1. Аннотация

Целью данного обзора является рассмотрение существующих классических подходов, решающих проблему маршрутизации транспортных средств с некоторыми ограничениями. В будущем планируется разработать собственный алгоритм машинного обучения в задаче построения оптимизированных маршрутов на реальных географических данных с ограничениями, а полученные результаты сравнить с результатами из данного обзора.

В работе проведены экспериментальные исследования на одинаковых наборах данных, на которых алгоритмы оптимизируют конкретные задачи. Также приведены сравнения времени работы и оптимальности решений всех алгоритмов, описанных в данном обзоре.

2. Введение

2.1 Задача VRP

Проблема маршрутизации транспортных средств (***Vehicle Routing Problem***) – это класс задач комбинаторной оптимизации, суть которой в построении оптимального маршрута между депо и клиентами, зачастую с дополнительными ограничениями на:

- время;
- объем товаров;
- последовательность доставок.

VRP легко представим в виде графа с $N + 1$ вершиной, где N вершин это клиенты и одно депо. Каждое ребро отвечает за маршрут, пройденный одним или несколькими транспортными средствами.

Цель задачи состоит в том, чтобы уменьшить некую заданную стоимость этого маршрута (пройденное расстояние, время или количество транспортных средств и т.п.). В целом данная задача может быть сформулирована следующим образом:

«Каков оптимальный набор маршрутов для транспортных средств, чтобы доставить продукт определенной группе клиентов с минимальными затратами?»

В такой постановке можно заметить, что задача является модификацией задачи Коммивояжера, а значит является NP-трудной, что накладывает свои ограничения.

2.2 Актуальность

Проблемы маршрутизации транспортных средств имеют большое практическое значение. Например, распределение заказов такси, все возможные логистические задачи распределения товаров, курьерская доставка и еще множество других. Сейчас существует множество открытых и коммерческих полностью готовых инструментов, способных решать задачу VRP, и при этом демонстрировать хорошие результаты.

Стоит отметить, что чаще всего задача построения маршрутов является динамической или стохастической:

Динамическая постановка VRP (Dynamic VRP)

Данная проблема является одним из важных вариантов задачи VRP. Цель задачи состоит в построении оптимальных маршрутов для транспортных средств, необходимых для обслуживания определенного набора клиентов, пока поступают новые заказы от клиентов во время выполнения доставки заказов. Таким образом, маршруты должны быть перестроены динамически. То есть в течение доставки заказов маршрут может меняться, например:

- отменяются или добавляются заказы;

- существуют ограничения по времени на доставку заказов;
- во время доставки могут возникать различные обстоятельства, такие как сильно отличающиеся загруженности в часпик и в повседневное время.

Стохастическая постановка VRP (Stochastic VRP)

Цель данной задачи состоит в построении оптимальных наборов маршрутов для транспортных средств, необходимых для обслуживания определенного набора клиентов, но один или несколько параметров рассматриваются как случайные или неизвестные в течение планирования маршрута. Случайными параметрами могут быть

- количество клиентов;
- характер потребительского спроса в данном местоположении;
- время, такое как время обслуживания и время в пути;
- возникновение пробок, например загруженность определенных магистралей с течением времени может изменяться.

В современных задачах различных ограничений достаточно много. Например, одни из самых распространенных это:

Временные окна (VRP with Time Windows)

Этим занимается «*Проблема маршрутизации транспортных с временными окнами*».

Транспортное средство затрачивает время на перемещение и на обслуживание клиентов. Каждому клиенту сопоставляется некий промежуток времени состоящий из двух моментов - это время наиболее ранней допустимой доставки и время наиболее поздней. Таким образом, задача заключается в построении оптимального маршрута для транспортных средств, чтобы посетить как можно большее число клиентов, не нарушая, выделенные временные рамки для каждого клиента.

Удовлетворение спроса (Capacity VRP)

Этим занимается «*Проблема маршрутизации транспортных средств с ограниченной вместимостью*».

Формально спрос представляет собой некоторую неотрицательную величину, характеризующую объем товара запрашиваемого клиентом. Для удовлетворения спроса необходима перевозка заданного объема товара из места хранения к клиенту транспортным средством, имеющим определенную вместимость. Таким образом, на маршрут накладывается следующее ограничение: вдоль пути от депо и обратно возможно посетить лишь то число клиентов, суммарный спрос которых не превышает вместимость транспортного средства.

3. Постановка задачи

Основной задачей данного исследования является разработка собственного алгоритма машинного обучения в задаче построения оптимизированных маршрутов на реальных географических данных с ограничениями на:

- вместимость транспортных средств;
- время доставки и обслуживании клиентов.

На данный момент решалась второстепенная задача, результаты которой будут в будущем использованы при решении основной задачи.

Данная задача заключалась в следующем:

- Изучение точных и эвристических подходов для решения задачи.
- Реализация изученных алгоритмов.
- Сравнительный анализ реализаций алгоритмов на реальных географических данных.

4. Обзор существующих решений

4.1 Цели обзора

В рамках данного обзора для достижения необходимых результатов предполагается решение следующих подзадач:

1. Обзор существующих видов алгоритмов задачи VRP.
2. Обзор примеров классических эвристических алгоритмов «*Simulated Annealing*» и «*LKH*», и точного алгоритма «*Branch-and-Cut*».
3. Подготовка реализаций алгоритмов для решения задач *CVRP*, *CVRP-TW*.
4. Проведение экспериментов на открытых наборах данных.
5. Подготовка экспериментального стенда и документации.
6. Формулировка направлений дальнейших исследований.

4.2 Виды VRP алгоритмов

Обзор VRP алгоритмов начнем с эвристических алгоритмов. ***Эвристические алгоритмы*** являются наиболее распространенными алгоритмами для таких задач и можно выделить два больших класса: <u>*конструктивные эвристики*</u> и <u>*метаэвристики*</u>.

4.2.1 Конструктивные эвристики

Конструктивные эвристические алгоритмы вычисляют единственный маршрут для рассматриваемой задачи и не улучшают его в дальнейшем. Наиболее известными эвристиками среди них являются семейство жадных алгоритмов и алгоритмов ближайшего соседа, а также алгоритм Кристофидеса и различные модификации методов со вставками. Указанные методы генерируют допустимое решение, последовательно вставляя вершины в текущий маршрут, пока итоговый маршрутный план не будет составлен. Таким образом, часть маршрутного плана, сгенерированная к настоящему времени, не изменится и в итоговом решении.

В данном обзоре упор сделан на изучение метэавристических алгоритмов, т.к. во многих статьях, где решаются модификации задачи VRP используются в основном метаэвристики. Например, в данной статье [7] авторы предложили подход, который в дальнейшем стал широко использоваться для решения практических задач **MDVRP** (**Multi Depot VRP** -

VRP с несколькими депо) и при создании данного подхода использовался метаэвристический алгоритм имитации отжига.

4.2.2 Метаэвристики

Наиболее распространенным подходом для области VRP являются ***метаэвристики*** — это общие эвристики, позволяющие находить близкие к оптимальным решениям, решения различных задач оптимизации за приемлемое время.

Различные описания метаэвристик в литературе позволяют сформулировать некоторые фундаментальные свойства, которыми характеризуются метаэвристики:

- Метаэвристики - это стратегии, которые «управляют» процессом поиска решения.
- Цель метаэвристических алгоритмов состоит в эффективном исследовании пространства поиска для нахождения приближенных решений к оптимальным.
- Метаэвристические алгоритмы варьируются от простых процедур локального поиска до сложных процессов обучения.
- Метаэвристические алгоритмы могут включать механизмы избегания попадания в ловушку в ограниченной области пространства поиска.
- Современные метаэвристики используют сохраненный в памяти опыт поиска решения для управления поиском.

