МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)
01.03.02. Прикладная математика и информатика
(код и наименование направления подготовки, специальности)
Системное программирование и компьютерные технологии
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Применение эвристического алгоритма для решения задачи маршрутизации транспорта в условиях ограничений по грузоподъемности

Студент И.Е. Неплохов
(И.О. Фамилия) (личная подпись)

Руководитель А.В. Очеповский
(И.О. Фамилия) (личная подпись)

Консультант Н.В. Ященко
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2018

АННОТАЦИЯ

Тема бакалаврской работы: «Применение эвристического алгоритма для решения задачи маршрутизации транспорта в условиях ограничений по грузоподъемности».

Бакалаврская работа посвящена разработке программного кода для решения задачи маршрутизации транспорта с помощью эвристического алгоритма. Используемый язык программирования – Java.

Объект исследования бакалаврской работы – вычислительный процесс нахождения оптимального пути эвристическим алгоритмом в условиях ограничений по грузоподъемности.

Предмет исследования бакалаврской работы – программа, определяющая оптимальный путь с помощью эвристического алгоритма для решения задачи маршрутизации транспорта.

Цель бакалаврской работы – применить эвристический алгоритм для решения задачи маршрутизации транспорта в условиях ограничений по грузоподъемности.

Для достижения цели работы необходимо решить следующие задачи:

- программная реализация алгоритма оптимизации с помощью эвристического алгоритма;
- вычислительный эксперимент;
- анализ данных вычислительного эксперимента.

Бакалаврская работа состоит из введения, трех глав, заключения, списка используемых источников.

В результате работы была разработана программа, решающая задачу маршрутизации транспорта при помощи эвристического алгоритма, в условиях ограничений по грузоподъемности. Проведен вычислительный эксперимент.

Бакалаврская работа представлена на 46 страницах, включает 22 иллюстрации, 3 таблицы, 9 формул, список используемой литературы, состоящий из 27 источников.

ABSTRACT

This graduation work deals with the use of a heuristic algorithm for solving the vehicle routing problem(VRP) by capacity limitation conditions.

The aim of the work is to use a heuristic algorithm to solve the VRP with a constraint.

The object of the graduation work is the computational process of finding the optimal path by a heuristic algorithm with a constraint on the carrying capacity.

The subject of the graduation work is the program that determines the optimal path using a heuristic algorithm, to solve the problem of transport routing.

To achieve the goal of the work the following tasks were solved:

1) software implementation of the optimization algorithm with the help of a heuristic algorithm;

2) computational experiment;

3) analysis of data from a computational experiment.

This graduation work is actual, because since even if there are a plenty of earlier implemented heuristic algorithms, it is necessary to further implement and improve these algorithms to find the best result, because the use of automated systems in the field of transport logistics is one of the ways to save resources when transporting goods.

The main goal in these tasks is to build routes for vehicles that serve a certain number of customers. The algorithms that solve the VRP are divided into exact, heuristic and meta-heuristic algorithms. A heuristic algorithm was used in this work. Heuristic methods are called methods, that based on a certain rule (heuristics), which are not always the result of strict mathematical principles, that give a solution close to the exact one.

In conclusion, a program was developed that solves the vehicle routing problem with heuristic algorithm by capacity limitation conditions. A computer experiment was carried out.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5
ГЛАВА 1 ОБЗОР СУЩЕСТВУЮЩИХ СПОСОБОВ РЕШЕНИЯ ЗАДАЧ МАРШРУТИЗАЦИИ ТРАНСПОРТА 7
1.1 Описание задачи маршрутизации транспорта 7
1.2 Алгоритмы для решения ЗМТ 8
1.2.1 Точные алгоритмы 9
1.2.2 Эвристические алгоритмы 10
1.2.3 Метаэвристические алгоритмы 17
1.3 Ограничения в задачах маршрутизации транспорта 20
1.4 Математическая модель задачи маршрутизации транспорта 22
ГЛАВА 2 РЕАЛИЗАЦИЯ ЭВРИСТИЧЕСКОГО АЛГОРИТМА ДЛЯ РЕШЕНИЯ ЗМТ 25
2.1 Обоснование выбора технических средств 26
2.2 Исходные данные 27
2.3 Реализация алгоритма 32
2.4 Проведение вычислительного эксперимента 36
ЗАКЛЮЧЕНИЕ 38
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 39
ПРИЛОЖЕНИЕ А. Диаграмма классов 42

