САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ ЭНЕРГЕТИЧЕСКИХ СИСТЕМ

Землина Мария Анатольевна

Магистерская диссертация

Одна задача двухуровневой маршрутизации для управления транспортными потоками

Направление 01.04.02

Прикладная математика и информатика

Магистерская программа:

"Математическое и информационное обеспечение экономической деятельности"

Научный руководитель,

кандидат физ.-мат. наук,

доцент

Губар Е. А.

Санкт-Петербург

2017

Содержание

1. Введение

В данной работе предложено решение задачи оптимизации двухуровневой транспортной системы. Двухуровневая система применяется преимущественно в крупных городах и основана на так называемой стратегии объединенного распределения, которая использует два уровня площадок складирования и разнородные парки транспортных средств. Такая система реализуется в логистике многих компаний.

Первый уровень связан с крупным грузовым транспортом, относящимся к распределительным центрам (city distribution centres, CDCs) на окраинах города – внешним зонам. Крупногабаритные транспортные средства поставляют продукцию из этого уровня во второй (внутренний) – депо и промежуточные станции, стратегически расположенные в городе. Транспортировка грузов с них и последующая доставка непосредственно заказчикам осуществляется городскими малогабаритными транспортными средствами.

Двухуровневая маршрутизация актуальна, так как ее применение обеспечивает:

1. избежание присутствия крупного транспорта в центре города;
2. продуктивность цепи поставок;
3. надежный сервис клиентов;
4. уменьшение вредных воздействий на окружающую среду: загрязнения воздуха, потребления топлива и сильной загруженности транспортных сетей.

Данная работа организована следующим образом:

- В главе 2 представлен обзор литературы по рассматриваемой задаче.
- В главе 3 сформулирована математическая постановка задачи.
- В главе 4 рассмотрены методы решения проблемы с подробным описанием алгоритмов.
- В главе 5 показаны результаты проверки на тестовых примерах разработанного комплексного подхода.
- В главе 6 перечислены основные выводы по проделанной работе.

2. Обзор литературы

Двухуровневые проблемы маршрутизации подразделяются на три класса [1]:

1. Двухуровневая задача размещения и маршрутизации (Two-echelon location routing problem, 2E-LRP) связана со стратегическими и тактическими решениями. Товар, находящийся в различных депо или платформах, необходимо доставить до соответствующих пунктов/клиентов посредством сети промежуточных станций. Каждому депо и промежуточной станции соответствует цена открытия. Для открытия депо или промежуточных станций, сначала необходимо выбрать их местоположение из доступного множества. Данный класс задач представлен в работах [2], [3], [4] и [5].

2. Двухуровневая задача маршрутизации транспортных средств (Two-echelon vehicle routing problem, 2E-VRP) связана только с тактическими решениями. Здесь уже определено множество депо и промежуточных станций. Цены открытия равны нулю. Наилучший эвристический алгоритм предложен в статье [6], а наилучший точный алгоритм в статье [7].

3. В третьем классе (Truck and trailer routing problem, TTRP) перевозка товара осуществляется с помощью грузовых машин и трейлеров. Одно подмножество клиентов должно быть обслужено только грузовыми машинами, к остальным клиентам могут подъезжать как полное транспортное средство (грузовик, тянущий трейлер), так и просто грузовая машина. В данном случае маршрут полного транспортного средства относится к первому уровню маршрутизации, а маршрут, проходимый грузовой машиной без прицепа, составляет второй уровень системы. Третий класс задач представлен в работах [8], [9], [10], [11], [12] и [13].

Таким образом, в проблеме с тактическими решениями необходимо решить только задачу маршрутизации. В случае стратегических задач необходимо также выбрать оптимальные месторасположения депо и/или промежуточных станций из доступного множества.

Обычно проблема разбивается на две части (по уровням) и затем решается при помощи точных, эвристических или гибридных алгоритмов. Однако данный класс задач относится к NP-трудным и поэтому преимущественно решение находится при помощи эвристических алгоритмов.

Далее представлены существующие методы решения проблем двухуровневой маршрутизации:

1. По типу оптимизацинной модели:
   - Целочисленное линейное программирование;
   - Смешанное целочисленное линейное программирование.
2. По типу алгоритма:
   - Adaptive Large Neighborhood Search (ALNS) — адаптивный расширенный локальный поиск;
   - Branch-and-cut (B&C) — метод ветвей и отсечений;
   - Branch-and-price (B&P) — метод ветвей и цен;
   - Greedy randomized adaptive search procedure (GRASP) — вероятностный жадный алгоритм поиска;
   - Iterated local search (ILS) — итеративный локальный поиск;
   - Simulated annealing (SA) — имитация отжига;
   - Tabu search (TS) — поиск с запретами;

- Variable neighborhood search (VNS) — переменный поиск ближайшего соседа;
- Генетические алгоритмы;
- Метод роя частиц;
- Муравьиный алгоритм.

Самым эффективным в настоящее время является метод Adaptive large neighborhood search (ALNS), представленный в работе Contardo C., Crainic T.G. и Hemmelmayr V. [2].

Основная идея алгоритма заключается в исключении на каждой итерации подмножества клиентов из имеющегося решения с помощью разрушающего оператора и, затем, в обратном их включении в решение на другие позиции, используя оператор восстановления. Каждому оператору назначается рейтинг, который определяется функцией распределения вероятностей, построенной на результатах предыдущего применения этого оператора. Другими словами, оператор, нашедший несколько улучшающих решений, имеет больший рейтинг, чем другие операторы, и, таким образом, большую вероятность быть выбранным.

3. Постановка задачи

В данной работе рассматривается двухуровневая задача размещения и маршрутизации (2E-VRP), включающая в себя решение как тактических задач (маршрутизация), так и стратегических – определение множества промежуточных станций из имеющихся.

2E-VRP может быть определена на ориентированном графе $G = (V, A)$, где V – множество вершин, A – множество дуг (пар вершин). Множество V состоит из трех подмножеств вершин:

1. вершина депо $V_0 = \{v_0\}$,
2. подмножество V_s, содержащее m промежуточных станций,
3. подмножество V_c из n клиентов.

Стоимость перемещения из вершины i в вершину j обозначим как c_{ij}. Каждый клиент i имеет спрос d_i, который не может быть разделен на партии, то есть весь заказ конкретного клиента доставляет одна машина за единственное посещение. Заказ не может быть доставлен прямо из депо, сначала он должен быть собран в промежуточной станции. Поставки в промежуточные станции в рамках первого уровня могут быть разбиты по грузовым машинам. Также необходимо учитывать максимальную емкость транспортного средства. Эта емкость одинакова для всех машин одного уровня, но может отличаться для каждого уровня. Емкости машин первого и второго уровня обозначаются K_1 и K_2 соответственно. Количество имеющихся машин первого уровня — m_1, а второго уровня — m_2.

На рис. 1 показано одно из решений 2E-VRP, рисунок предложен в работе [6]. Депо представлено в виде квадрата, промежуточные станции в виде

крупных точек, а клиенты обозначены мелкими точками. Маршруты из депо в промежуточные станции принадлежат первому уровню и представлены в виде пунктирных линий. Маршрутами второго уровня называются маршруты транспортных средств, начинающиеся в промежуточных станциях и обслуживающие клиентов.