Каждая метаэвристика имеет свои собственное поведение и характеристики. Однако все метаэвристики имеют ряд основных компонент и выполняют операции в пределах ограниченного числа категорий.

1. *Инициализация*

Выбирается метод нахождения начального решения.

2. *Окрестности*

Каждому решению x соответствует множество окрестностей и связанные с ними переходы: $\{N_1, N_2, \ldots, N_q\}$.

3. *Критерий выбора окрестности*

Данный критерий определяется в случае наличия более одной окрестности, а также он должен указать не только выбираемую окрестность, но и условия ее выбора. Альтернативы варьируют от «на каждой итерации» (например, генетические методы) до «при данных условиях».

4. *Отбор кандидатов.*

Так как окрестности могут быть очень большими, то рассматривается только подмножество переходов на каждой итерации. Соответствующий список кандидатов $C(x) \subseteq N(x)$ может быть постоянным и обновляемым от итерации к итерации (например, табу-поиск), или же он может быть построен на каждой новой итерации (например, генетические методы). Во всех случаях критерий выбора определяет, каким образом могут быть выбраны решения для включения в список кандидатов.

5. *Критерий принятия.*

Переходы оцениваются с помощью функции $g(x, y)$ зависящей от таких параметров двух решений, как значение целевой функции, штрафы за нарушение некоторых ограничений и т.п. Выбирается наилучшее решение по отношению к этому критерию $\hat{x} = argopt\{g(x, y);\ y \in C(x)\}$ (с учетом необходимости предотвращения зацикливания).

6. *Критерии остановки.*

Метаэвристика может быть остановлена согласно различным критериям:

1. время вычислений;
2. число итераций;
3. темпы улучшения решения и т.п.

Может быть определен более чем один критерий для управления различными фазами поиска.

Используя эти определения, опишем метаэвристическую процедуру:

Общая метаэвристика

1. Инициализация x^0;
2. Выбор окрестности $\aleph \in \{N_1, \dots , N_q\}$;
3. Выбор кандидата $\beta(x) \subseteq \aleph(x)$;
4. Оценка перехода/исследование окрестности $g(x, y),\ y \in \beta(x)$;
5. Реализация перехода $\hat{x} = argopt\{g(x, y)\}$;
6. Оценка решения, обновить параметры поиска;
7. Проверка критериев остановки: *Стоп* или *Переход к шагу 3* (продолжить локальный поиск) или *Переход к шагу 2* (начать новый этап поиска).

Стоит отметить, что метаэвристики включают две категории: метаэвристики локального поиска и эволюционные алгоритмы.

4.2.3 Точные алгоритмы

Следующим видом VRP алгоритмов являются **точные алгоритмы**. К точным алгоритмам относятся алгоритмы линейного программирования. Это задачи, в которых функция цели и ограничений являются линейными. Задачи линейного программирования хорошо изучены и известны их свойства с точки зрения существования решения.

Задача линейного программирования (***LP***) — это задача оптимизации в стандартной форме, выражаемая следующими соотношениями:

$$max\{c^T x | Ax \leq b, x \geq 0\},$$

где $A \in R^{m,n}$ это технологическая матрица, $b \in R^m$ это вектор ресурсов, $c \in R^n$ это вектор цен, и $x \in R^n$ неизвестный вектор. Если некоторые или все переменные в векторе $\boldsymbol{x}$ являются целыми числами, проблема называется *проблемой смешанного целочисленного линейного программирования* (***MILP***).

Сформулируем постановку задачи целочисленного программирования для решения VRP.

Итак, пусть имеется ориентированный или неориентированный граф $G(V, E)$, который состоит из множества вершин $V = 0, 1,..., n$ и множества рёбер $E = \{(i, j)|i, j \in V, i \neq j\}$. Вершина отправления или депо - это вершина 0. Вершина отправления содержит не более m одинаковых транспортных средств одинаковой вместимости Q.

С каждой вершиной-потребителем $i \in V\backslash\{0\}$ связан неотрицательный запрос на поставку товара $q_i \leq Q$. Каждое ребро, соединяющее вершины i и j, связано с ценой c_{ij}.

Пусть теперь имеется множество n потребителей, которых необходимо обслужить $\{1,..., n\}$, причем каждому потребителю сопоставлен груз весом d_i, и пусть имеется множество m транспортных средств равной вместимости Q, находящихся в одном депо. Известно расстояние между каждой парой потребителей, а также между депо и каждым потребителем (c_{ij}, $i, j = 0,..., n$). Задача состоит в том, чтобы доставить товар всем потребителям, минимизировав общее расстояние, проходимое транспортными средствами, при соблюдении ограничений их вместимости.

Рассматривается полный граф с вершинами, соответствующими потребителям и депо. Допустимое решение состоит из семейства k ($k \leq m$) маршрутов, исходящих из депо, так что все потребители принадлежат одному из маршрутов.

Для начала назначим вершины потребительским транспортным средствам. Параметры, которые служат этой целью: y_{ih} = {1, если вершина i принадлежит маршруту транспортного средства h; 0, в противном случае}. Ограничения, связанные с этими параметрами:

$$(1)\ \sum_{h=1}^{m} y_{ih} = 1,\ i = 1,..., n$$

Уравнение (1) означает, что каждый потребитель должен принадлежать ровно одному маршруту. Количество транспортных средств можно задать, установив ограничение на количество раз, когда вершина 0 назначается маршруту, т.е. депо принадлежит каждому из маршрутов (2):

$$(2)\ \sum_{h=1}^{m} y_{0h} = m$$

Ограничение вместимости транспортного средства (3) также может быть выражено путем установления того, что сумма потребительских требований, отнесенных к одному и тому же транспортному средству, не должна превышать максимальную вместимость. Следует отметить, что требование депо равно нулю:

$$(3)\ \sum_{i=0}^{n} d_i y_{ih} \le Q,\ h = 1,..., m$$

Необходимо измерить расстояние, пройденное каждым из транспортных средств. Следовательно, вводится параметр x_{ij}^{h} для определения маршрута транспортного средства h как:

x_{ij}^{h} = {1, если транспортное средство h идет из вершины i в j; 0, в противном случае}, $i, j = 0, ..., n$, $h = 1, \ldots, m$.

Для того чтобы маршрут был приемлемым в качестве решения, необходимо соблюдение ограничений на количество транспортных средств, прибывающих к каждому клиенту и въезжающих в депо (4), выезжающих транспортных средств (5) и на исключение маршрутов, которые гарантируют, что решение не содержит циклов, отключенных от депо (6):

$$(4)\ \sum_{j=0, j\ne i}^{n} x_{ij}^{h} = y_{ih};\ i = 0,..., n;\ h = 1,..., m$$

$$(5)\ \sum_{j=0, j\neq i}^{n} x_{ji}^{h} = y_{ih};\ i = 0,..., n;\ h = 1,..., m$$

$$(6)\ \sum_{i,j\in S} x_{ij}^{h} \leq |S| - 1;\ S \subseteq \{1,..., n\}, |S| \geq 2, h = 1,..., m$$

$$(6')\ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij}^{h} = \sum_{i=1}^{n} y_{i}^{h} - 1;\ h = 1,..., m$$

Ограничения (4) и (5) определяют, что в наблюдаемом маршруте из наблюдаемой вершины может идти только одно ребро выхода в одну из других вершин, включая депо, если наблюдаемая вершина принадлежит наблюдаемому маршруту. В противном случае ни одно ребро не может перейти от наблюдаемой вершины к любой другой вершине. Ограничение (6) эквивалентно ограничению (6'), которое более удобно записать для целей программирования. Это ограничение устраняет петли, т.е. для n вершин, принадлежащих одному и тому же маршруту h, допускается существование $n - 1$ ответвлений между этими n вершинами. Одна вершина не учитывается, так как это депо.