ВВЕДЕНИЕ

Каждый год спрос на транспортно-логистические услуги повышается по всему миру. В следствии этого, проблема решения задач маршрутизации транспорта становится все более актуальной и востребованной. Главная цель в рассматриваемых задачах заключается в построение маршрутов для транспортных средств, которые обслуживают заданное количество клиентов.

Количество видов задач маршрутизации транспорта является очень большим, как и алгоритмов, решающих их. Это точные, эвристические и метаэвристические алгоритмы. Т.к. задача маршрутизации транспорта является NP-трудной [26], очевидно, что точные подходы целесообразно использовать только при малоразмерных задачах. Поэтому, для решения задач маршрутизации транспорта предпочтительней и чаще всего используются эвристические алгоритмы. Они создают решения приближенные к оптимальному, но за меньшее время (по сравнению с точными методами). Еще одна особенность эвристических алгоритмов проявляется в многообразие решений, которые получаются в ходе работы алгоритма по одному и тому же примеру.

Таким образом, **актуальность** темы бакалаврской работы обусловлена тем, что даже при наличии большого количества уже реализованных эвристических алгоритмов, необходимо и дальше реализовывать и улучшать данные алгоритмы для поиска наилучшего результата, т.к. применение автоматизированных систем в области транспортной логистики – один из способов сэкономить ресурсы.

Данная бакалаврская работа отличается высокой **практической значимостью**. В ходе его создания была разработана программа, решающая задачу маршрутизации транспорта в условиях ограничений по грузоподъемности, позволяющая сделать процесс выбора оптимального пути наиболее результативным.

Объект исследования бакалаврской работы – вычислительный процесс нахождения оптимального пути эвристическим алгоритмом в условиях ограничений по грузоподъемности.

Предмет исследования бакалаврской работы – программа, определяющая оптимальный путь с помощью эвристического алгоритма для решения задачи маршрутизации транспорта.

Цель бакалаврской работы – применить эвристический алгоритм для решения задачи маршрутизации транспорта в условиях ограничений по грузоподъемности.

Для достижения цели работы необходимо решить следующие задачи:

1) программная реализация алгоритма оптимизации с помощью эвристического алгоритма;
2) вычислительный эксперимент;
3) анализ данных вычислительного эксперимента.

Первая глава описывает задачу маршрутизации транспорта, алгоритмы, которые решают данную задачу.

Вторая глава посвящена разработке алгоритма и программе, использующая данный алгоритм с дополнительными условиями, подводятся результаты работы разработанного алгоритма.

ГЛАВА 1 ОБЗОР СУЩЕСТВУЮЩИХ СПОСОБОВ РЕШЕНИЯ ЗАДАЧ МАРШРУТИЗАЦИИ ТРАНСПОРТА

1.1 Описание задачи маршрутизации транспорта

Задача маршрутизации транспорта (ЗМТ) – это комбинаторная задача оптимизации и целочисленного программирования, которая задается вопросом: «Каков оптимальный набор маршрутов для прохождения парка транспортных средств для доставки к определенному набору клиентов?». Он обобщает известную задачу коммивояжера (ЗК).

Впервые понятие ЗМТ появляется в статье [1]. Авторы данной статьи в 1959 году поставили математическую формулировку, а также решили задачу о поставке бензина на заправочные станции. Даже на сегодняшний день эта задача является одной из самых известных в области комбинаторной оптимизации.