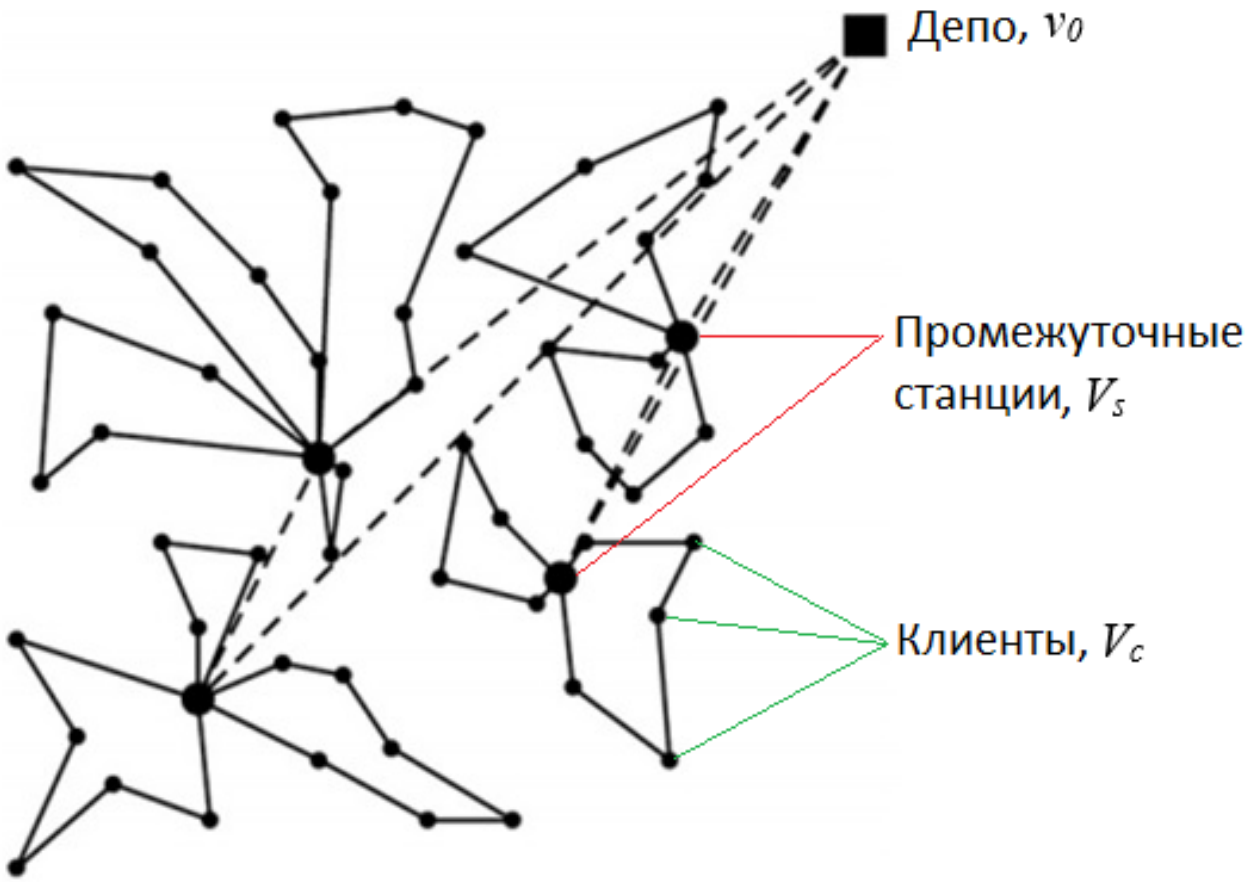

Рис. 1. Решение 2E-VRP.

Спрос промежуточной станции определяется суммарным спросом прикрепленных к ней клиентов. Таким образом, любое изменение в распределении клиентов между станциями влияет на маршрутизацию первого уровня, и некогда оптимальное решение VRP первого уровня может стать не только неоптимальным, но и вообще недопустимым.

Для решения стратегической задачи смоделируем LRP в виде 2E-VRP. В задаче LRP есть множество V_s из m всевозможных депо, где для каждого депо определена своя емкость W_j и стоимость открытия O_j, и множество V_c из n клиентов с заданным спросом. Необходимо открыть подмножество депо и построить маршруты так, чтобы удовлетворить спрос всех клиентов, учесть ограничения на емкость депо и грузовых машин, и минимизировать затраты на открытие депо и затраты, связанные с транспортировкой.

Чтобы смоделировать LRP в виде 2E-VRP, представим v_0 как фиктивную вершину. Тогда множеству возможных депо соответствуют промежуточные станции. Главная разница заключается в том, что сейчас первый уровень состоит только из маршрутов к единственному клиенту (станции) и стоимость открытия станции характеризуется стоимостью маршрута из v_0 в эту станцию.

Итак, для построения математической модели (взята из [14]) введем следующие обозначения:

$V_0 = \{v_0\}$ – депо;

V_s – множество промежуточных станций;

V_c – множество покупателей;

n_s – количество промежуточных станций;

n_c – количество покупателей;

m_1 – количество машин первого уровня;

m_2 – количество машин второго уровня;

m_{s_k} – максимальное количество машин второго уровня для промежуточной станции k;

K_1 – грузоподъемность машин первого уровня;

K_2 – грузоподъемность машин второго уровня;

d_i – спрос покупателя i;

c_{ij} – длина дуги (i, j);

Q^1_{ij} – количество перевозимого груза по дуге (i, j) первого уровня;

Q^2_{ijk} – количество перевозимого груза по дуге (i, j) второго уровня из промежуточной станции k;

x_{ij} – бинарная переменная, равна 1, если дуга (i, j) включена в маршрут первого уровня;

y^k_{ij} – бинарная переменная, равна 1, если дуга (i, j) включена в маршрут второго уровня из промежуточной станции k;

z_{kj} – бинарная переменная, равная 1, если покупатель c_i обслуживается промежуточной станцией k;

$D_k = \sum_{j \in V_c} d_j z_{kj}, \forall k \in V_s$ – суммарный спрос, консолидируемый промежуточной станцией k.

Тогда целевая функция, которую необходимо минимизировать, выглядит следующим образом:

$$\min \sum_{i,j \in V_0 \cup V_s, i \neq j} c_{ij} x_{ij} + \sum_{k \in V_s} \sum_{i,j \in V_s \cup V_c, i \neq j} c_{ij} y^k_{ij}.$$

Ограничения:

$$\sum_{i \in V_s} x_{0i} \leqslant m_1; \tag{1}$$

$$\sum_{j \in V_s \cup V_0, j \neq k} x_{jk} = \sum_{i \in V_s \cup V_0, i \neq k} x_{ki}, \quad \forall k \in V_s \cup V_0; \tag{2}$$

$$\sum_{k \in V_s} \sum_{j \in V_c} y_{kj}^k \leqslant m_2; \tag{3}$$

$$\sum_{j \in V_c} y_{kj}^k \leqslant m_{s_k}, \quad \forall k \in V_s; \tag{4}$$

$$\sum_{j \in V_c} y_{kj}^k = \sum_{j \in V_c} y_{jk}^k, \quad \forall k \in V_s; \tag{5}$$

$$\sum_{i \in V_s \cup v_0, i \neq j} Q_{ij}^1 - \sum_{i \in V_s \cup v_0, i \neq j} Q_{ji}^1 =$$

$$= \begin{cases} D_j & \text{, если } j \text{ не депо} \\ \sum_{i \in V_c} -d_i & \text{иначе} \end{cases} \quad \forall j \in V_s \cup V_0; \tag{6}$$

$$\sum_{i \in V_c \cup k, i \neq j} Q_{ijk}^2 - \sum_{i \in V_c \cup k, i \neq j} Q_{jik}^2 =$$

$$= \begin{cases} z_{kj} d_j & \text{, если } j \text{ не промежуточная станция} \\ -D_j & \text{иначе} \end{cases} \tag{7}$$

$$\forall j \in V_c \cup V_s, \forall k \in V_s;$$

$$Q_{ij}^1 \leqslant K^1 x_{ij}, \quad \forall i, j \in V_s \cup V_0, i \neq j; \tag{8}$$

$$Q_{ijk}^2 \leqslant K^2 y_{ij}^k, \quad \forall i, j \in V_s \cup V_c, i \neq j, \forall k \in V_s; \tag{9}$$

$$\sum_{i \in V_s} Q^1_{iv_0} = 0; \tag{10}$$

$$\sum_{j \in V_c} Q^2_{jkk} = 0, \quad \forall k \in V_s; \tag{11}$$

$$y^k_{ij} \leqslant z_{kj}, \quad \forall i \in V_s \cup V_c, \forall j \in V_c, \forall k \in V_s; \tag{12}$$

$$y^k_{ji} \leqslant z_{kj}, \quad \forall i \in V_s, \forall j \in V_c, \forall k \in V_s; \tag{13}$$

$$\sum_{i \in V_s \cup V_c} y^k_{ij} = z_{kj}, \quad \forall j \in V_c, \forall k \in V_s; \tag{14}$$

$$\sum_{i \in V_s} y^k_{ji} = z_{kj}, \quad \forall j \in V_c, \forall k \in V_s; \tag{15}$$

$$\sum_{i \in V_s} z_{ij} = 1, \quad \forall j \in V_c; \tag{16}$$

$$y^k_{ki} \leqslant \sum_{l \in V_s \cup V_0} x_{kl}, \quad \forall j \in V_c, \forall k \in V_s; \tag{17}$$

$$y^k_{ij} \in \{0, 1\},\ z_{kj} \in \{0, 1\}, \quad \forall i, j \in V_c, \forall k \in V_s; \tag{18}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V_0 \cup V_s; \tag{19}$$

$$Q^1_{ij} \geq 0,\ \forall i, j \in V_0 \cup V_s,\ Q^2_{ijk} \geq 0,\ \forall i, j \in V_s \cup V_c, \forall k \in V_s. \tag{20}$$

Целевая функция минимизирует сумму длин всех маршрутов первого и второго уровня. Ограничение (2) показывает, для $k = v_0$, что каждый маршрут первого уровня начинается и заканчивается в депо. Если же k – это промежуточная станция, то подразумевается необходимость баланса между въезжающим в эту станцию транспортом и выезжающим. (4) — ограничения на вместительность промежуточной станции. То есть ограничения на максимальное количество маршрутов, начинающихся в каждой станции, указывают максимальную грузовместительность станций. Согласно выражению (5) каждый маршрут второго уровня начинается и заканчивается в одной и той же промежуточной станции. Количество маршрутов в каждом уровне не должно превышать допустимое количество транспортных средств для соответствующего уровня, как и говорится в выражениях (1) и (3).

