вывоза и доставки // Труды 46-й международной научной конференции аспирантов и студентов / науч. ред. тома Н. В. Смирнов. СПб.: Издательский Дом Федоровой Г.В, 2015. Том 2(18) №1. С. 591-596.

[2] Cordeau J. F., Laporte G., Ropke S. Recent models and algorithms for one-to-one pickup and delivery problems // The vehicle routing problem: latest advances and new challenges / ed. by B. Golden, S. Raghavan, E. Wasil. 2008, New York: Springer. P. 327–357.

[3] Gribkovskaia I., Laporte G. One-to-Many-to-One Single Vehicle Pickup and Delivery Problems // Operations Research Computer Science Interfaces, 2008. Series 43. P. 359-377.

[4] Psaraftis H. N. Analysis of an O(N2) Heuristic for the Single Vehicle Many-to-Many Euclidian Dial-A-Ride Prodlem // Transpn Res, 1983. Vol. 17B, No 2. P. 133-145.

[5] Chen H. K., Chou H. W., Hsueh C. F., Yu Y. J. The paired many-to-many pickup and delivery problem: an application // TOP, April 2015. No 23. P. 220–243.

[6] Sahina M., Cavuslara G., Oncanc T., Sahina G., Aksub D. T. An efficient heuristic for the Multi-vehicle One-to-one Pickup and Delivery Problem with Split Loads // Transportation Research Part C: Emerging Technologies, February 2013. Volume 27. P. 169–188.

[7] Cordeau J. F., Laporte G. The dial-a-ride problem: models and algorithms // Annals of Operations Research, September 2007. Volume 153. Issue 1. P. 29–46.

[8] Liub M., Luoa Z., Limc A. A branch-and-cut algorithm for a realistic dial-a-ride problem // Transportation Research Part B: Methodological, November 2015. Volume 81, Part 1. P. 267–288.

[9] Fan J. The Vehicle Routing Problem with Simultaneous Pickup and Delivery Based on Customer Satisfaction // Procedia Engineering, 2011. Volume 15. P. 5284-5289.

[10] Wassan N. A., Nagy G. Vehicle Routing Problem with Deliveries and Pickups: Modelling Issues and Meta-heuristics Solution Approaches // International Journal of Transportation, 2014. Vol.2, No 1. P. 95-110.

[11] Chen J. F., Wu T. H Vehicle Routing Problem with Simultaneous Deliveries and Pickups // The Journal of the

Operational Research Society, 2006. Vol. 57, No 5. P. 579-587.

[12] Ropke S., Pisinger D. A unified heuristic for a large class of Vehicle Routing Problems with Backhauls // European Journal of Operational Research, 2006. Volume 171, Issue 3. P. 750–775.

[13] Gendreau M., Hertz A., Laporte G. The traveling salesman problem with backhauls. // Computers and Operations Research, 1996. No 23. P. 501–508.

[14] Gendreau M., Laporte G., Vigo D. Heuristics for the traveling salesman problem with pickup and delivery. // Computers and Operations Research, 1999. No 26. P. 699–714.

[15] Mladenovi'c N., Hansen P. Variable neighborhood search. // Computers and Operations Research, 1997. No 24. P. 1097–1100.

[16] Abdulkader M., Gajpal Y., ElMekkawy T. Hybridized ant colony algorithm for the Multi Compartment Vehicle // Applied Soft Computing, 2015. No 37. P. 196–203.

[17] Rizzoli A. E., Oliverio F., Montemanni R., Gambardella L. M. Ant Colony Optimization for vehicle problems from theory to applications 2008. 50 p.

[18] Liao L., Cai X., Huang H., Liu Y. Vehicle Routing with Time Windows Based on Two-stage Optimization Algorithm // 28th Chinese Control and Decision Conference (CCDC), 2016. P. 4741-4745.

[19] Afshar-Nadjafi B., Afshar-Nadjafi A. A constructive heuristic for time-dependent multi-depot vehicle routing problem with time-windows and heterogeneous fleet // Journal of King Saud University – Engineering Sciences, 2017. No 29, P. 29–34.

[20] Xu S. H., Liu J. P., Zhang F. H., Wang L., Sun L. J. A Combination of Genetic Algorithm and Particle Swarm Optimization for Vehicle Routing Problem with Time Windows // Sensors, 2015. No 15. P. 21033-21053.

[21] Zhang T., Art W., Zhang Y. Integrated Ant Colony and Tabu Search approach for time dependent vehicle routing problems with simultaneous pickup and delivery // Springer Science+Business Media New York, 2014. No 28. P. 288–309.