Модель

Полная формулировка задачи маршрутизации транспортных средств принимает следующий вид:

Минимизировать

$$(7)\ min \sum_{h=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}^{h}$$

При условии

$$(8)\ \sum_{j=0, j\neq i}^{n} x_{ij}^{h} = \sum_{j=0, j\neq i}^{n} x_{ji}^{h} = y_{ih};\ i = 0,..., n;\ h = 1,..., m$$

$$(9)\ \sum_{h=1}^{m} \sum_{j=0}^{n} x_{ij}^{h} = 1;\ i = 1,..., n$$

(10) $\sum_{i=1}^{n}\sum_{j=0}^{n} d_i x_{ij}^{h} \leq Q;\ h = 1,\dots,m$

(11) $x_{ij}^{h} \in \{0, 1\};\ i, j = 0,\dots,n$

Основная трудность с задачами линейного программирования заключается в их размерности, так как могут быть тысячи переменных задачи и ограничений. Объем памяти и время решения могут увеличиваться экспоненциально по мере добавления целочисленных переменных. Поэтому были разработаны метаэвристики специально для того, чтобы найти решение за меньшее время и, которое приближено к оптимальному. Специально для сложных проблем, метаэвристики часто могут предложить лучший компромисс между качеством решения и вычислительным временем. Более того, метаэвристики более гибкие, чем точные методы.

Поскольку метаэвристические рамки определены в общих чертах, метаэвристические алгоритмы могут быть адаптированы для удовлетворения потребностей большинства задач оптимизации в реальном времени с точки зрения ожидаемого качества решения и допустимого вычислительного времени, которое может сильно различаться в разных задачах и в разных ситуациях.

4.2.4 Обучение с подкреплением (Reinforcement Learning)

Последний вид VRP алгоритмов это алгоритмы относящиеся к **обучению с подкреплением**.

Обучение с подкреплением - это обучение модели машинного обучения для принятия последовательности решений. Модель обучается тому, что делать и как сопоставлять ситуации с действиями, чтобы максимизировать вознаграждения. Модели не говорят, какие действия совершать, но вместо этого она должна выяснить, какие действия приносят наибольшую награду, попробовав их. В наиболее интересных и сложных случаях действия могут повлиять не только на немедленную награду, но и на следующую ситуацию, а через нее и на все последующие награды. Эти две характеристики — поиск методом проб и ошибок и отсроченное вознаграждение - являются двумя наиболее важными отличительными особенностями обучения с подкреплением. Используя возможности поиска и многочисленных испытаний, обучение с подкреплением в настоящее время является наиболее эффективным способом.

Далее будут рассмотрены постановки модификаций задачи VRP, которые решались в данном обзоре, а также описания алгоритмов, использовавшиеся для их решения.

4.3 Постановка задачи «CVRP»

Пусть $G = (V, E)$ полный ориентированный граф с набором вершин $V = \{0, 1, 2,..., n\}$ и набором рёбер $E = \{(i, j): i, j \in V, i \neq j\}$. Множество $K = \{1, 2, ..., |K|\}$ это парк транспортных средств с одинаковой вместимостью $Q \geq 0$. Каждый клиент $i \in V/\{0\}$, где $\{0\}$ - это депо, имеет определенный положительный спрос $d_i \leq Q$. Неотрицательная стоимость проезда c_{ij} связана с каждым ребром $(i, j) \in E$. Матрица стоимостей проездов симметрична, т.е. $c_{ij} = c_{ji}$ для всех $i, j \in V,\ i \neq j$ и удовлетворяет треугольному неравенству $c_{ij} + c_{jk} \geq c_{ik}$ для всех $i, j, k \in V$. Минимальное количество транспортных средств, необходимое для обслуживания всех клиентов: верхняя граница$(\frac{\sum_{i=1}^{n} d_i}{Q})$.

Существует переменная принятия решения, указывающая, выбрало ли транспортное средство k путь e_{ij}:

$x_{ij}^{k} = \{1,\ \text{решение прнято};\ 0,\ \text{решение не принято}\}$.

Данная задача заключается в минимизации суммы путевых расходов (расстояние в пути) для всех транспортных средств. Задача включает в себя следующие ограничения:

1. Только одно транспортное средство может посетить клиента;
2. Депо является началом и концом маршрута, откуда и куда должны вернуться все транспортные средства;
3. Вместимость транспортного средства меньше или равна максимальной вместимости.

Модель

Модель для задачи CVRP может быть сформулирована следующим образом:

Минимизировать

(1) $min \sum_{k\in K} \sum_{(i,j)\in E} c_{ij} x_{ij}^{k}$

При условии

(2) $\sum_{k\in K} \sum_{i\in V, i\neq j} x_{ij}^{k}, \forall j \in V\setminus\{0\}$

(3) $\sum_{j\in V\setminus\{0\}} x_{0j}^{k} = 1, \forall k \in K$

(4) $\sum_{i\in V,\, i\neq j} x_{ij}^{k} - \sum_{i\in V} x_{ji}^{k} = 0, \forall j \in V, \forall k \in K$

(5) $\sum_{i\in V} \sum_{j\in V\setminus\{0\}, i\neq j} q_j x_{ij}^{k} \leq Q, \forall k \in K$

(6) $\sum_{k\in K} \sum_{(i,j)\in S, i\neq j} x_{ij}^{k} \leq |S| - 1, S \subseteq V\setminus\{0\}$

(7) $x_{ij}^{k} \in \{0, 1\}, \forall k \in K, \forall (i, j) \in E$

1. Целевая функция (1) требуется, чтобы минимизировать транспортные расходы (сумму пройденного расстояния) всех транспортных средств.
2. Ограничение (2) требуется, чтобы в зависимости от местоположения клиента было только одно посещение транспортным средством.
3. Ограничение (3) требуется, чтобы покинуть склад;
4. Ограничение (4) требуется, чтобы количество въездов и выездов было одинаково.
5. Ограничение (5) требуется, чтобы вместимость каждого транспортного средства не превышала максимальную вместимость.
6. Ограничение (6) требуется для удаления промежуточных этапов
7. Ограничение (7) требуется для переменных принятия решения.

4.5 Постановка задачи «VRPTW»

Пусть есть набор идентичных транспортных средств V, набор клиентов C и ориентированный граф Q. Граф состоит из $|C| + 2$ вершин, где клиенты обозначены $1, 2,..., n$, а депо представлено вершинами 0 («начальное депо») и $n + 1$ («депо возвращения»). Стоит отметить, что при решении данной задачи использовалось только одно депо, которое являлось и началом и концом рассматриваемого подмаршрута. Множество всех вершин $\{0, 1, 2,..., n + 1\}$ обозначается за N. Набор рёбер, обозначенный A, представляет прямые связи между депо и клиентами, а также дуги между самими клиентами. Нет рёбер, заканчивающихся в вершине 0 или исходящей из вершины $n + 1$. С каждым ребром (i, j), где $i \neq j$ связаны стоимость c_{ij} и время t_{ij}, которые могут включать время обслуживания у клиента i.