Несмотря на то, что ЗМТ является развитием ЗК, и эти задачи родственны, необходимо заметить, что в ЗМТ принята терминология, которая расходится от терминологии ЗК. В случае если в ЗК применяют подобные понятия как «коммивояжёр», «город» и «путь», то в ЗМТ «коммивояжёр» заменён на «транспортное средство», «город» — на «вершину», «путь» — на «маршрут». В отдельных вариантах термин «вершина» заменяется синонимом «клиент». Кроме этого, в ЗМТ появляется ещё одно вспомогательное слово — «депо». Это такая некая особенная вершина, где начинаются и заканчиваются маршруты абсолютно всех транспортных средств. Как правило, депо предполагает из себя склад с целью хранения продукции, нужной для развозки и доставки клиентам. Во всех вариантах ЗМТ предполагается наличие не менее одного депо. В отдельности акцентируется вариация задачи, в которой рассматриваются случаи с более чем одним депо. В случае если в задаче только лишь одно депо, то всетранспортные средства должны начинать путь с него и заканчивать его в этом же депо. Другими словами, все маршруты должны включать в себя вершину депо. Если же депо несколько, то каждый маршрут

должен содержать вершину только одного из них. В задаче присутствует такое понятие, как стоимость объезда последовательности вершин транспортным средством. Это ключевой параметр, требующий минимизации. На практике он выражает любые затраты посещения клиентов и может представлять как стоимость потребляемого топлива, так и время, требуемое для выполнения работы. Для исследования алгоритмов не важно, что за ним скрывается в действительности, примем его как обобщение всех видов затрат на передвижение. Но нужно учитывать, что стоимость проезда не может быть отрицательной.

1.2 Алгоритмы для решения ЗМТ

Алгоритмы, решающие ЗМТ, разделяют на точные, эвристические и метаэвристические, как показано на рисунке 1. Проведем краткий обзор по распространенным алгоритмам.