Выражения (6) и (7) показывают, что количество выгружаемого товара в каждой точке равно величине спроса в соответствующей точке. За исключением двух случаев:

- если в выражении (6) для первого уровня $j = v_0$ – депо, то исходящий поток грузов равен общему спросу всех клиентов;
- если в выражении (7) для второго уровня $j \in V_s$ – промежуточная станция, то входящий поток равен нулю, а исходящий поток равен сумме спросов всех клиентов, обслуживаемых данной станцией.

Более того, выражения (6) и (7) запрещают существование подмаршрутов, несодержащих депо или промежуточную станцию соответственно уровню. Фактически, каждая точка получает количество товаров, соответствующее её спросу, предотвращая наличие подмаршрутов. Рассмотрим, например, случай, когда существует подмаршрут первого уровня между точками i, j и

k. Легко проверить, что в данном случае нет такого значения переменных Q^1_{ij}, Q^1_{jk} и Q^1_{ki}, удовлетворяющего ограничениям (6) и (7). Ограничения на вместительность первого и второго уровня сформулированы в выражениях (8) и (9) соответственно. Выражения (10) и (11) запрещают маршруты с грузовыми излишками/остатками на последнем промежутке, то есть обратно в депо или промежуточные станции машины едут пустыми.

Ограничения (12) и (13) говорят о том, что клиент j обслуживается промежуточной станцией k ($z_{kj} = 1$), только если он получает груз от этой станции ($y^k_{ij} = 1$). Ограничение (16) приписывает каждого клиента к одной и только одной станции. Ограничения (14) и (15) показывают, что через каждого клиента проходит только один маршрут. В то же время они подразумевают, что маршрут второго уровня начинается в промежуточной станции k и обслуживает конкретного клиента, только если клиент приписан к данной станции. Ограничение (17) позволяет маршруту второго уровня начаться в промежуточной станции k, только если она уже была обслужена маршрутом первого уровня.

4. Методы решения

Были рассмотрены различные алгоритмы для реализации поставленной задачи. Как говорилось ранее, наилучшее решение для 2E-VRP на данный момент можно получить с помощью применения алгоритма ALNS (Adaptive large neighborhood search). Это реализовали Hemmelmayr V.C., Cordeau J.F. и Crainic T.G. и описали в статье 2012 года, опубликованной в Computers & Operations Research [6].

В данной работе был разработан и протестирован видоизмененный алгоритм ALNS. Для основы взят комплексный подход, предложенный в статье [6], который состоит из подразделов:

1. Распределение клиентов по промежуточным станциям при помощи механизма колеса рулетки.
2. Нахождение начального решения с помощью алгоритма savings.
3. Применение алгоритма ALNS в сочетании с локальным поиском к начальному решению.

Изменения в комплексном подходе, предложенном в статье [6], которые реализованы в данной выпускной работе:

1. Начальное решение находится иным способом: savings алгоритм заменен на гибридный генетический алгоритм, представленный в статье Chang Y. и Chen L. [15].
2. Вместо локального поиска, применяемого после каждого оператора восстановления, было использовано сочетание алгоритма splitting и алгоритма имитации отжига.

3. Исключен из реализации оператор разрушения Route Redistribution, так как он удаляет большую часть маршрутов и приводит решение в начальные условия.

4. Исключен из реализации оператор восстановления Greedy Insertion Forbidden, так как он практически не отличается от оператора Greedy Insertion и, как показали в своей статье [6] Hemmelmayr V.C., Cordeau J. F. и Crainic T.G., решение этот оператор существенно не улучшает.

5. Исключен оператор первого уровня Satellite Swap, который также, по статистике, не улучшает решение.

Рассмотрим подробнее алгоритмы ALNS, гибридный генетический и имитации отжига.

4.1. Адаптивный расширенный локальный поиск

Как и в статье [6] основная идея алгоритма заключается в исключении на каждой итерации q клиентов из имеющегося решения с помощью разрушающего оператора, помещении их в пул клиентов и, затем, в обратном их включении, используя оператор восстановления. Некоторые используемые операторы явным образом открывают или закрывают промежуточные станции, в то время как другие влияют на более ограниченную область поиска, то есть, исключая небольшое число клиентов и сохраняя имеющуюся конфигурацию промежуточных станций неизменной. Операторы выбираются с помощью механизма колеса рулетки в зависимости от степени успешности их предыдущих применений. Каждому оператору назначается рейтинг. Если оператор находит новое улучшающее решение, то его рейтинг возрастает, и, таким образом, вероятность быть выбранным также увеличивается.

ALNS впервые разработали Ropke S. и Pisinger D. [16] для решения проблемы вывоза и доставки. Позднее, эти же авторы решили несколько вариаций задачи маршрутизации транспорта (VRP), используя алгоритм, основанный на той же методологии [17]. По сравнению с этим методом в данной работе реализован упрощенный механизм для пересчета рейтинга, а критерий принятия решения не имеет параметров. К тому же в этой работе после некоторых из операторов применяется комбинация алгоритма splitting и алгоритма имитации отжига.

Сам алгоритм ALNS работает следующим образом (рис. 2). Найдя начальное решение, применяем оператор разрушения, удаляющий q клиентов, и затем применяем оператор восстановления для возвращения этих точек в решение. Так как мы имеем дело с двухуровневой проблемой, мы используем

иерархическую структуру операторов разрушения. Разрушающие операторы разделяются на два класса: одни изменяют конфигурацию имеющегося решения путем открытия или закрытия промежуточных станций, а другие влияют на более ограниченную область поиска. Далее, разрушающие операторы, оказывающие значительные изменения (операторы первого уровня), обозначим D_L, а разрушающие операторы с меньшим масштабом влияния (операторы второго уровня) — D_S. В свою очередь, R — множество операторов восстановления. Разрушающие операторы первого уровня используются каждый раз, когда в течение w итераций не происходит никакого улучшения системы. Новое решение, полученное в результате использования какого-либо из этих операторов, дальше улучшается с помощью имитации отжига. Более того, оно принимается как новое локальное решение, даже если оно хуже предыдущего локального решения, то есть оно получает свободный проход через стадию согласования принятия текущего решения. Решения, полученные в результате применения операторов из множества D_S, проходят дальше в фазу имитации отжига, только если они отличаются от глобального решения s^* (наилучшего найденного), не более чем на θ процентов. И такое решение принимается как новое локальное решение, только когда оно имеет меньшее значение целевой функции, чем предыдущее локальное решение.

Введение разделения операторов разрушения по масштабу влияния необходимо по следующим причинам. Операторы из множества D_L сильно изменяют решение, особенно на первом уровне. Операторы восстановления созданы для того, чтобы возвращать ограниченное количество клиентов в частичное решение. Таким образом, операторы разрушения первого уровня имеют больший эффект, чем возможности операторов восстановления. Да-

лее решение может быть улучшено с помощью прохождения через процедуру имитации отжига. Более того, право свободного прохода через стадию согласования принятия текущего решения помогает тщательнее исследовать область поиска с помощью операторов из множества D_S, сохраняя при этом конфигурацию промежуточных станций неизменной.

4.1.1. Начальное решение

Для построения начального решения каждый клиент приписывается к определенной промежуточной станции механизмом колеса рулетки, основным критерием для которого служит расстояние от станции до клиента. Затем, строятся маршруты второго уровня и решается задача VRP для каждой промежуточной станции с помощью гибридного генетического алгоритма. И наконец, получив спрос каждой промежуточной станции, решаем задачу VRP для первого уровня, опять используя гибридный генетический алгоритм, который будет описан позднее.