Каждое транспортное средство обладает вместимостью q, а каждый клиент i обладает спросом d_i. У каждого клиента i есть собственное временное окно $[a_i, b_i]$, и транспортное средство должно прибыть к клиенту до времени b_i. Если транспортное средство прибудет до того, как откроется временное окно, то оно должно дождаться момента времени a_i для обслуживания клиента. Предполагается, что временные окна для обоих депо идентичны $[a_0, b_0]$, которые представляют горизонт планирования (т.е. насколько далеко простираются планы). Транспортные средства не могут покинуть склад до времени a_0 и должны вернуться не позднее момента времени b_0.

Предполагается, что q, a_i, b_i, d_i, c_{ij} являются неотрицательными целыми числами, а t_{ij}- целыми положительными числами. Заметим, что это предположение необходимо для разработки алгоритма кратчайшего пути с ограничениями ресурсов. Кроме того, предполагается, что неравенство треугольника выполняется как для c_{ij}, так и для t_{ij}.

Модель содержит два набора переменных решения x и s. Для каждого ребра (i, j), где $i \neq j, i \neq n + 1, j \neq 0$ и каждого транспортного средства k

определяется параметр x_{ijk} как

$x_{ijk} = \{1,$ если транспортное средство k может пройти путь от клиента i до клиента j; $0,$ в противном случае$\}$

Параметр решения s_{ik} определен для каждой вершины i и каждого транспортного средства k и обозначает время, когда транспортное средство k начинает обслуживать клиента i. В случае, если транспортное средство k не обслуживает клиента i, s_{ik} не имеет значения, и, следовательно, его значение считается несущественным. Предполагается, что $a_0 = 0$ и, следовательно, $s_{0k} = 0, \forall k$.

Цель состоит в том, чтобы составить набор маршрутов, который минимизирует общую стоимость, таким образом, чтобы

- каждый клиент обслуживался ровно один раз,
- каждый маршрут начинался в вершине 0 и заканчивался в вершине $n + 1$, и
- соблюдались временные рамки по обслуживанию клиентов и ограничению пропускной способности транспортных средств.

Модель

Задача CVRP-TW может быть сформулирована математически как проблема многомодового сетевого потока с временными окнами и ограничениями пропускной способности:

Минимизировать

(1) $min \sum_{k\in V} \sum_{i\in N} \sum_{j\in N} c_{ij} x_{ijk},$

При условии

(2) $\sum_{k\in V} \sum_{j\in N} x_{ijk} = 1, \ \forall i \in C,$

(3) $\sum_{i\in C} d_i \sum_{j\in N} x_{ijk} \le q, \ \forall k \in V$

(4) $\sum_{j \in N} x_{0jk} = 1,\ \forall k \in V$

(5) $\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0,\ \forall h \in C, \forall k \in V$

(6) $\sum_{i \in N} x_{i,n+1,k} = 1,\ \forall k \in V$

(7) $x_{ijk}(s_{ik} + t_{ij} - s_{jk}) \leq 0,\ \forall i, j \in N, \forall k \in V$

(8) $a_i \leq s_{ik} \leq b_i,\ \forall i \in N, \forall k \in V$

(9) $x_{ijk} \in \{0, 1\},\ \forall i, j \in N, \forall k \in V$

1. Целевая функция (1) минимизирует общую стоимость поездки.
2. Ограничение (2) обеспечивает, что каждый клиент посещен ровно один раз.
3. Ограничение (3) указывает, что транспортное средство не может быть загружено больше его вместимости.
4. Уравнения (4), (5) и (6) указывают, что каждое транспортное средство должно покинуть депо 0; после прибытия транспортного средства к клиенту оно должно отправиться в другой пункт назначения; и, наконец, все транспортные средства должны прибыть в депо n + 1.
5. Неравенство (7) устанавливает взаимосвязь между временем отправления транспортного средства от клиента и его непосредственным преемником.
6. Ограничение (8) требует, чтобы временные окна были соблюдены.
7. Услоиве (9) является ограничениями целочисленности.

Стоит отметить, что не используемое транспортное средство моделируется путем движения по «пустому» маршруту (0, n + 1).

Для каждого транспортного средства переменные начала обслуживания задают уникальное направление маршрута, тем самым устраняя любые промежуточные маршруты. Следовательно, классические ограничения исключения подтуров задачи VRP становятся излишними. Наконец, целевая функция (1) повсеместно использовалась при решении CVRP-TW до оптимальности. В исследованиях по эвристике было принято сводить к минимуму количество транспортных средств, которое может привести к дополнительным расходам на проезд.

После изучения постановок задач, далее будут рассмотрены алгоритмы, которые реализованы для их решения.

4.6 Алгоритм «Имитации отжига»

Алгоритм «Имитации отжига» (**Simulated Annealing**) интерпретируется как медленное уменьшение вероятности принятия худших решений по мере исследования пространства решений. Принятие худших решений позволяет проводить более широкий поиск глобального оптимального решения.

В общем случае алгоритмы имитационного отжига работают следующим образом:

Температура постепенно снижается от начального положительного значения до нуля. На каждом временном шаге алгоритм случайным образом выбирает решение, близкое к текущему, измеряет его качество и переходит к нему в соответствии с зависящими от температуры вероятностями выбора лучших или худших.

Опишем один шаг алгоритма «Имитации отжига» для оптимизации рассматриваемого подмаршрута:

1. Выбирается некоторое начальное решение S, т.е. это маршрут полученный путем случайным перемешиванием городов.
2. Задается параметр T, который называется *начальной температурой* и некоторый коэффициент $r \in (0, 1)$, который требуется для уменьшения начальной температуры. Также задается *минимальная температура* T_{min} это температура, до которой опускается температура T.
3. Сравниваются начальная и минимальная температуры: $T \geq T_{min}$. Если данное условие не выполняется, то алгоритм заканчивает свою работу и

происходит переход к шагу 1 для оптимизации следующего подмаршрута, иначе осуществляется переход к шагу 4.
4. Происходит следующая модификация маршрута. В маршруте случайным образом выбирается 2 города, между которыми инвертируются города за исключением выбранных двух городов. Тем самым получается новый маршрут S'.
5. Вычисляется разность расстояний старого маршрута и модифицированного: $\Delta = dist(S) - dist(S')$.
6. Если $\Delta \leq 0$, то старый маршрут заменяется на новый: $S := S'$ и далее происходит переход к шагу 8.
7. Если $\Delta > 0$, то вычисляется вероятность по формуле: $e^{-\frac{\Delta}{T}}$, и случайным образом выбирается число $p \in (0; 1)$. Если $e^{-\frac{\Delta}{T}} < p$, то $S := S'$, в противном модифицированный маршрут никак не учитывается. Далее осуществляется переход к шагу 8.
8. Происходит обновление начальной температуры: $T := T \cdot r$ и осуществляется переход к шагу 3.

Преимущества и недостатки метода

Из преимуществ можно отметить следующее:

1. Относительно высокая скорость поиска решения.
2. Метод достаточно прост в реализации.

Из недостатков отметим следующее:

1. Алгоритм находит локальное оптимальное решение, которое не обязательно будет являться глобальным.
2. В алгоритме присутствует несколько мест, где используется «random», что влияет на итоговое решение. Например,
    a. случайным образом перемешивается начальный маршрут;
    b. случайным образом выбираются 2 города, между которыми инвертируются все города;
    c. случайным образом выбирается вероятность принятия/непринятия неоптимального решения.