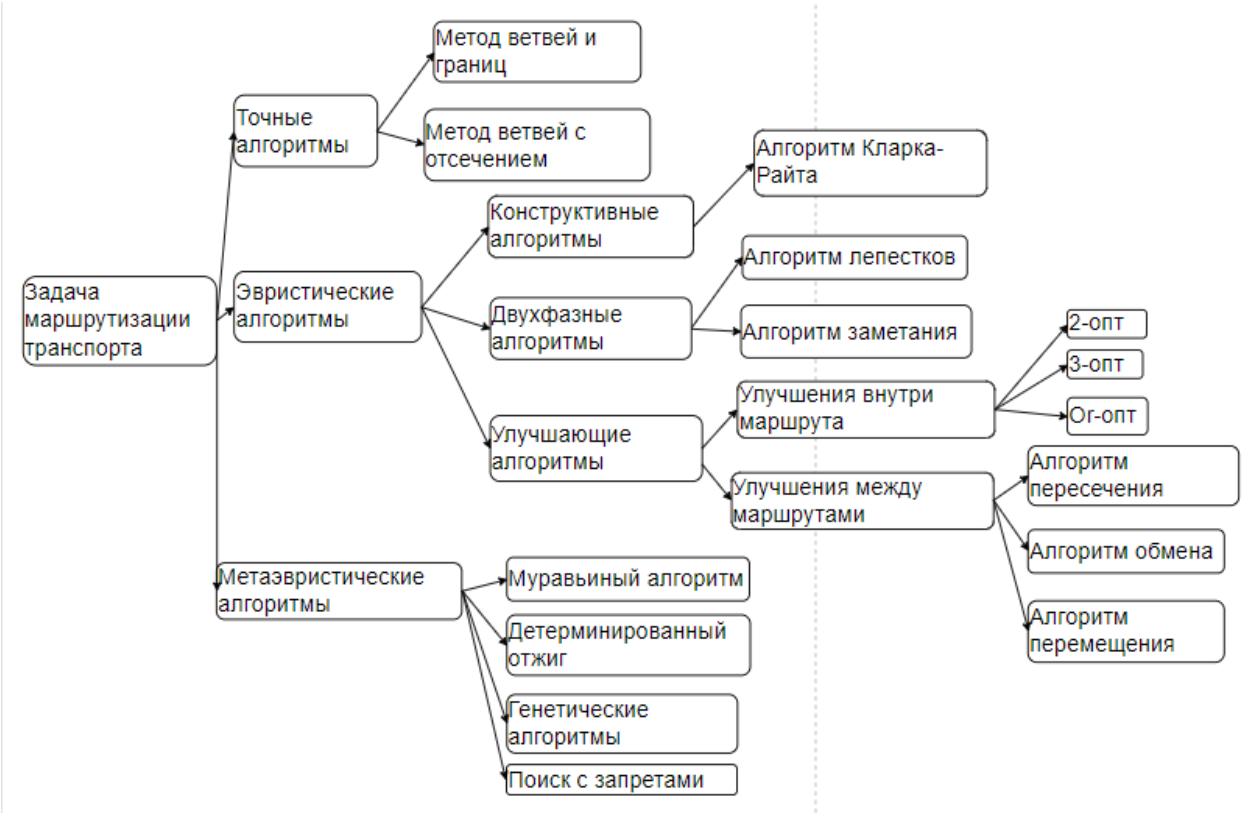

Рисунок 1 – Алгоритмы для решения ЗМТ

1.2.1 Точные алгоритмы

Данные алгоритмы предлагают вычислить все возможные решения, пока не будет достигнут лучший из них. Из-за того, что данная задача относится к классу NP-трудной, точные алгоритмы могут быть применены только для задач с малым количеством входных данных, поэтому они не подходят для решения реальных задач. В связи с этим, на практике больше всего применяют эвристические алгоритмы. Рассмотрим список из некоторых точных алгоритмов:

Рассмотрим метод ветвей и границ [20], который имеет следующие шаги для определения оптимального целочисленного решения:

1) найти оптимальное решение модели линейного программирования с релаксацией целых чисел;

2) на узле 1 пусть релаксированное решение является верхней границей, а округленное целочисленное решение – нижней границей;

3) выбрать переменную с наибольшей дробной частью для ветвления. Создать два ограничения для этой переменной, отражающие значения секционированного целого. Результатом будет новое $\leq$ ограничение и новое $\geq$ ограничение;

4) создать два новых узла: один для $\leq$ ограничения и один для $\geq$ ограничения;

5) решить свободную модель линейного программирования с новым ограничением, добавленным в каждом из этих узлов;

6) расслабленным решением является верхняя граница в каждом узле, а существующее максимальное целочисленное решение (на любом узле) является нижней границей;

7) если процесс создает допустимое целочисленное решение с наибольшим значением верхней границы любого конечного узла, достигается

оптимальное целочисленное решение. Если возможное целочисленное решение не возникает, переходите от узла с наибольшей верхней границей;

8) вернуться к шагу 3.

Для модели минимизации расслабленные решения округляются, а верхняя и нижняя границы меняются на противоположные.

Вторым методом рассмотрим метод ветвей с отсечением [21]. Метод основан на следующей идеи. Если нижняя граница значений функции на подобласти A дерева поиска больше, чем верхняя граница на какой-либо ранее просмотренной подобласти B, то A может быть исключена из дальнейшего рассмотрения (правило отсева). Как правило, минимальную из полученных верхних оценок записывают в глобальную переменную m. Любой узел дерева поиска, нижняя граница которого больше значения m, может быть исключён из дальнейшего рассмотрения.

Если нижняя граница для узла дерева совпадает с верхней границей, то это значение является минимумом функции и достигается на соответствующей подобласти.

1.2.2 Эвристические алгоритмы

Эвристические алгоритмы – это алгоритмы, которые будучи основанными на неком правиле (эвристике), не всегда следующей из строгих математических принципов, в подавляющем большинстве случаев дают решение, близкое к точному.

Эвристические алгоритмы делятся на:

1) двухфазные алгоритмы;
2) конструктивные алгоритмы;
3) улучшающие алгоритмы.

Конструктивные алгоритмы. Это алгоритмы, которые шаг за шагом выстраивают решение, учитывая общую стоимость, которая получается в ходе решения.