4.1.2. Механизм колеса рулетки

Механизм колеса рулетки распределяет клиентов между промежуточными станциями по значениям функции приспособленности, в данном случае – по расстоянию от клиента до станции. Метод рулетки свое название получил по аналогии с известной азартной игрой. Каждому клиенту может быть сопоставлен сектор колеса рулетки, величина которого устанавливается пропорционально значению функции приспособленности данного клиента. Поэтому чем больше значение функции приспособленности, тем больше сектор на колесе рулетки. Все колесо рулетки соответствует сумме значений функций приспособленности всех клиентов рассматриваемого множества. Каждому клиенту $i \in V_c, i = 1, 2, ..., N$ (где N - число клиентов) соответствует сектор колеса

Algorithm 1 ALNS

$s \leftarrow InitialSolution$, $InitializeScore(\pi)$, $i \leftarrow 0$
repeat
  if $i = \omega$ **then**
    $N^- \leftarrow ChooseDestroyOperator(D_L, \pi)$
  else
    $N^- \leftarrow ChooseDestroyOperator(D_S, \pi)$
  end if
  $N^+ \leftarrow ChooseRepairOperator(R, \pi)$
  $s' \leftarrow DestroyAndRepairs(s, N^-, N^+)$
  if $i = \omega$ **then**
    $s' \leftarrow SimulatedAnnealing(s')$
    $s \leftarrow s'$
    $i \leftarrow 0$
  else if $f(s') < (1 + \theta)f(s^*)$ **then**
    $s' \leftarrow SimulatedAnnealing(s')$
  end if
  if $f(s') < f(s)$ **then**
    $s \leftarrow s'$
    $i \leftarrow 0$
  else
    $i \leftarrow i + 1$
  end if
  if $f(s) < f(s^*)$ **then**
    $s^* \leftarrow s$
  end if
  $UpdateScores(\pi)$
until text
return s^*

$g(i)$, выраженный в процентах согласно формуле:

$$g(i) = p_s(i) \cdot 100\% \tag{21}$$

$$p_s(i) = \frac{F(i)}{\sum_{i=1}^{N} F(i)} \tag{22}$$

Причем $F(i)$ – значение функции приспособленности клиента i, зависящее от расстояния между клиентом и станцией, а $p_s(i)$ – вероятность прикрепления данного клиента к соответствующей станции. Селекция клиента может быть представлена как результат поворота колеса рулетки, поскольку «выигравший» (т.е. выбранный) клиент относится к выпавшему сектору этого колеса. Очевидно, что чем больше сектор, тем больше вероятность «победы» соответствующего клиента. Поэтому вероятность выбора данного клиента оказывается пропорциональной значению его функции приспособленности.

4.1.3. Операторы разрушения

Далее опишем используемые в алгоритме разрушающие операторы. Их предложили Ropke S. и Pisinger D. [16] и Hemmelmayr V.C. в статье [6]. Операторы введенные Ropke S. и Pisinger D. были адаптированы для задачи 2E-VRP. Везде количество клиентов к удалению q выбирается в случайном порядке из интервала $[\rho, \tau]$.

1. Satellite Removal

   Среди всех открытых промежуточных станций выбираем случайным образом одну и закрываем. Все приписанные к ней клиенты тоже удаляются и помещаются в пул клиентов. Следовательно, все маршруты, берущие начало из этой станции, также удаляются. Далее, выбираем случайным

образом еще одну станцию и открываем ее, в случае если она ещё не открыта. Это предотвращает ситуацию, когда все промежуточные станции закрыты, потому что единственная открытая была закрыта этим оператором. Данный механизм также важен для диверсификации решения.

2. Satellite Opening

   Оператор случайным образом выбирает промежуточную станцию среди закрытых и открывает её. Тогда q наиболее близких к ней клиентов удаляются из текущих маршрутов и помещаются в пул клиентов.

3. Random Removal

   Данный оператор предложили Ropke S. and Pisinger D. [16]. Он случайным образом выбирает q клиентов и помещает их в пул.

4. Worst Removal

   Оператор Worst Removal также основан на похожем операторе, разработанным Ropke S. и Pisinger D. [16]. Этот оператор удаляет q клиентов с наибольшей величиной выигрыша от удаления. Величина выигрыша от удаления определяется как разница между стоимостью маршрута, когда клиент находится в решении, и стоимостью маршрута без этого клиента. Выигрыш нормируется делением на среднюю стоимость дуг из соответствующей вершины. Цель такого нормирования заключается в избежании повторяющегося выбора наиболее дальних клиентов. Стоимость также корректируется на величину $d \in [0.8, 1.2]$ в целях увеличения пространства получаемых решений. Оптимальный интервал для величины d подобран в [6].

5. Related Removal

Случайным образом выбирается первый удаляемый клиент (ключевой). Идентифицируются клиенты, расположенные вблизи ключевого клиента. Все эти клиенты затем удаляются из своих текущих маршрутов и помещаются в пул. Related Removal схож с оператором, который предложили Pisinger S. и Ropke D. в работе [17]. Однако в то время как они выбирают цепь клиентов, наиболее близко расположенных друг к другу в смысле расстояния, мы просто удаляем клиентов, находящихся вблизи ключевого клиента.

6. Route Removal

   Оператор Route Removal удаляет случайно выбранный маршрут и помещает его клиентов в пул клиентов. Открытие нового маршрута у соответствующей промежуточной станции оператором восстановления запрещается во избежание зацикливания. Как и Satellite Removal, данный оператор имеет механизм, позволяющий открывать ранее закрытые промежуточные станции, потому что иногда может возникнуть такая ситуация, что все клиенты обслуживаются единственным маршрутом, берущим начало в единственной открытой промежуточной станции.

4.1.4. Операторы восстановления

Операторы восстановления могут вставлять клиентов только в маршруты, исходящие из открытых станций. То есть станций, к которым на текущий момент привязаны клиенты, или открытых в результате применения оператора разрушения.

1. Greedy Insertion

   Клиенты в случайном порядке помещаются в позицию, минимизирующую

стоимость вставки по всем открытым промежуточным станциям и маршрутам. Также учитывается возможность открытия нового маршрута, если это не запрещено (например к соответствующей станции мог быть применен оператор Route Removal). Далее проверяем, остаются ли маршруты первого уровня осуществимыми после возрастания спроса у конкретной промежуточной станции. Если нет, ищем наиболее выгодную позицию для данной станции в маршрутах первого уровня. Впоследствии, происходят перестановки на первом уровне для улучшения маршрутов. Чтобы оператор стал простым и быстрым, вставка происходит в случайном порядке. Regret Insertion использует более сложный механизм с пересчетом.

2. Greedy Insertion Perturbation

   Этот оператор работает так же, как Greedy Insertion. Разница в том, что когда мы рассчитываем стоимость вставки клиента в каждую позицию, стоимость корректируется на величину $d \in [0.8, 1.2]$ в целях диверсификации получаемых решений. Идею такого оператора предложили Ropke S. и Pisinger D. [16], которые корректировали стоимость маршрутов для каждой потенциальной позиции вставки клиента на случайное значение из определенного интервала.

3. Regret Insertion

   Regret Insertion – оператор вставки с потерями. В рамках данной эвристики клиенты сортируются в порядке их величины потерь. Величина потерь рассчитывается как разница в стоимости между наилучшей позиции для вставки и второй/следующей наилучшей. Таким образом, клиенты с высоким значением величины потерь должны быть вставлены в первую очередь.

4.1.5. Выбор операторов разрушения и восстановления

Выбор основан на успешности применения каждого оператора на предыдущих итерациях. Операторы разрушения и восстановления взвешиваются и выбираются независимо друг от друга. Каждый раз, когда оператор j находит новое глобальное наилучшее решение, σ прибавляется к рейтингу π_j. В основе выбора операторов лежит механизм колеса рулетки.

Параметры и обозначения модели:

q — количество удаляемых из решения клиентов.

$[\rho, \tau]$ — интервал для выбора количества удаляемых клиентов.

D_L — операторы разрушения первого уровня.

D_S — операторы разрушения второго уровня.

R — операторы восстановления.

w — количество итераций без улучшения, после чего вызываются разрушающие операторы первого уровня D_L.

s — решение, получаемое в результате очередного применения операторов разрушения и восстановления.

s' — последнее принятое для дальнейшей модификации решение.

s^* — глобальное (наилучшее найденное решение).

θ — максимальное значение в процентах, на которое может отличатся решение, полученное в результате применения операторов из множества D_S, от глобального решения s^*, чтобы пройти дальше в фазу имитации отжига.

$d \in [0.8, 1.2]$ — величина, на которую корректируется стоимость маршрута для операторов Worst Removal и Greedy Insertion Perturbation.

π_j — рейтинг оператора.

σ — величина, на которую увеличивается рейтинг успешного оператора.

4.2. Гибридный генетический алгоритм

Гибридный генетический алгоритм является объединением алгоритма генетического и splitting [18]. Он позволяет создать начальное решение s для дальнейшего улучшения с помощью ALNS.

Гибридный генетический алгоритм:

1. Генерация начальной популяции решений (тысячи случайных последовательностей из маршрутов):
   1.1. Создание тысячи случайных последовательностей из всех точек (решений-хромосом).
   1.2. Разбиение этих последовательностей на маршруты для каждой машины (алгоритм splitting, предложил Prins C. [19]).
   1.3. Вычисление стоимости каждого решения.
   1.4. Сортировка решений по возрастанию стоимости.
2. Формирование новой популяции. Замена наихудшей половины имеющейся популяции новыми особями. Начало цикла:
   2.1. Выбор двух решений-родителей.
   2.2. Создание двух потомков при помощи операторов скрещивания и мутации.
   2.3. Разбиение решений-потомков на маршруты.
   2.4. Вычисление целевой функции (стоимости) для новых особей.
   2.5. Добавление потомков к формирующейся популяции.
   2.6. Повтор п. 2.1 – 2.5 до получения первоначального объема популяции.
   2.7. Конец цикла. Сортировка решений по возрастанию стоимости.