[22] Sayyah M., Larki H., Yousefikhoshbakht M. Solving the Vehicle Routing Problem with Simultaneous Pickup and Delivery by an Effective Ant Colony Optimization // Journal of Industrial Engineering and Management Studies (JIEMS), 2016. Vol. 3, No 1. P. 16-38.

[23] Alshamrani A., Mathur K., Ballou R. H. Reverse logistics: Simultaneous design of delivery routes and return strategies. // Computers and Operations Research, 2007. No 34. P. 595–619.

[24] Mitrovi'c-Mini'c S. and Laporte G. The pickup and delivery problem with time windows and transshipment. // INFOR, 2006. No 44. P. 217–227.

[25] Ruland K. S., Rodin E. Y. The Pickup and Delivery Problem: Faces and Branch-and-Cut Algorithm // Computers Math. Applic., 1997. Vol. 33, No 12. P. 1-13.

[26] Lau H. C., Liang Z. Pickup and Delivery with Time Windows: Algorithms and Test Case Generation //International Journal on Artificial Intelligence Tools, 2002. No 11. P. 455-472.

[27] Hosny M. I. Investigating Heuristic and Meta-Heuristic Algorithms for Solving Pickup and Delivery Problems 2010. 266 p.

[28] Pillac V., Gendreau M., Gu'eret C., Medaglia A. A review of dynamic vehicle routing problems // European Journal of Operational Research, 2012. P. 1-35.

[29] Corominas A., Garcia-Villoria A., Pastor R. Improving parametric Clarke and Wright algorithms by means of iterative

empirically adjusted greedy heuristics // SORT, January-June 2014. No 1. P. 3-12.

[30] Desrochers M., Verhoog T. W. A Matching Based Savings Algorithm for the Vehicle Routing Problem // GERAD, 1989. No 4. P. 1-19.

[31] Ferdinand F. N., Kim K. H., Koh S. G., Ko C. S. A Genetic Algorithm Based Heuristic for the Design of Pick-up and Delivery Routes for Strategic Alliance in Express Delivery Services // IFAC Proceedings, 2013. Volume 46, Issue 9. P. 1938-1943.

[32] Cheng L. A Genetic Algorithm for the Vehicle Routing Problem with Time Windows 2005. 33 p.

[33] Bakr M. A., Hedar A. R. Three Strategies Tabu Search for Vehicle Routing Problem with Time Windows // Computer Science and Information Technology, 2014. No 2. P. 108-119.

[34] Fu Z., Eglese R., Li L. A Unified Tabu Search Algorithm for Vehicle Routing Problems with Soft Time Windows // Journal of the Operational Research Society, May 2008. No 59(5). P. 663-673.

[35] Gendreau M., Iori M., Laporte G., Martello S. A Tabu Search heuristic forthe vehicle routing problem with two-dimensional loading constraints 2005. 25 p.

[36] Yao X. A New Simulated Annealing Algorithm // International Journal of Computer Mathematics, 1995. No 56. P. 161-168.

[37] Szu H. H., Hartley R. L. Fast Simulated Annealing // Physical Letters, 1987. No 122. P. 157-162.

[38] Colorni A., Dorigo M., Maniezzo V. Distributed Optimization by Ant Colonies // actes de la premiere conference europeenne sur la vie artificielle, Paris, France, Elsevier Publishing, 1991. P. 134-142.

[39] Ермолаев С. Ю. Муравьиные алгоритмы оптимизации // Инфокоммуникационные технологии, 2008. Том 6. 1. С. 23-29.

[40] Applying the Ant System to the Vehicle Routing Problem // Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization,1999. Chapter 20. P.285-296.

[41] Gajpal Y., Abad P. L. An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup //

Computers and Operations Research, 2009. No 36, Vol 12. P. 3215-3223.

[42] Catay B. Ant Colony Optimization and Its Application to the Vehicle Routing Problem with Pickups and Deliveries // Natural Intelligence for Scheduling, Planning and Packing Problems. Volume 250. P. 219-244.

[43] Gambardella L. M. Coupling Ant Colony System with Local Search 2014-2015. 178 p.