Данный алгоритм имеет много весомых недостатков, поэтому после изучения и реализации алгоритма «Имитации отжига» возникла потребность рассмотреть точный алгоритм, с которым можно было бы сравнить полученные результаты. Для экспериментов была использована коммерческая реализация алгоритма решения задачи целочисленного программирования «Gurobi», о котором известно, что он базируется на целочисленном программировании и в нем используются методы, аналогичные методам «Ветвям и границам», поэтому следующим этапом обзора заключалось изучение данного метода.

4.7 Метод «Ветвей и границ»

Метод «Ветвей и границ» («Branch & cut») является вариантом известного алгоритма ветвей и границ Лэнда и Дойга [1960 г.], наиболее широко используемого алгоритма для решения многих видов невыпуклых задач оптимизации. Метод «Ветвей и границ» относится к группе комбинаторных методов и позволяет существенно уменьшить объем перебора вариантов. Рассмотрим постановку Данцига-Фалкерсона-Джонсона:

Минимизировать

1. $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \rightarrow min_x, x_{ij} \in \{0, 1\},$

При условии

2. $\sum_{i=1}^{n} x_{ij} = 1, \forall j,$
3. $\sum_{j=1}^{n} x_{ij} = 1, \forall i,$
4. $\sum_{i \in Q}\sum_{j \in Q, j \neq i} x_{ij} \leq |Q| - 1, |Q| \geq 2,$

где x_{ij} принимает значения 0 или 1 в зависимости от того, входит ли ребро из вершины i в вершину j, c_{ij} - расстояние между вершинами i и j. Условия (2), (3) гарантируют, что из каждой вершины входит и выходит ровно одно ребро. Условие (4) гарантирует, что полученное решение является единым циклом, проходящем через все вершины, а не объединением непересекающихся циклов.

Основная идея решения данной задачи с помощью метода «Ветвей и границ» заключается в том, чтобы ослабить ограничения на переменные в постановке задачи целочисленного программирования до линейных, затем решить задачу в новой постановке и если решение получилось целочисленное, то получить ответ. Если же решение не является целочисленным, то можно выбрать одну из переменных и решить относительно нее две постановки - изначальные ослабленные условия, а также равенство нулю в одной постановке и единице в другой. Иной способ приблизить к целочисленному решению это отделить полученное вещественное решение гиперплоскостью от возможных целочисленных, т.е. создав так называемую «границу». Итеративное решение с использованием указанных приемов составляет метод «Ветвей и границ», который позволяет получать точные решения.

Изучив, данный метод была подготовлена его реализация с использованием коммерческого проекта «Gurobi». Следующим этапом возникла необходимость изучить и реализовать один из самых лучших и точных эвристических алгоритмов, а именно «Эвристика Лина-Кёрнигана» для сравнения с результатами, полученными «SA» и «Gurobi».

4.8 «Эвристика Лина-Кёрнигана»

Эвристика Лина-Кернигана относится к классу так называемых алгоритмов локальной оптимизации. Алгоритм задается в терминах *opt* (обмена или ходы), которые могут преобразовать один маршрут в другой. При наличии осуществимого маршрута алгоритм неоднократно выполняет обмены, которые сокращают продолжительность текущего маршрута, пока не будет достигнут маршрут, для которого никакой обмен не приведет к улучшению. Этот процесс может повторяться много раз с начальных маршрутов, сгенерированных каким-либо рандомизированным способом.

В алгоритме предполагается, что некоторое начальное разбиение маршрута на подмаршруты уже существует, затем имеющееся приближение улучшается в течение некоторого количества итераций. Применяемый способ улучшения состоит в обмене вершинами в каждом подмаршруте в отдельности.

В описании общей схемы алгоритма будет использовано выражение «оптимизация подмаршрута», поэтому стоит заранее прояснить, что подразумевается под «*оптимальностью*» подмаршрута:

Маршрут считается λ–*оптимальным (или просто оптимальным)*, если невозможно получить более короткий маршрут, заменив любое его λ рёбер любым другим набором λ рёбер.

Из этого определения следует, что любой λ - оптимальный маршрут также является λ' - оптимальным для $1 \leq \lambda' \leq \lambda$. Также легко видеть, что маршрут, содержащий n городов, оптимален, если и только если этот маршрут n-оптимален.

В общем, чем больше значение λ, тем больше вероятность того, что последний маршрут оптимален. При достаточно больших λ кажется, по крайней мере интуитивно, что λ - оптимальный маршрут должен быть оптимальным.

Теперь опишем общую схему алгоритма на примере оптимизации рассматриваемого подмаршрута:

Пусть T текущий подмаршрут. На каждой итерации алгоритм пытается найти два набора рёбер, $X = \{x_1, \dots, x_r\}$ и $Y = \{y_1, \dots, y_r\}$, таким образом, что, если рёбра X будут удалены из T и заменены рёбрами Y, то результатом будет лучший маршрут. Этот обмен ссылками называется движением *r-opt*.

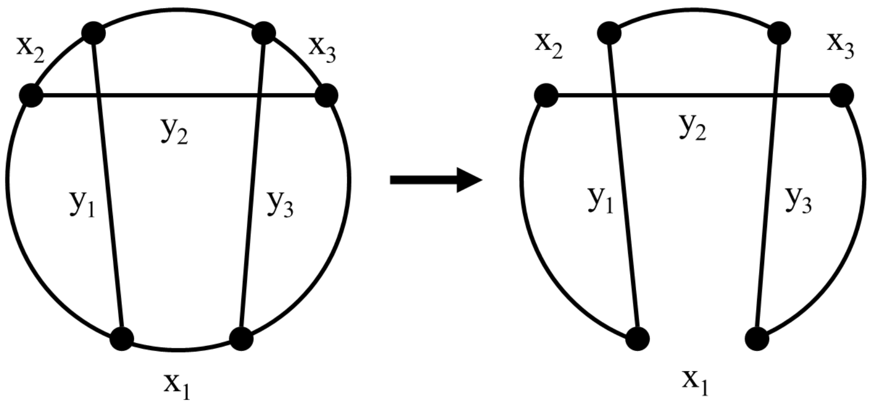


Рисунок 1. Движение 3-opt

Два набора X и Y строятся поэлементно. Изначально X и Y пусты. На шаге i пара рёбер, x_i и y_i, добавляются в X и Y соответственно.

Для достижения достаточно эффективного алгоритма в X и Y могут входить только рёбра, соответствующие следующим критериям:

(1) Критерий последовательного обмена

Рёбра x_i и y_i должны иметь общую конечную точку, как и y_i и x_{i+1}. Если t_1 обозначает один из двух концов x_1, то в общем случае имеем: $x_i = (t_{2i-1}, t_{2i})$, $y_i = (t_{2i}, t_{2i+1})$ и $x_{i+1} = (t_{2i+1}, t_{2i+2})$ для $i \geq 1$. Смотреть Рисунок 2.

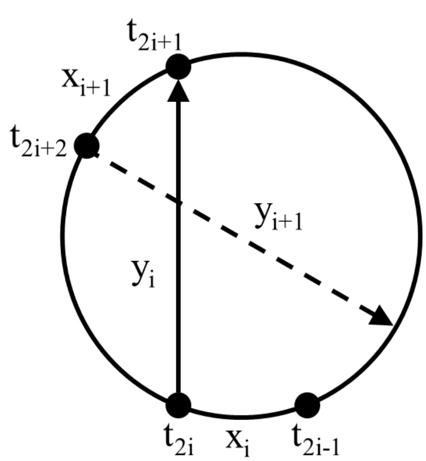


Рисунок 2. Ограничение выбора x_i, y_i, x_{i+1}, и y_{i+1}

Как видно, последовательность $(x_1, y_1, x_2, y_2, \dots, x_r, y_r)$ представляет собой цепочку смежных рёбер.