3. Повтор п. 2 до достижения максимально возможного значения итераций.

4. Решение, находящееся в начале окончательной популяции и есть наилучшее из найденных.

4.2.1. Алгоритм splitting

Так как решение-хромосома, состоящее из последовательности точек, может быть разбито на множество маршрутов, Prins [19] предложил процедуру splitting, которая находит оптимальное разбиение. То есть такое разбиение на маршруты между машинами, которое минимизирует общие издержки, в данном случае общее расстояние. Главная идея может быть описана следующим образом. Пусть $S = (1, 2, 3, ..., n)$ — хромосома из n точек. Рассмотрим вспомогательный граф $H = (\bar{V}, \bar{A})$, где $\bar{V} = (0, 1, 2, ..., n)$, $\bar{A}$ — множество дуг (пар вершин). Последовательность точек $i, ..., j \in \bar{A}$, если

$$\sum_{k=i+1}^{j} d_k \leq K_2, \tag{23}$$

где K_2 — вместительность машин второго уровня.

Тогда $\bar{A}_{ij}$ — общая стоимость маршрута $(i+1, i+2, ..., j)$. Оптимальное разбиение хромосомы-решения S для конкретной промежуточной станции соответствует кратчайшему пути P из вершины 0 в вершину n на графе H.

Вспомогательный граф помогает нам понять идею разбиения хромосомы-решения (последовательности точек) S на оптимальные маршруты. Однако на практике нам не обязательно строить такой граф H. Разбиение может быть сделано с помощью алгоритма индексации и процедуры splitting. Пусть $S = (1, 2, ..., n)$ — текущая хромосома. Для каждой вершины j из S вычисляются два индекса, V_j и P_j.

V_j — стоимость кратчайшего пути из точки 0 в точку j из H.

P_j — предшественник точки j в данной последовательности.

В конце алгоритма получаем минимальную стоимость — V_n. Необходимо учесть, что для любой точки i увеличение j прекращается, если превышена максимальная вместительность машины K_2. Далее описан алгоритм индексации:

Algorithm 2 Splitting

$V_0 \leftarrow 0$
for $i = 1$ to n **do**
  $V_i \leftarrow +\infty$
end for
for $i = 1$ to n **do**
  $cost \leftarrow 0$, $load \leftarrow 0$, $j \leftarrow i$
  repeat
    $load \leftarrow load + q_{S_j}$
    if $i = j$ **then**
      $cost \leftarrow c_{0,S_j} + d_{S_j} + c_{S_j,0}$
    else
      $cost \leftarrow cost - c_{S_{j-1},0} + c_{S_{j-1},S_j} + d_{S_j} + c_{S_j,0}$
    end if
    if $cost \leqslant L$ and $load < Q$ **then**
      if $V_{i-1} + cost < V_j$ **then**
        $V_j \leftarrow V_{i-1} + cost$
        $P_j \leftarrow i - 1$
      end if
      $j \leftarrow j + 1$
    end if
  until $j > n$ or $cost > L$ or $load > Q$
end for

4.3. Алгоритм имитации отжига

Для улучшения решения s, найденного в результате применения операторов разрушения и восстановления, используем имитацию отжига. Алгоритм имитации отжига (Simulated annealing) — алгоритм решения различных оптимизационных задач. Он основан на моделировании реального физического процесса, который происходит при кристаллизации вещества из жидкого состояния в твёрдое, в том числе при отжиге металлов.

Целью алгоритма является минимизация некоторого функционала. В процессе работы алгоритма хранится текущее решение, которое является промежуточным результатом. А после завершения работы алгоритма оно и будет ответом.

Основные шаги алгоритма [20]:

1. Выбор начального решения и начальной температуры.
2. Оценка начального решения.
3. Основной шаг алгоритма.
   - 3.1. Случайное изменение текущего решения.
   - 3.2. Оценка измененного решения.
   - 3.3. Критерий допуска.
4. Уменьшение температуры и, если температура больше некоторого порога, то переход к основному шагу.

Выбор начального решения

В качестве начального решения в алгоритм поступает решение s, полученное в результате применения операторов разрушения и восстановления.

Это предоставляет алгоритму базу, на основании которой он будет строить лучшее решение.

Оценка решения

Этот этап полностью зависит от специфики задачи. Единственным требованием является получение в качестве оценки одного вещественного числа, которое будет характеризовать оптимальность предлагаемого решения. Это число в алгоритме имитации отжига принято называть энергией. В данном случае оценкой решения является значение целевой функции.

Основной шаг алгоритма

Основной шаг, представленный в п. 3 алгоритма, при некоторой температуре повторяется несколько раз. Возможно, что один раз. Также возможен вариант с зависимостью числа повторов от температуры.

Случайное изменение решения

Этот этап сильно зависит от специфики задачи. Однако, изменения стоит производить локальные. Например, для задачи 2E-VRP, хорошей стратегий будет обмен в текущем порядке следования городов, двух случайных городов местами, перемещение точки в новое место в маршруте. В результате изменения у нас будет два решения: текущее и измененное.

Критерий допуска

Для определенности будем считать, что оптимизация заключается в минимизации энергии. В большинстве случаев этот подход справедлив. Критерий допуска заключается в проверке и возможной замене текущего решения измененным.

Если измененное решение имеет меньшую энергию, то оно принимается за текущее. Если же измененное решение имеет большую энергию, то оно

принимается с вероятностью $P = e^{-\delta E/T}$, где:

P — вероятность принять измененное решение,

δE — модуль разности между энергией оптимального решения и энергий измененного решения,

T — текущая температура.

Уменьшение температуры

Важной частью алгоритма является уменьшение температуры. При большой температуре вероятность выбора относительно худшего решения высока. Однако, в процессе работы алгоритма температура снижается, и вероятность выбора худшего решения снижается.

Выбор способа уменьшения температуры может быть различным и выбирается экспериментально. Главное, чтобы температура монотонно убывала к нулю. Хорошей стратегий является умножение на каждом шаге температуры на некоторый коэффициент α немного меньший единицы. В данном работе $\alpha = 0,98$.

Выбор начальной и пороговой температуры

Этот выбор тоже следует производить экспериментально. Естественными рекомендациями могут служить выбор пороговой температуры T_{min} близкой к нулю, а начальной T_{init} достаточно высокой. В нашей задаче $T_{init} = 100$.

5. Результаты

В данной работе был разработан и протестирован видоизмененный алгоритм ALNS.

Изменеия в ALNS:

1. Начальное решение находится иным способом: savings алгоритм заменен на гибридный генетический алгоритм, представленный в статье Chang Y. и Chen L. [15].
2. Вместо локального поиска, применяемого после каждого оператора восстановления, было использовано сочетание алгоритма splitting и алгоритма имитации отжига.
3. Исключен из реализации оператор разрушения Route Redistribution, так как он удаляет большую часть маршрутов и приводит решение в начальные условия.
4. Исключен из реализации оператор восстановления Greedy Insertion Forbidden, так как он практически не отличается от оператора Greedy Insertion и, как показали в своей статье [6] Hemmelmayr V.C., Cordeau J.F. и Crainic T.G., решение этот оператор существенно не улучшает.
5. Исключен оператор первого уровня Satellite Swap, который также, по статистике, не улучшает решение.

Далее этот гибридный алгоритм ALNS был реализован на языке MATLAB и протестирован на множестве стандартных примеров для задачи 2E-VRP (set2 и set3).

5.1. Описание тестовых примеров

Для проверки работы разработанного комплексного подхода в данной работе были использованы три множества тестовых примеров из литературы. Множества под номерами 2 и 3 (set 2 и set 3)предложили Perboli G. и другие [21]. Они основаны на следующих тестовых примерах: E-n13-k4, E-n22-k4, E-n33-k4 и E-n51-k5, авторы которых — Christofides и Eilon. Эти авторы также предложили множество примеров малой размерности (set 1), состоящих из одного депо, двух промежуточных станций и двенадцати клиентов. Последнее множество не использовалось в данной работе в связи с его простотой.

Обозначим:

- m — количество промежуточных станций;
- n — количество клиентов.

Вспомним, что:

- m_1 — количество машин первого уровня;
- m_2 — количество машин второго уровня;
- K_1 — вместительность машин первого уровня;
- K_2 — вместительность машин второго уровня.