Приложение

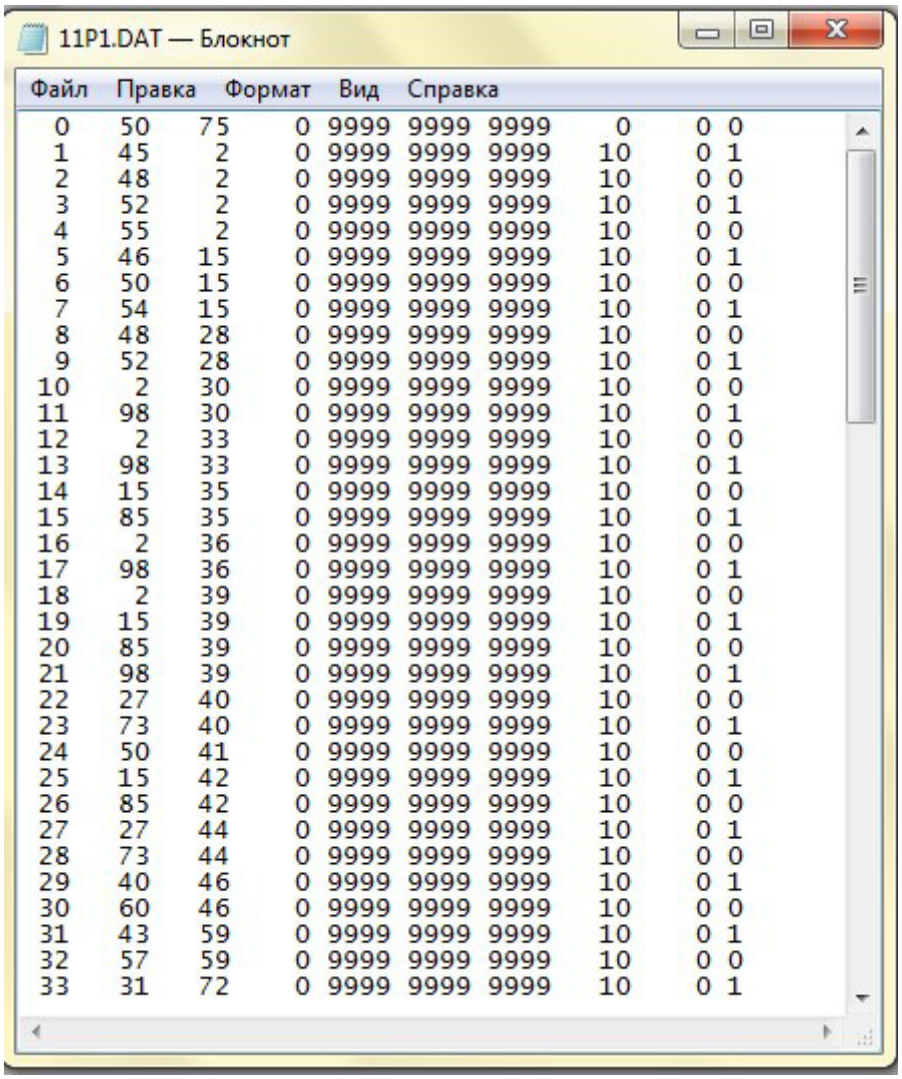
Приложение 1. Формат текстового файла, содержащего начальные данные

README.P1 — Блокнот

Файл Правка Формат Вид Справка

```
SET P1

60 VRP instances
Nr. of Depots: 1
Nr. of Stops: 100
Demand of stop: q=10
Mixed pickup and delivery

Nr. of Vehicles: unlimited
Capacity of Vehicles:Q=100

File structure

    0   50   50    0 9999 9999 9999    0    0   0  (depot)
    1    5    5    0 9999 9999 9999   10    0   0  (stop 1)
         ....                                  ...
  100   95   95    0 9999 9999 9999   10    0   0  (stop 100)
    |    |    |    |    |    |    |    |    |   |
    |    |    |    |    |    |    |    |    |   delivery=0 / pickup=1
    |    |    |    |    |    |    |    |    service time
    |    |    |    |    |    |    |    demand
    |    |    |    |    |    |    closing time TW2
    |    |    |    |    |    opening time TW2
    |    |    |    |    closing time TW1
    |    |    |    opening time TW1
    |    |    Y coordinate
    |    X coordinate
    stop nr.
```