Необходимым (но не достаточным) условием того, что обмен рёбрами X с рёбрами Y приведет к замкнутому маршруту, т. е. $y_r = (t_{2r}, t_1)$.Такой обмен называется *последовательным* (***sequential***).

Как правило, улучшение маршрута может быть достигнуто путем последовательного обмена соответствующей нумерацией затронутых рёбер. Однако это не всегда так. На рисунке 3 показан пример, в котором последовательный обмен невозможен.

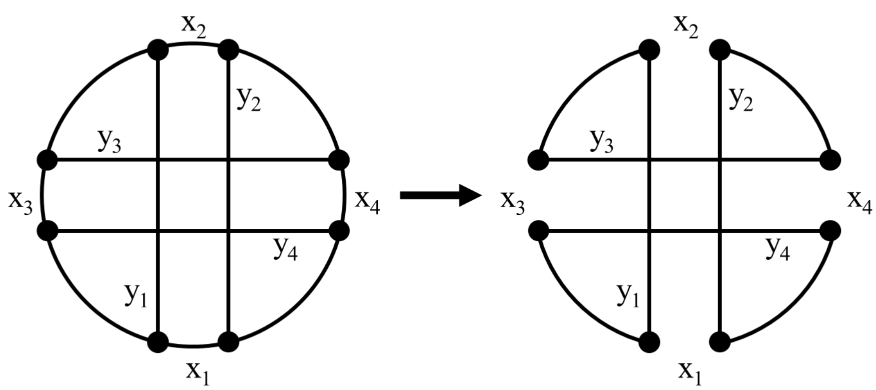


Рисунок 3. Последовательный обмен (r = 4)

(2) Критерий осуществимости

Требуется, чтобы $x_i = (t_{2i-1}, t_{2i})$ всегда выбирался так, чтобы, если t_{2i} соединяется с t_1, то итоговая конфигурация - это маршрут. Этот критерий осуществимости используется для $i \geq 3$ и гарантирует, что можно вплотную приблизиться к замкнутому маршруту. Этот критерий был включен в алгоритм как для сокращения времени выполнения, так и для упрощения реализации.

(3) Критерий положительного усиления

Требуется, чтобы ребро y_i всегда выбиралось так, чтобы коэффициент усиления (***gain***), G_i из предложенного набора обменов являлся положительным. Предположим, что $g_i = c(x_i) - c(y_i)$ является улучшением при замене x_i на y_i. Тогда $G_i = g_1 + g_2 + \ldots + g_i$.

Этот критерий остановки играет большую роль в эффективности алгоритма. Требование, чтобы каждая частичная сумма, G_i, была положительной, сразу кажется, что это слишком ограничительно. Однако то, что это не так, следует из следующего простого факта:

Если последовательность чисел имеет положительную сумму, существует циклическая перестановка этих чисел, так что каждая частичная сумма положительна.

(4) Критерий дизъюнктивности

Наконец, требуется, чтобы множества X и Y не пересекались. Это упрощает реализацию, сокращает время выполнения и обеспечивает эффективный критерий остановки.

После подробного описания алгоритма для оптимизации рассматриваемого подмаршрута составим его краткую версию:

1. В начальном маршруте случайным образом перемешиваются города. Тем самым получаем новый маршрут T.
2. Пусть i = 1. Выбирается точка t_1.
3. Выбирается ребро $x_1 = (t_1, t_2) \in T$.
4. Выбирается ребро $y_1 = (t_2, t_3) \notin T$ так, чтобы $G_1 > 0$. Если это невозможно, переходим к шагу 12.
5. Пусть i = i + 1
6. Выбирается $x_i = (t_{2i-1}, t_{2i}) \in T$ так, чтобы

a. Если точка t_{2i} соединена с точкой t_1, тогда исходом в результате будет маршрут T';
b. Для всех $s < i$ выполняется $x_i \neq y_s$.

Если тур T' лучше маршрута T, то $T = T'$, и осуществляется переход к шагу 2.

7. Выбирается ребро $y_i = (t_{2i}, t_{2i+1}) \notin T$ так, чтобы
   a. $G_i > 0$;
   b. Для всех $s \leq i$ выполняется $y_i \neq x_s$;
   c. Существует x_{i+1}.

Если такой y_i существует, то осуществляется переход к шагу 5.

8. Если есть неиспытанный вариант для y_2, пусть i = 2, то осуществляется переход к шагу 7.
9. Если есть неиспытанный вариант для x_2, пусть i = 2, то осуществляется переход к шагу 6.
10. Если есть неиспытанный вариант для y_1, пусть i = 1, то осуществляется переход к шагу 4.
11. Если есть неиспытанный вариант для x_1, пусть i = 1, то осуществляется переход к шагу 3.
12. Если есть неиспытанный вариант для t_1, пусть i = 1, то осуществляется переход к шагу 2.
13. Если оптимизация данного подмаршрута закончена, то осуществляется переход к шагу 1 для оптимизации следующего подмаршрута.

Комментарии к алгоритму:

Шаг 1.

В качестве отправной точки для исследований выбирается случайный подмаршрут, который далее оптимизируется по алгоритму.

Шаг 3.

Пусть выбирается ребро $x_1 = (t_1, t_2)$ в маршруте. Когда выбрана точка t_1, то возникает два варианта для построения x_1. Здесь глагол «выбрать» означает «выбрать неиспытанный вариант». Однако каждый раз, когда было найдено улучшение тура (на шаге 6), все альтернативы считаются неиспробованными.

Шаг 6.

Существует два варианта x_i. Однако для данного y_{i-1}при $i \geq 2$ только один из них позволяет «замкнуть» маршрут путем добавления y_i. Другой выбор приводит к двум несвязанным маршрутам. Однако только в одном случае допускается такой неосуществимый выбор, а именно при i = 2. На рисунке продемонстрирована данная ситуация.

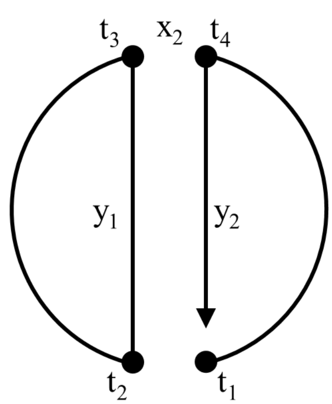


Рисунок 4. Отсутствие замкнутости ребром x_2

Если y_2 выбран так, чтобы t_5 лежал между t_2 и t_3, то маршрут может быть замкнут на следующем шаге. Но тогда t_6 может находиться по обе стороны от t_5 (см. Рисунок 5).

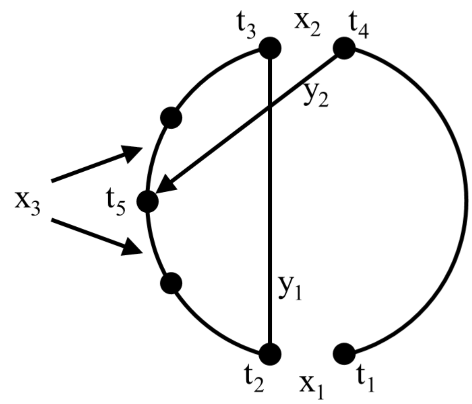


Рисунок 5. Два случая для x_3

С другой стороны, если y_2 выбран так, чтобы t_5 лежал между t_4 и t_2, для t_6 есть только один выбор (он должен лежать между t_4 и t_5), а t_7 должен лежать между t_2 и t_3. Но тогда t_8 может находиться по обе стороны от t_7.