Алгоритм Кларка-Райта [9]. Является одним из концептуально простых эвристик. Алгоритм выполняется следующим образом:

1) сделать n маршрутов: $v_0 \rightarrow v_i \rightarrow v_0$, для каждого $i \geq 1$;

2) вычислить экономию для слияния мест доставки i и j, которая задается $s_{ij} = d_{i0} + d_{0j} - d_{ij}$, для всех $i, j \geq 1$ и $i \neq j$;

3) отсортировать в порядке убывания сбережения;

4) начиная с верхнего (оставшегося) списка сбережений, объедините два связанных маршрута с наибольшей (оставшейся) экономией при условии, что:

- два места доставки не находятся на том же маршруте;
- ни одно местоположение доставки не является внутренним по отношению к его маршруту, что означает, что обе записи по-прежнему непосредственно связанны с депо на соответствующих маршрутах;
- требование G и ограничения расстояния D не нарушаются объединенным маршрутом.

5) повторить шаг 3, пока не будет достигнута дополнительная экономия.

Двухфазные алгоритмы. Задача в двухфазном алгоритме делится на две части.

1) сбор вершин в группы;

2) построение маршрута для каждой группы.

Алгоритм лепестков [22]. Естественным расширением алгоритма развертки является создание нескольких маршрутов, называемых лепестками, и окончательный выбор путем решения заданной задачи разбиения формы.

Минимизируем

$$_{k \in S} d_k x_k, \quad (1)$$

при условии

$$_{k\in S}\, d_k\, x_k = 1, \tag{2}$$

где $i = 1, \ldots, n$, $x_k = 0$ or 1,

$k \in S$, S – множество маршрутов,

$x_k = 1$ тогда и только тогда, когда маршрут k принадлежит решению,

a_{ik} – двоичный параметр, равный 1, только если вершина i принадлежит маршруту k, а d_k – стоимость лепестка k.

Если маршруты соответствуют смежным секторам вершин, то эта проблема обладает свойством округления и может быть решена в полиномиальное время.

Алгоритм заметания [23]. Так как в традиционной задаче ЗМТУГ узлы окружают центральное депо, алгоритм заметания применяется таким образом, как показано на рисунке 2:

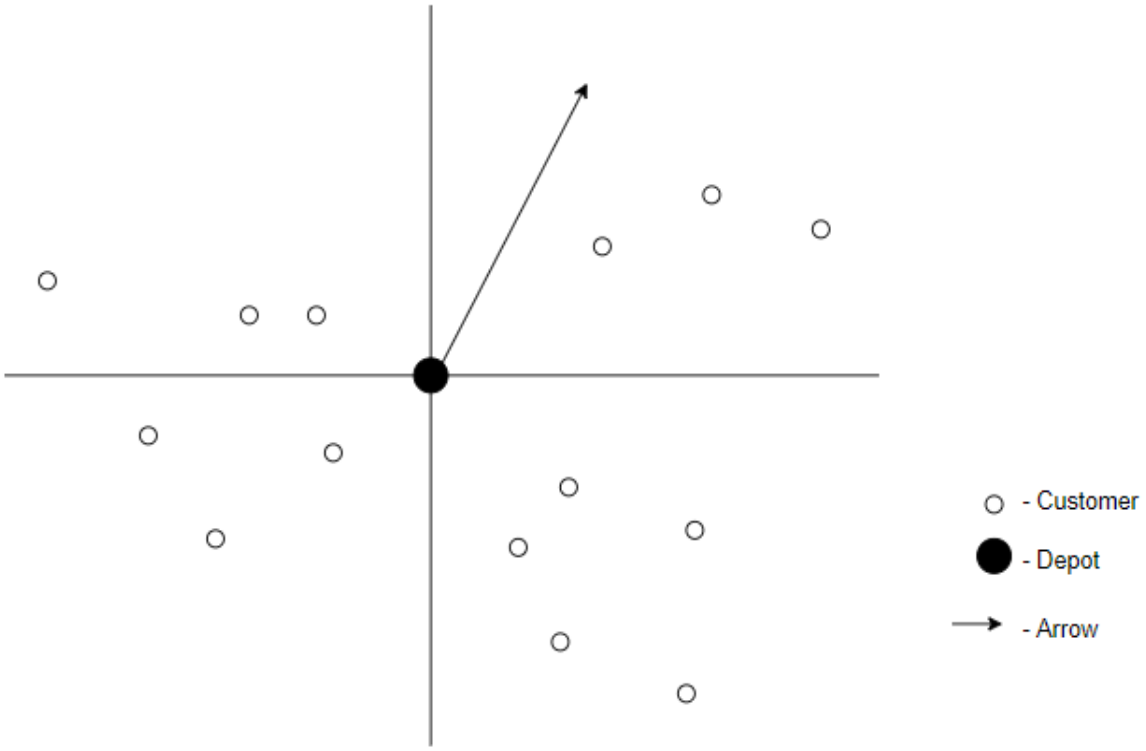

Рисунок 2 – Алгоритм заметания

1) радиальная линия с центральным депо в качестве центральной точки начинается с 0^0 и проходит по узлам, окружающим депо, по часовой стрелке или против часовой стрелки;

2) когда первый узел встречается, он присваивается первому транспортному средству и проверяется на ограничение пропускной способности. Если он подходит, узел назначается как посещенный;

3) затем линия снова прокручивается для получения второго узла;

4) когда второй узел назначен, сначала применяется ограничение пропускной способности, и если оно проходит, оно назначается транспортному средству, и узел устанавливается как посещенный;

5) «подметание» продолжается до тех пор, пока емкость транспортного средства не будет полной или какие-либо ограничения не будут превышены;

6) затем процедура начинается с номера транспортного средства 2 для остальных узлов;

7) повторение процедуры длится до тех пор, пока все узлы не будут назначены транспортным средствам.

Алгоритм Фишера-Джекумера [8] использует для формирования кластеров обобщённую задачу о назначениях. Предполагается, что количество транспортных средств k в данном методе задано явно. Работа алгоритма:

1) происходит выбор по одной вершине j_k из множества всех вершин V для начального заполнения каждого кластера k;

2) далее вычисляется стоимость d_{ik} назначения каждой вершины i каждому кластеру k как $d_{ijk} = \min \quad c_{0i} + c_{ij_k} + c_{i_k 0}, c_{0j_k} + c_{j_k i} + c_{i0} \; - (c_{0j_k} + c_{j_k 0})$ = min{c0i + cijk + cjk0, c0jk + cjki + ci0} − (c0jk + cjk0);

3) решается обобщённая задача о назначениях со стоимостями d_{ij}, с учётом весов клиентов q_i и грузоподъёмности транспортных средств Q;

4) в заключении решается ЗК для каждого кластера на основе результатов, найденного при помощи обобщённой задачи о назначениях.

Улучшающие алгоритмы. Сначала происходит формирование допустимого решения (любого), а затем улучшение его путем применения последовательных небольших изменений.

Основные из улучшающих алгоритмов:

1. Улучшения внутри маршрута (intra-route):
    1) 2-опт (2-opt);
    2) 3-опт (3-opt);
    3) Or-опт (Or-Opt).
2. Улучшения между маршрутами (inter-route):
    1) пересекающий (Cross);
    2) обмен (Exchange);
    3) перемещение (Relocation).

Улучшающие алгоритмы внутри маршрута, изображенных на рисунках 3, 4, 5 для ЗМТ описаны в терминах λ-опт операций, которые в свою очередь были предложены Лином [2]. В данных алгоритмах удаляются λ рёбер из маршрута, и происходит соединение λ оставшихся сегментов во всех комбинациях. И если первое достигнутое улучшающие соединение (наиболее удачное) найдено, то новые изменения заносятся в маршрут. Если более невозможно найти подходящие варианты замен, то работа завершается. Для проверки λ-оптимальности решения требуется времени $O(n\,\lambda)$.

Алгоритм 2-опт [2]. 2-опт алгоритм был предложен Croes [24] в 1958 году для решения задачи коммивояжера. Основная идея заключается в том, чтобы пройти маршрут, как показано на рисунке 3, который пересекает себя и переупорядочивает его так, чтобы он этого не делал.