Приложение 2. Пример данных из тестовой выборки



11P1.DAT — Блокнот

Файл Правка Формат Вид Справка

0	50	75	0	9999	9999	9999	0	0	0
1	45	2	0	9999	9999	9999	10	0	1
2	48	2	0	9999	9999	9999	10	0	0
3	52	2	0	9999	9999	9999	10	0	1
4	55	2	0	9999	9999	9999	10	0	0
5	46	15	0	9999	9999	9999	10	0	1
6	50	15	0	9999	9999	9999	10	0	0
7	54	15	0	9999	9999	9999	10	0	1
8	48	28	0	9999	9999	9999	10	0	0
9	52	28	0	9999	9999	9999	10	0	1
10	2	30	0	9999	9999	9999	10	0	0
11	98	30	0	9999	9999	9999	10	0	1
12	2	33	0	9999	9999	9999	10	0	0
13	98	33	0	9999	9999	9999	10	0	1
14	15	35	0	9999	9999	9999	10	0	0
15	85	35	0	9999	9999	9999	10	0	1
16	2	36	0	9999	9999	9999	10	0	0
17	98	36	0	9999	9999	9999	10	0	1
18	2	39	0	9999	9999	9999	10	0	0
19	15	39	0	9999	9999	9999	10	0	1
20	85	39	0	9999	9999	9999	10	0	0
21	98	39	0	9999	9999	9999	10	0	1
22	27	40	0	9999	9999	9999	10	0	0
23	73	40	0	9999	9999	9999	10	0	1
24	50	41	0	9999	9999	9999	10	0	0
25	15	42	0	9999	9999	9999	10	0	1
26	85	42	0	9999	9999	9999	10	0	0
27	27	44	0	9999	9999	9999	10	0	1
28	73	44	0	9999	9999	9999	10	0	0
29	40	46	0	9999	9999	9999	10	0	1
30	60	46	0	9999	9999	9999	10	0	0
31	43	59	0	9999	9999	9999	10	0	1
32	57	59	0	9999	9999	9999	10	0	0
33	31	72	0	9999	9999	9999	10	0	1

Приложение 3. Головная функция алгоритма

```
1 function [L_opt] = Test()
2 tic
3 A = dlmread('10P1.dat');
4 [X, Y, E, L, q, post, potr, depot] = readfile(A)
      ;
5 [tmax, N, number_of_ants, capacity, v,
```

```
    vehicle_load, a, b, e, Q, fer0, q0] =
    SetParametrs(X);
6
7
8 tau = ones(N) * fer0;
9 D = zeros(N);
10 eta = zeros(N);
11 opt_route = ones(N + 1, 1);
12 L_opt = inf;
13 ver = zeros(1, N);
14
15 routes_of_one_colony = zeros(N, number_of_ants);
16 distances_of_one_colony = zeros(number_of_ants,
    1);
17 best_distances_of_each_colony = zeros(tmax, 1);
18 best_routes_of_each_colony = zeros(N + 1, tmax);
19
20 for i = 1 : N
21     for j = 1 : N
22         if i ~= j
23             D(i,j) = sqrt((X(i) - X(j))^2 + (Y(i
                ) - Y(j))^2);
```

```
24                eta(i,j) = 1/D(i,j)^b;
25            else
26                D(i,j) = 0;
27                eta(i,j) = inf;
28            end
29        end
30    end
31
32    for iteration = 1 : tmax
33        for k = 1 : number_of_ants
34            job_list = zeros(1, ((N - length(depot))
                  /2))
35            for i = 1 : length(potr)
36                job_list(i) = potr(i);
37            end
38            routes_of_one_colony(1, k) = 1;
39            for i = 2 : N
40                get_point = ACOmain(i, k,
                      routes_of_one_colony, tau, eta, a,
                       b, vehicle_load, potr, job_list,
                      post, capacity, q0);
41                routes_of_one_colony(i, k) =
```

```
                get_point;
42              index = find(post == get_point, 1, '
                  first');
43              if (~isempty(index))
44                  vehicle_load = vehicle_load + q(
                      get_point);
45                  index1 = potr(index);
46                  index2 = find(job_list == index1
                      , 1, 'first');
47                  job_list(index2) = [];
48              else
49                  vehicle_load = vehicle_load - q(
                      get_point);
50              end
51          end
52          tau = LocalUpdate(e, k, tau, N, fer0,
                routes_of_one_colony);
53      end
54      [tau, L_opt, opt_route,
          best_routes_of_each_colony,
          best_distances_of_each_colony] =
          GlobalUpdate(tau, Q, number_of_ants,...
```

```
55        routes_of_one_colony,
            distances_of_one_colony, iteration,
            opt_route, ...
56        L_opt, best_routes_of_each_colony,
            best_distances_of_each_colony, e, N, D);
57 end
58 transpose(opt_route);
59 sprintf('%17.15f',L_opt);
60 MarshrutX = X(opt_route(:));
61 MarshrutY = Y(opt_route(:));
62 plot([MarshrutX(:); MarshrutX(1)],[MarshrutY(:);
       MarshrutY(1)],'.r-');
63 clearvars -except L_opt opt_route
     best_routes_of_each_colony
     best_distances_of_each_colony
64 toc
```