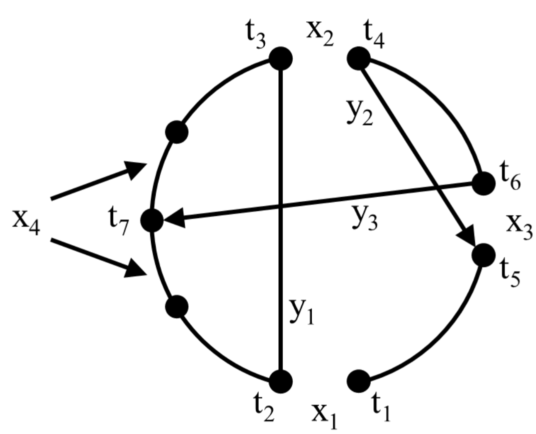


Рисунок 6. Один случай для x_3. Ограниченный выбор для y_3. Два варианта для x_4.

Условие (b) на шаге 6 и шаге 7 гарантирует, что множества X и Y не пересекаются: y_i не должен быть ранее разорванной связью, а x_i не должен быть ранее добавленной связью.

Шаги 8-12

Эти шаги вызывают откат назад. Стоит обратить внимание, что возврат разрешен только в том случае, если улучшения не обнаружено, и только на уровнях 1 и 2.

Шаг 13

Алгоритм завершается маршрутом решения, когда все значения t_i были проверены без улучшения. При необходимости на шаге 1 может быть рассмотрен новый случайный начальный маршрут.

После обзора алгоритма LKH мной были реализованы алгоритмы «2-opt» и «3-opt», являющиеся наиболее часто используемыми при решении задач оптимизации маршрутов.

4.9 Эксперименты

Эксперименты проводились на наборах данных города Алматы. Всего составлено 6 наборов данных, т.е. 3 набора данных для задачи CVRP и 3 набора данных для задачи CVRP-TW. Каждая задача в первых наборах включает в себя 20 клиентов и депо, каждая задача во вторых 50 клиентов и депо, и в третьих - 100 клиентов и депо.

Файл с данным содержит в себе:

- первый и второй столбец это широта и долгота клиенов и депо;
- третий столбец состоит и весов грузов необходимых для доставки. Данный параметр у депо = 0. Важно отметить,что в наборах данных встречались веса грузов больше максимальной вместимости и было принято решение не учитывать данные точки и исключить их из построения нового маршрута.
- Если решается задача CVRP-TW то добавляется еще 3 столбца (время в каждом столбце выражено в минутах):
  - первый столбец содержит левую границу временного окна, раньше которого груз не может быть доставлен клиенту. Данный параметр выражен в минутах, а у депо = 0;
  - второй столбец содержит правую границу временного окна, позже которого груз не может быть доставлен клиенту. Данный параметр выражен в минутах, а у депо = 0;

- третий столбец содержит время обслуживания клиента. Данный параметр выражен в минутах, а у депо = 0.

На решение каждой задачи в каждом наборе данных отводилось по 100 секунд

Перед тем, как сравнивать метаэвристические алгоритмы с коммерческим проектом «Gurobi» необходимо было узнать какой стоит использовать алгоритм для сравнений «2-opt» или «3-opt». Поэтому были проведены эксперименты алгоритмов «2-opt» и «3-opt» для решения задач CVRP и CVRP-TW.

Стоит отметить, что оба алгоритма сходятся практически к одинаковому решению. На наборах 20, 50 точек разницы между ними не будет видно, так как большинство оптимизаций происходит меньше,чем за секунду и маршруты достаточно маленькие. Однако на 100 точках данная разница заметна, хоть и не так ярко выражена. Оба алгоритма практически за одинаковое время решают задачи, но 3-opt это делает лучше, что продемонстрировано на графике. А также это из-за того, что у 2-opt меньше вариантов изменения маршрута, чем у 3-opt. Приведем это сравнение на примере 3 подмаршрутов.

Для 2-opt:

1. $O_r\, O\, O$
2. $O\, O_r\, O$
3. $O\, O\, O_r$,

где **O** - подмаршрут, **r** - реверсирование подмаршрута

Для 3-opt помимо 3 случаев в 2-opt добавляется еще 4 случая:

4. $O\, [O\, O]$
5. $O_r\, [O\, O]$
6. $O\, [O_r\, O]$
7. $O\, [O\, O_r]$,

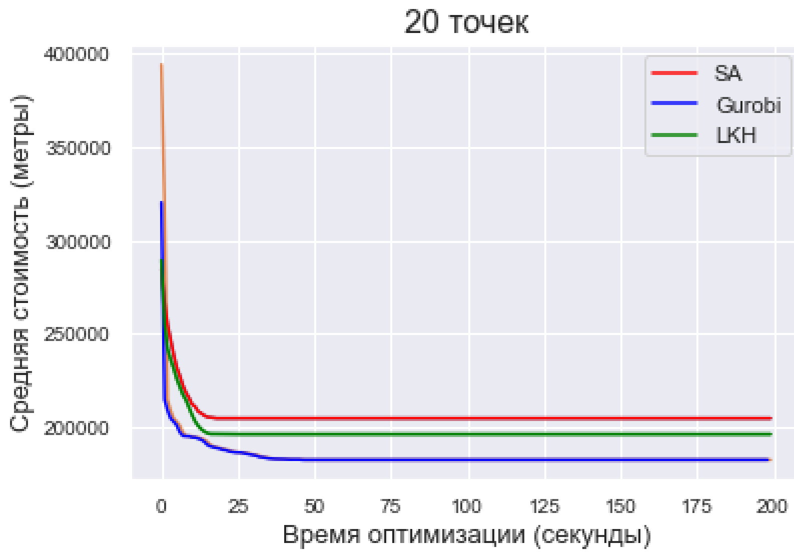
где **[O, O]** - перестановка двух подмаршрутов местами.

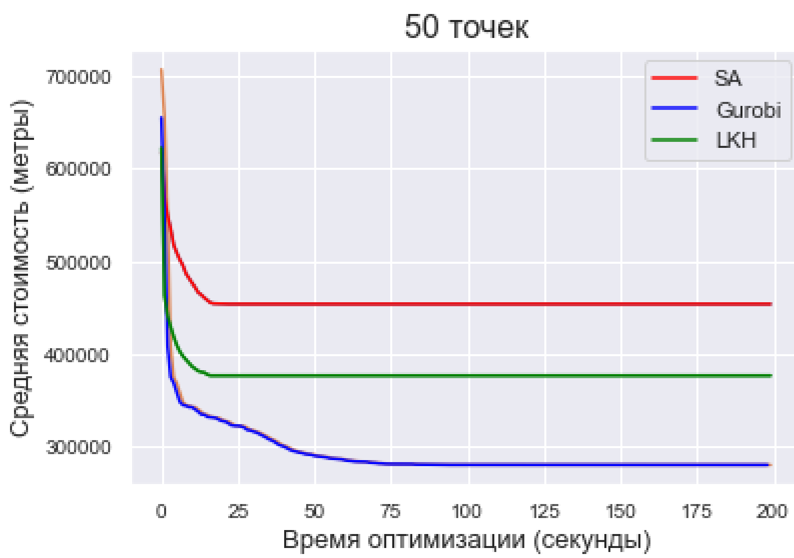
Стоит отметить, что приоритет операции реверсирования выше, чем приоритет операции перестановки двух подмаршрутов.