Ниже в таблице представлены основные характеристики тестовых примеров:

Так, например, множество set 2 состоит из четырех подкатегорий примеров:

- тестовые примеры размерностью в 24 точки;
- тестовые примеры размерностью в 35 точек;

№ множества тестовых примеров	Количество примеров в подкатегории множества	m	n	m_1	m_2	K_1	K_2
2	6	2	21	3	4	15 000	6000
2	6	2	32	3	4	20 000	8000
2	6	2	50	3	5	400	160
2	3	4	50	4	5	400	160
3	6	2	21	3	4	15 000	6000
3	6	2	32	3	4	20 000	8000
3	6	2	50	3	5	400	160

Таблица 1. Характеристики тестовых примеров для 2E-VRP.

- тестовые примеры размерностью в 53 точки и с двумя промежуточными станциями;
- тестовые примеры размерностью в 55 точек и с четырьмя промежуточными станциями.

5.2. Подбор параметров

Для подбора оптимальных значений параметров ω и τ было выбрано репрезентативное подмножество тестовых примеров задачи 2E-VRP. Оказалось, что 2E-VRP примеры нечувствительны к изменению значения параметра ω, который является максимальным значением допущения итераций без улучшения решения, при достижении ω вызываются разрушающие операторы первого уровня.

Количество удаляемых клиентов — случайное целое число между ρ и τ. Параметру ρ присвоено значение 1. Pisinger D. и Ropke S. [17] советуют установить следующее значение для τ: $\tau = min\{60, 0.4n\}$.

Величина σ , на которую увеличивается рейтинг успешно использованного оператора π_j, равна единице. Максимальное количество итераций гибридного ALNS равно 5000 – 20000 итераций, в зависимости от размерности тестового примера. Для оператора Regret Insertion была использована regret-3 эвристика и интервал потенциально лучшего решения, входящего в фазу имитации отжига, определяется параметром $\theta = 2\%$.

5.3. Результаты вычислений

В таблице ниже и далее информация по результатам тестирования примеров представлена следующим образом:

Столбец 1 — название тестового примера.

Столбец 2 — наилучшее известное значение целевой функции.

Столбец 3 — значение целевой функции s^* в результате использования алгоритма ALNS, модифицированного в данной работе.

Столбец 4 — величина отклонения в процентах столбца 3 от столбца 4, другими словами, на сколько применение модифицированного ALNS улучшило/ухудшило решение. Жирным шрифтом обозначены улучшения (значение целевой функции уменьшилось), красным цветом – ухудшения (значение целевой функции увеличилось).

Результаты для множества тестовых примеров из set 2 для задачи 2E-VRP:

1. Параметры примеров E-n22:

   - размерность: 24;
   - количество промежуточных станций m: 2;
   - количество клиентов n: 21;
   - вместительность транспорта первого уровня K_1: 15000;
   - вместительность транспорта второго уровня K_2: 6000;
   - количество машин первого уровня m_1: 3;
   - количество машин второго уровня m_2: 4.

Тестовый пример	Наилучшее известное решение	hybrid ALNS avg.	Отклонение в %
E-n22-k4-s6-17	417.07	417.07	0.00
E-n22-k4-s8-14	384.96	384.96	0.00
E-n22-k4-s9-19	470.60	459.70	**-2.32**
E-n22-k4-s10-14	371.50	371.50	0.00
E-n22-k4-s11-12	427.22	427.22	0.00
E-n22-k4-s12-16	392.78	384.72	**-2.05**
Среднее	410.69	407.53	**-0.77**

Таблица 2. Результаты для примеров E-n22 из set2.

Для примеров E-n22-k4-s9-19 и E-n22-k4-s12-16 было найдено решение s^* с меньшей стоимостью, чем наилучшее известное. Для остальных примеров, описанных в таб. 2, найдены решения, равные наилучшим известным. Для этого нам понадобилось только 5000 итераций гибридного ALNS против 500000 итераций ALNS, предложенного в статье [6]. Модифицированный ALNS показывает хорошие результаты на примерах размерностью в 24 точки.

2. Параметры примеров E-n33:

   - размерность: 35;
   - количество промежуточных станций m: 2;
   - количество клиентов n: 32;
   - вместительность транспорта первого уровня K_1: 20000;
   - вместительность транспорта второго уровня K_2: 8000;
   - количество машин первого уровня m_1: 3;
   - количество машин второго уровня m_2: 4.

Тестовый пример	Наилучшее известное решение	hybrid ALNS avg.	Отклонение в %
E-n33-k4-s1-9	730.16	730.16	0.00
E-n33-k4-s2-13	714.63	701.90	**-1.78**
E-n33-k4-s3-17	707.41	713.47	0.85
E-n33-k4-s4-5	778.73	816.59	4.86
E-n33-k4-s7-25	756.84	753.35	**-0.46**
E-n33-k4-s14-22	779.05	840.31	7.86
Среднее	744.47	759.30	1.99

Таблица 3. Результаты для примеров E-n33 из set2.

Для примеров E-n33-k4-s2-13 и E-n33-k4-s7-25 было найдено решение s^* с меньшей стоимостью, чем наилучшее известное. Для E-n33-k4-s1-9 решение не отличается от наилучшего известного. Для остальных примеров, описанных в таб. 3, найдены решения, значение целевой функции которых больше наилучших известных решений в среднем на 4,52%. В данном случае для некоторых примеров необходимо увеличить количество итераций или ввести оператор разрушения большего масштаба.

Результаты для множества тестовых примеров из set 3 для задачи 2E-VRP:

1. Параметры примеров E-n22:

   - размерность: 24;
   - количество промежуточных станций m: 2;
   - количество клиентов n: 21;
   - вместительность транспорта первого уровня K_1: 15000;
   - вместительность транспорта второго уровня K_2: 6000;
   - количество машин первого уровня m_1: 3;
   - количество машин второго уровня m_2: 4.

Тестовый пример	Наилучшее известное решение	hybrid ALNS avg.	Отклонение в %
E-n22-k4-s13-14	526.15	523.06	**-0.59**
E-n22-k4-s13-16	521.09	516.46	**-0.89**
E-n22-k4-s13-17	496.38	490.69	**-1.15**
E-n22-k4-s14-19	498.80	476.40	**-4.49**
E-n22-k4-s17-19	512.80	502.58	**-1.99**
E-n22-k4-s19-21	520.42	511.18	**-1.78**
Среднее	512.61	503.39	**-1.80**

Таблица 4. Результаты для примеров E-n22 из set3.

Для всех примеров, описанных в таб. 4, было найдено решение с меньшей стоимостью, чем наилучшее известное. Для этого нам понадобилось только 5000 итераций гибридного ALNS против 500000 итераций ALNS, предложенного в статье [6]. Модифицированный ALNS показывает хорошие результаты на примерах размерностью в 24 точки.

2. Параметры примеров E-n33:

- размерность: 35;
- количество промежуточных станций m: 2;
- количество клиентов n: 32;
- вместительность транспорта первого уровня K_1: 20000;
- вместительность транспорта второго уровня K_2: 8000;
- количество машин первого уровня m_1: 3;
- количество машин второго уровня m_2: 4.

Тестовый пример	Наилучшее известное решение	hybrid ALNS avg.	Отклонение в %
E-n33-k4-s16-22	672.17	671.04	**-0.17**
E-n33-k4-s16-24	666.02	675.97	1.49
E-n33-k4-s19-26	680.36	677.35	**-0.44**
E-n33-k4-s22-26	680.37	693.24	1.89
E-n33-k4-s24-28	670.43	670.40	**-0.01**
E-n33-k4-s25-28	650.58	663.47	1.98
Среднее	669.99	675.25	0.79

Таблица 5. Результаты для примеров E-n33 из set3.

Для примеров E-n33-k4-s16-22, E-n33-k4-s19-26 и E-n33-k4-s24-28 было найдено решение с меньшей стоимостью, чем наилучшее известное. Для остальных примеров, описанных в таб. 5, найдены решения, значение целевой функции которых больше наилучших известных решений в среднем на 1.79%. В данном случае для некоторых примеров необходимо увеличить количество итераций или ввести оператор разрушения большего масштаба.

5.4. Пример полученного решения

Ниже представлен пример полученного решения для тестового примера E-n33-k4-s2-13 из set 2.

Параметры примера E-n33-k4-s2-13:

- размерность: 35;
- количество промежуточных станций m: 2;
- количество клиентов n: 32;
- вместительность транспорта первого уровня K_1: 20000;
- вместительность транспорта второго уровня K_2: 8000;
- количество машин первого уровня m_1: 3;
- количество машин второго уровня m_2: 4.

Наилучшее известное значение целевой функции: **714.63**.

Значение целевой функции, полученное с помощью реализации разработанного комплексного подхода: **701.90**.