Приложение 4. Функция чтения файла с тестовыми данными для алгоритма

```
1 function [X, Y, E, L, q, post, potr, depot] =
     readfile(File)
2 A = File;
```

```
3 X = transp(A(1:length(A),2));
4 Y = transp(A(1:length(A),3));
5 E = transp(A(1:length(A),4));
6 L = transp(A(1:length(A),5));
7 q = transp(A(1:length(A),8));
8 post = [];
9 potr = [];
10 depot = [];
11 loc = size(A);
12 N = loc(1);
13 for i = 1 : N
14     if ((A(i, 8) ~= 0) && ((A(i, 10)) == 1))
15         post =  [post, i];
16     else if ((A(i, 8) ~= 0) && ((A(i, 10)) == 0)
       )
17         potr =  [potr, i];
18         else if (A(i, 8) == 0)
19                 depot = [depot, i];
20             end
21         end
22     end
23 end
```

Приложение 5. Функция инициализации параметров алгоритма

```
function [tmax, N, number_of_ants, capacity, v,
    vehicle_load, a, b, e, Q, fer0, q0] =
    SetParametrs(X)
tmax = 50;
N = length(X);
number_of_ants = N;
capacity = 100;
vehicle_load = 0;
v = 40;
a = 1;
b = 2;
e = 0.5;
fer0 = 0.1;
q0 = 0.9;
```

Приложение 6. Функция выбора следующего клиента

```
function [tow] = ACOmain(i, k, ant_route,
    matrica_fer, eta, a, b, vehicle_load, potr,
    job_list, post, capacity, q0)
 if i == 2
```

```
3       random = randi(100);
4       tow = random + 1;
5   else
6       ind = ant_route(i - 1, k);
7       ver = matrica_fer(ind, :).^a .* eta(ind, :);
8       ver(ant_route(1 : i - 1, k)) = 0;
9           ver(1) = 0;
10      if (vehicle_load == 0)
11          for key = potr
12              ver(key) = 0;
13          end
14      end
15      if (~isempty(job_list))
16          for key = job_list
17              ver(key) = 0;
18          end
19      end
20
21      if (vehicle_load == capacity)
22          for key = post
23              ver(key) = 0;
24          end
```

```
25      end
26      ver = ver ./ sum(ver);
27      random = rand;
28      if rand < q0
29          Pmax = max(ver);
30          tow = find(ver == Pmax, 1,'first');
31      else
32      tow = find(cumsum(ver) >= random, 1, 'first'
            );
33      end
34  end
```

Приложение 7. Функция локального обновления феромона

```
1 function [fer] = LocalUpdate(e, k, fer, N, fer0,
      ant_route)
2      route = [transpose(ant_route(1 : end, k)),
          ant_route(1, k)];
3      for t = 1 : N
4          x = route(t);
5          y = route(t + 1);
6          fer(x, y) = fer(x, y) * (1 - e) + e *
```

```
        fer0;
7       end
```

Приложение 8. Функция глобального обновления феромона

```
1 function [tau, L_opt, opt_road,
    best_routes_of_each_colony,
    best_distances_of_each_colony] = GlobalUpdate(
    tau, Q, number_of_ants,...
2       routes_of_one_colony,
          distances_of_one_colony, iteration,
          opt_road, ...
3       L_opt, best_routes_of_each_colony,
          best_distances_of_each_colony, e, N, D)
4 tau = (1 - e) * tau;
5 for j = 1 : number_of_ants
6       all_route_of_one_ant = [routes_of_one_colony
          (1 : end, j); routes_of_one_colony(1, j)];
7       S = 0;
8           for i = 1 : N
9               S = S + D(all_route_of_one_ant(i),
                    all_route_of_one_ant(i+1));
10          end
```

```
11            distances_of_one_colony(j) = S;
12 end
13 mindist = min(distances_of_one_colony);
14 ind = find(distances_of_one_colony == mindist,1,
    'first');
15 best_distances_of_each_colony(iteration, 1) =
    mindist;
16 best_routes_of_each_colony(:, iteration) = [
    routes_of_one_colony(1 : end, ind);
    routes_of_one_colony(1, ind)];
17 if (mindist < L_opt)
18     for t = 1 : N
19     x = best_routes_of_each_colony(t, iteration)
         ;
20     y = best_routes_of_each_colony(t + 1,
         iteration);
21     tau(x, y) = tau(x, y) + Q / mindist;
22     end
23     L_opt = mindist;
24     opt_road = [routes_of_one_colony(1 : end,
         ind);routes_of_one_colony(1, ind)];
25 end
```