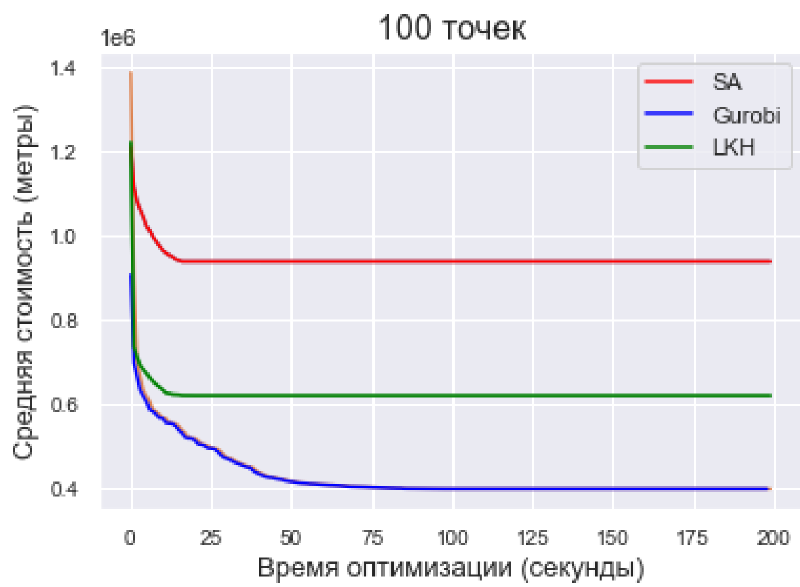
Таким образом, для дальнейших экспериментов в качестве алгоритма LKH будет выступать алгоритм 3-opt.

Следующий этап экспериментов заключался в сравнении «SA», «LKH» и коммерческого проекта «Gurobi» для задачи CVRP. Важно отметить, что вместимость транспортного средства для

- 20 городов равна 30;
- 50 городов равна 40;
- 100 городов равна 50


20 точек
400000
350000
300000
250000
200000
Средняя стоимость (метры)
0
25
50
75
100
125
150
175
200
Время оптимизации (секунды)
SA
Gurobi
LKH


50 точек
700000
600000
500000
400000
300000
Средняя стоимость (метры)
0
25
50
75
100
125
150
175
200
Время оптимизации (секунды)
SA
Gurobi
LKH



По полученным результатам можно сделать выводы:

- Для маленького количества клиентов (20 клиентов) алгоритм «SA» демонстрирует неплохое решение, приближенное к «Gurobi». Однако с увеличением количества городов, например как видно для случаев 50, 100 городов, результат ухудшается, поэтому для больших задач данный алгоритм не подходит.
- Коммерческий проект «Gurobi», в основе которого находится метод «Ветвей и границ», превосходит по результатам метаэвристические алгоритмы «LKH» и «SA». Данный вывод явно можно сделать для случаев 50 и 100 городов. Однако для получения итогового решения метаэвристическим алгоритмам требуется гораздо меньше времени. По графикам видно, что в среднем на оптимизацию требуется не больше 25 секунд, в то время «Gurobi» требуется около 50-75 секунд. На задачах большего размера данная разница во времени будет только увеличиваться. Таким образом, подтверждается тот факт, что хоть метаэвристические алгоритмы дают приближенное решение, однако им требуется в несколько раз меньше времени на решение задачи, чем MIP алгоритмам.

Программная реализация

В данной главе изложено описание программной реализации.Экспериментальный стенд можно разделить не следующие части:

1. Реализация функции для обработки наборов данных отдельно для задач *CVRP* и *CVRP-TW*.
2. Реализация алгоритмов, приведенных в разделах 4.6, 4.7, 4.8. Стоит отметить, что вместо реализации метода «Ветвей и границ» использовался коммерческий проект Gurobi, который необходимо было настроить для решения задач *CVRP* и *CVRP-TW* на обработанных данных.
3. Реализация интерфейса для функций реализованных алгоритмов и средства визуализации результатов.

Экспериментальный стенд написан на языках программирования Python и C.

Также на языке Python реализована среда, которая представлена несколькими классами.

1. Класс *VRP*
   Данный класс использовался для объявления следующих переменных:
   - Имя файла с данными;
   - Путь к папке, в которой будет находится бинарный файл с матрицей расстояний между всеми клиентами, а также между клиентами и депо;
   - Количество городов;
   - Количество итераций для оптимизации маршрута.

2. Класс *CVRP*
   Для решения задачи CVRP добавляется новый параметр capacity, отвечающий за максимальную вместимость транспортного средства. В данном классе реализованы:
   - parse_file() - функция обработки данных;
   - sa(), lkh(), gurobi() - функции, использующие рассмотренные в обзоре алгоритмы, решающие задачу CVRP.

По окончании работы генерируется лог с данными, состоящий из двух колонок:
- В первом столбце записывается длина маршрута в определенный момент времени;
- Во втором столбце записывается время, которое потребовалось, чтобы оптимизировать маршрут до некоторой длины.
- Решенная задача отделяется пустым пространством,после чего записываются данные для следующей задачи. Это сделано для того,чтобы при построении графиков данные, относящиеся к разным задачам не пересекались и не перемешивались.

3. Класс *CVRP-TW*

   Данный класс отличается от класса CVRP тем, что добавляются еще два новых параметра:
   - Время начала решения оптимизационной задачи маршрута;
   - Время окончания решения оптимизационной задачи маршрута.

   Так как добавилось два новых параметра, то изменены функция обработки данных, а также рассмотренные в обзоре алгоритмы, которые уже решают задачу *CVRP-TW.*

Функции, реализующие алгоритмы «SA», «LKH», находятся в файлах logistic.h, cvrptw-logistic.c. Функции, использующие коммерческий проект «Gurobi», расположены в gurobi_CVRP.py, gurobi_CVRPTW.py.

Общая величина строк экспериментального стенда:

Заключение

Основным результатом данной работы являлось ознакомление с эвристическии и MIP алгоритмами, а также демонстрация конкурценции скорости и качества между ними для 20, 50 и 100 точек.

В дальнейшем планируется углубленное изучение обучения с подкреплением и разрработка собственной модели машинного обучения для решения задач CVRP и CVRP-TW, а также сравнение собственной разработанной модели с полученными результатами в данном обзоре и другими алгоритмами машинного обучения, решающих задачи CVRP и CVRP-TW.

Список литературы

1. https://tvim.info/files/56_72_Shcherbina.pdf
2. https://www.researchgate.net/publication/281003471_Vehicle_Routing_Problem_solution_using_Mixed_Integer_Linear_Programming
3. Czech, Z.J. and Czarnas, P., 2002, January. Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings 10th Euromicro workshop on parallel, distributed and network-based processing* (pp. 376-383). IEEE.
4. Kallehauge, B., Larsen, J., Madsen, O.B. and Solomon, M.M., 2005. Vehicle routing problem with time windows. In *Column generation* (pp. 67-98). Springer, Boston, MA.
5. Thangiah, S.R., Osman, I.H. and Sun, T., 1994. Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows. *Computer Science Department, Slippery Rock University, Technical Report SRU CpSc-TR-94-27*, *69*.
6. Helsgaun, K., 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European journal of operational research*, *126*(1), pp.106-130.
7. Talbi, El-Gh. Metaheuristics for Bi-level Optimization. Springer Publishing Company, Incorporated, 2013. 306 p.