Отклонение: **-1.78 %** (полученное решение лучше наилучшего известного на -1.78%).

Далее представлены построенные маршруты для двух уровней.

Обозначения:

- квадрат – депо (v_0),
- ромбы – промежуточные станции (V_s),
- выколотые точки – множество клиентов (V_c),
- пунктирные линии – маршруты первого уровня,

- сплошные линии – маршруты второго уровня.

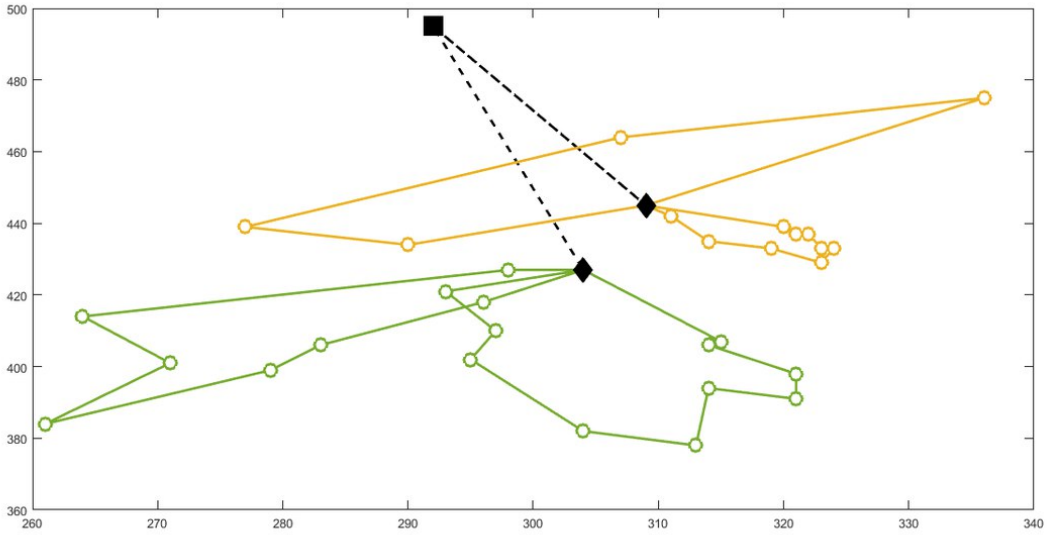

Рис. 2. Построенные маршруты для тестового примера E-n33-k4-s2-13.

Получаем два маршрута первого уровня и по два маршрута второго уровня для каждой промежуточной станции.

6. Заключение

В данной работе был разработан и протестирован видоизмененный алгоритм ALNS. За основу взят комплексный подход, предложенный в статье [6], который состоит из подразделов:

1. Распределение клиентов по промежуточным станциям при помощи механизма колеса рулетки.
2. Нахождение начального решения с помощью алгоритма savings.
3. Применение алгоритма ALNS в сочетании с локальным поиском к начальному решению.

Изменения в комплексном подходе, предложенном в статье [6], которые реализованы в данной выпускной работе:

1. Начальное решение находится иным способом: savings алгоритм заменен на гибридный генетический алгоритм, представленный в статье Chang Y. и Chen L. [15].
2. Вместо локального поиска, применяемого после каждого оператора восстановления, было использовано сочетание алгоритма splitting и алгоритма имитации отжига.
3. Исключен из реализации оператор разрушения Route Redistribution, так как он удаляет большую часть маршрутов и приводит решение в начальные условия.
4. Исключен из реализации оператор восстановления Greedy Insertion Forbidden, так как он практически не отличается от оператора Greedy

Insertion и, как показали в своей статье [6] Hemmelmayr V.C., Cordeau J. F. и Crainic T.G., решение этот оператор существенно не улучшает.

5. Исключен оператор первого уровня Satellite Swap, который также, по статистике, не улучшает решение.

Данный модифицированный комплекс был протестирован на множестве стандартных примеров для задачи 2E-VRP (set2 и set3).

На примерах малой размерности из 24 точек модифицированный ALNS показывает результаты в большинстве своем лучше, чем наилучшие известные, за меньшее количество итераций. Значение целевой функции для примеров из set2 в среднем меньше на 0.77%, а для примеров из set3 — в среднем меньше на 1,8%.

Примеры же большей размерности, 35 точек, дают достаточно хорошие решения. Но необходимо дальнейшее усовершенствование комплекса – внедрение разрушающих операторов большего масштаба, например операторов Satellite Swap и Route Redistribution, предложенных в статье [6].

В заключение стоит отметить, что уже сейчас разработанный в данной работе комплекс решений может быть адаптирован под нужды промышленности и успешно использован в конкретных прикладных задачах, с которыми в рамках логистики сталкиваются многие крупные компании.

Литература

1. Cuda R., Guastaroba G., Speranza M.G. A survey on two-echelon routing problems // Computers & Operations Research. 2015. Vol. 55, P. 185—199.

2. Contardo C., Crainic T.G., Hemmelmayr V. Lower and upper bounds for the two-echelon capacitated location routing problem // Computers & Operations Research. 2012. Vol. 39, P. 3215—3228.

3. Boccia M., Crainic T.G., Sforza A., Sterle C. A metaheuristic for a two echelon location-routing problem. // Experimental algorithms: lecture notes in computer science. Vol. 6049. Springer, 2010. P. 288–301.

4. Crainic T.G., Sforza A., Sterle C. Location-routing models for two-echelon freight distribution system design // Technical report CIRRELT. 2011. Vol. 40.

5. Schwengerer M., Pirkwieser S., Raidl G.R. A variable neighborhood search approach for the two-echelon location-routing problem // Evolutionary Computation in Combinatorial Optimization: Lecture Notes in Computer Science. Vol. 7245. Springer, 2012. P. 13–24.

6. Hemmelmayr V.C., Cordeau J.F., Crainic T.G. An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics // Computers & Operations Research. 2012. Vol. 39, P. 3215–3228.

7. Baldacci R., Mingozzi A., Roberti R., Wolfler C.R. An exact algorithm for the two-echelon capacitated vehicle routing problem // Operational Research. 2013. Vol. 2. P. 298–314.

8. Villegas J.G., Prins C., Prodhon C., Medaglia A., Velasco N. A GRASP with evolutionary path relinking for the truck and trailer routing problem // Computers & Operations Research. 2011. Vol. 38. P. 1319–34.

9. Villegas J.G., Prins C., Prodhon C., Medaglia A., Velasco N. A matheuristic for the truck and trailer routing problem // European Journal of Operational Research. 2013. Vol. 230. P. 231–244.

10. Chao I.M. A tabu search method for the truck and trailer routing problem // Computers & Operations Research. 2002. Vol. 29. P. 33–51.

11. Scheuerer S. A tabu search heuristic for the truck and trailer routing problem // Computers & Operations Research. 2006. Vol. 33. P. 894–909.

12. Caramia M., Guerriero F. A heuristic approach for the truck and trailer routing problem // Journal of the Operational Research Society. 2010. Vol. 61. P. 1168–1180.

13. Lin S., Yu V.F., Chou S. Solving the truck and trailer routing problem based on a simulated annealing heuristic // Computers & Operations Research. 2009. Vol. 36. P. 1683–1692.

14. Jesus Gonzalez-Feliu, Guido Perboli, Roberto Tadei, Daniele Vigo. The two-echelon capacitated vehicle routing problem. 2008.

15. Chang Y., Chen L. Solve the vehicle routing problem with time windows via a genetic algorithm // Discrete and Continuous Dynamical Systems. 2007. P. 240–249.

16. Ropke S., Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows // Transportation Science. 2006. Vol. 40. P. 455-–472.

17. Pisinger D., Ropke S. A general heuristic for vehicle routing problems // Computers & Operations Research. 2007. Vol. 34. P. 2403-–2435.

18. Yaw C., Lin C. Solve the vehicle routing problem with time windows via a genetic algorithm // Discrete and continuous dynamical system supplement. 2007. P. 240–249.

19. Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem // Computer & Operations Research. 2004. Vol. 31. P. 1985–2002.

20. Wayan F.M. Improved simulated annealing for optimization of vehicle routing problem with time windows (VRPTW) // KURSOR Journal. 2014. Vol. 7, No. 3. P. 109–116.

21. Perboli G., Tadei R., Vigo D. The two-echelon capacitated vehicle routing problem: models and math-based heuristics // Transportation Science. 2011. Vol. 45, P. 364-–380.

22. Mehmet S., Jacqueline M., Tolga B. The time-dependent two-echelon capacitated vehicle routing problem with environmental considerations // Production Economics. 2015. Vol. 164. P. 366–378.

23. Rodrigo R., El-Houssaine A. City Logistics and Traffic Management: Modelling the Inner and Outer Urban Transport Flows in a Two-Tiered System // Transportation Research Procedia. 2015. Vol. 5. P. 297–312.

24. Crainic T., Perboli G., Mancini S., Tadei R. The two-echelon capacitated vehicle routing problem: a satellite location analysis // PROCEDIA—Social and Behavioral Sciences. 2010. Vol. 2(3). P. 5944–5955.

Приложение

Основной фрагмент кода для разработанного модифицированного ALNS.

Файл ALNS.m

```
function [ FirstLevelRoute, SecondLevelRoute ] = ...
    ALNS( FirstLevelRoute, SecondLevelRoute,  Distances, Demand, MaxCost, MaxLoad)

        LargeDestroyScores = [50; 50];
        % 1 - SatelliteRemoval
        % 2 - SatelliteOpening
        SmallDestroyScores = [50; 50; 50; 50];
        % 1 - RandomRemoval
        % 2 - WorstRemoval
        % 3 - RelatedRemoval
        % 4 - RouteRemoval
        RepairScores = [50; 50; 50];
        % 1 - GreedyInsertion
        % 2 - GreedyInsertionPereturbarion
        % 3 - RegretInsertion
        w = 100;
        i = 0;
        OpenedSatellites = FirstLevelRoute{1,1}(2:end);
        ClosedSatellites = [];
        DeletedNodes = [];
        SecondLevelLength = 0;
        for j = 1:size(SecondLevelRoute, 1)
            SecondLevelLength = SecondLevelLength + SecondLevelRoute{j, 1}{1, 3};
        end
        MinFirstLevelRoute = FirstLevelRoute;
        MinSecondLevelRoute = SecondLevelRoute;
        MinLength = FirstLevelRoute{1, 3} + SecondLevelLength;
        GlobalMinFirstLevelRoute = FirstLevelRoute;
        GlobalMinSecondLevelRoute = SecondLevelRoute;
        GlobalMinLength = FirstLevelRoute{1, 3} + SecondLevelLength;
        k = 1;
        MAX_ITERATION = 5000;
        while k <= MAX_ITERATION
            disp(['GlobalIteration ' num2str(k) ': Best Cost = ' num2str(GlobalMinLength)]);
            if i == w
                summ = 0;
                distribution = zeros(size(LargeDestroyScores));
                for j = 1:size(LargeDestroyScores, 1)
                    distribution(j, 1) = ...
```

```
                    LargeDestroyScores(j)/sum(LargeDestroyScores) + summ;
                summ = distribution(j, 1);
            end
            roulette_indexes = find(rand() < distribution);
            large_destroy_index = roulette_indexes(1);
            if large_destroy_index == 1
                [FirstLevelRoute, SecondLevelRoute, DeletedNodes, OpenedSatellites, ...
                    ClosedSatellites]= SatelliteRemoval(FirstLevelRoute, ...
                    SecondLevelRoute, OpenedSatellites, ClosedSatellites, Demand, ...
                    MaxLoad(1), DeletedNodes, Distances);
            else
                q = randi(5);
                [ FirstLevelRoute, SecondLevelRoute, OpenedSatellites, ...
                    ClosedSatellites, DeletedNodes] = SatelliteOpening( ...
                    FirstLevelRoute, SecondLevelRoute, OpenedSatellites, ...
                    ClosedSatellites, Distances, DeletedNodes, Demand, MaxLoad(1), q);
            end
            i = 0;
            DeletedNodes = [DeletedNodes; ClosedSatellites];
        else
            summ = 0;
            distribution = zeros(size(SmallDestroyScores));
            for j = 1:size(SmallDestroyScores, 1)
                distribution(j, 1) = ...
                    SmallDestroyScores(j)/sum(SmallDestroyScores) + summ;
                summ = distribution(j, 1);
            end
            roulette_indexes = find(rand() < distribution);
            small_destroy_index = roulette_indexes(1);
            q = randi(5);
            if small_destroy_index == 1
                [ SecondLevelRoute, TmpDeletedNodes ] = ...
                    RandomRemoval( SecondLevelRoute, q, Distances );
            elseif small_destroy_index == 2
                [ SecondLevelRoute, TmpDeletedNodes ] = ...
                    WorstRemoval( SecondLevelRoute, q, Distances );
            elseif small_destroy_index == 3
                [ SecondLevelRoute, TmpDeletedNodes ] = ...
                    RelatedRemoval( SecondLevelRoute, q, Distances, OpenedSatellites );
            else
                [ SecondLevelRoute, TmpDeletedNodes ] = ...
                    RouteRemoval( SecondLevelRoute, Distances);
            end
            DeletedNodes = [DeletedNodes; TmpDeletedNodes(:, 1)];
        end

        if ~isempty(DeletedNodes)
            summ = 0;
```

```
88                  distribution = zeros(size(RepairScores));
89                  for j = 1:size(RepairScores, 1)
90                      distribution(j, 1) = ...
91                        RepairScores(j)/sum(RepairScores) + summ;
92                        summ = distribution(j, 1);
93                  end
94                  roulette_indexes = find(rand() < distribution);
95                  repair_index = roulette_indexes(1);
96
97                  if repair_index == 1
98                      [ FirstLevelRoute, SecondLevelRoute, DeletedNodes ] = ...
99                      GreedyInsertion( MaxLoad(1), FirstLevelRoute, ...
100                     MaxLoad(2), SecondLevelRoute, ...,
101                     Distances, Demand, DeletedNodes );
102                 elseif repair_index == 2
103                     [ FirstLevelRoute, SecondLevelRoute, DeletedNodes ] = ...
104                     GreedyInsertionPereturbation(...
105                         MaxLoad(1), FirstLevelRoute, ...
106                         MaxLoad(2), SecondLevelRoute, ...,
107                         Distances, Demand, DeletedNodes );
108                 else
109                     K = randi(5);
110                     [ FirstLevelRoute, SecondLevelRoute, DeletedNodes ] = RegretInsertion(...
111                         MaxLoad(1), FirstLevelRoute, ...
112                         MaxLoad(2), SecondLevelRoute, ...,
113                         Distances, Demand, DeletedNodes, K);
114                 end
115             end
116
117             if i == w
118                 [ SecondLevelRoute ] = SimulatedAnnealing( SecondLevelRoute, Distances, ...
119                 Demand, MaxCost, MaxLoad );
120                 SecondLevelLength = 0;
121                 for j = 1:size(SecondLevelRoute, 1)
122                     SecondLevelLength = SecondLevelLength + SecondLevelRoute{j, 1}{1, 3};
123                 end
124                 CurLength = FirstLevelRoute{1, 3} + SecondLevelLength;
125                 if CurLength < MinLength
126                     LargeDestroyScores(large_destroy_index) = ...
127                             LargeDestroyScores(large_destroy_index) + 1;
128                     RepairScores(repair_index) = RepairScores(repair_index) + 1;
129                 end
130                 MinFirstLevelRoute = FirstLevelRoute;
131                 MinLength = CurLength;
132                 MinSecondLevelRoute = SecondLevelRoute;
133                 i = 0;
134             else
135                 SecondLevelLength = 0;
```

```
136                 for j = 1:size(SecondLevelRoute, 1)
137                     SecondLevelLength = SecondLevelLength + SecondLevelRoute{j, 1}{1, 3};
138                 end
139                 CurLength = FirstLevelRoute{1, 3} + SecondLevelLength;
140                 if  CurLength < 1.02 * GlobalMinLength
141                     [ SecondLevelRoute ] = SimulatedAnnealing( SecondLevelRoute, ...
142                     Distances, Demand, MaxCost, MaxLoad );
143                 end
144                 SecondLevelLength = 0;
145                 for j = 1:size(SecondLevelRoute, 1)
146                     SecondLevelLength = SecondLevelLength + SecondLevelRoute{j, 1}{1, 3};
147                 end
148                 CurLength = FirstLevelRoute{1, 3} + SecondLevelLength;
149                 if CurLength < MinLength
150                     MinFirstLevelRoute = FirstLevelRoute;
151                     MinLength = CurLength;
152                     MinSecondLevelRoute = SecondLevelRoute;
153                     SmallDestroyScores(small_destroy_index) = ...
154                     SmallDestroyScores(small_destroy_index) + 1;
155                     RepairScores(repair_index) = RepairScores(repair_index) + 1;
156                     i = 0;
157                 else
158                     i = i + 1;
159                 end
160 
161             end
162             if MinLength < GlobalMinLength
163                 GlobalMinFirstLevelRoute = MinFirstLevelRoute;
164                 GlobalMinLength = MinLength;
165                 GlobalMinSecondLevelRoute = MinSecondLevelRoute;
166             end
167 
168             k = k + 1;
169         end
170         FirstLevelRoute = GlobalMinFirstLevelRoute;
171         SecondLevelRoute = GlobalMinSecondLevelRoute;
172 end
```