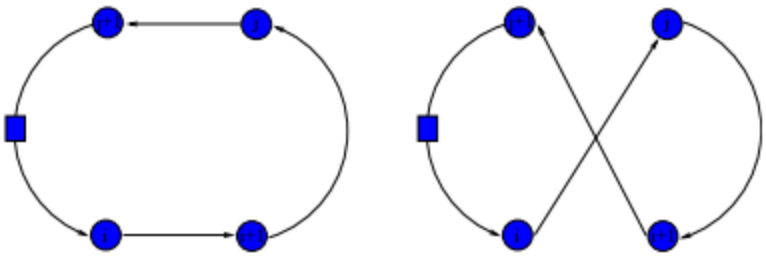

Рисунок 3 – 2-опт алгоритм

Алгоритм 3-опт [2].

3-опт алгоритм, представленный на рисунке 4, включает удаление 3-х соединений (или краев) в маршруте для создания 3-х суб-маршрутов. Затем анализируются 7 различных способов повторного подключения маршрута, чтобы найти оптимальный. Далее этот процесс повторяется для другого набора из 3 подключений, пока все возможные комбинации не будут проверены в маршруте. Он имеет временную сложность $O\ n^3$.

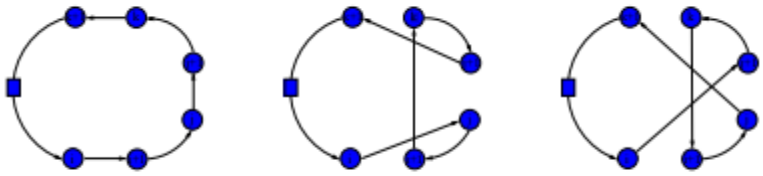

Рисунок 4 – 3-опт алгоритм

Алгоритм Or-опт [2].

Назван в честь Ilhan Or (1976). Or-opt алгорим изображен на рисунке 5. Шаги выполнения алгоритма:

1) перемещать цепочки из трех последовательных вершин во всех возможных положениях;

2) внедрить любой прибыльный шаг;

3) повторять до тех пор, пока это выгодно;

4) начать с строк из 2 вершин и, наконец, с единственными вершинами.

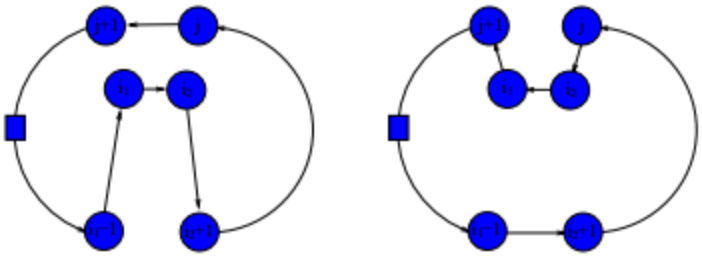

Рисунок 5 – Or-опт алгоритм

Улучшающие межмаршрутные алгоритмы кратко описываются в [3]. Оператор перемещения, представленный на рисунке 6, перемещает одного клиента в другой маршрут, в то время как оператор обмена, изображенный на рисунке 7, двух клиентов в двух разных турах, сохраняя свои позиции в турах. Оператор пересечения, как на рисунке 8, обменивает вторую часть двух разных маршрутов, сохраняя последовательность запросов в обменных частях маршрута.

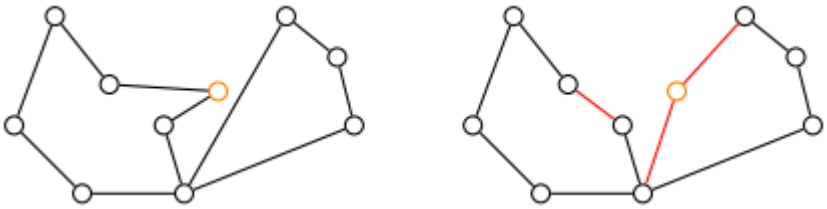

Рисунок 6 – Алгоритм перемещения

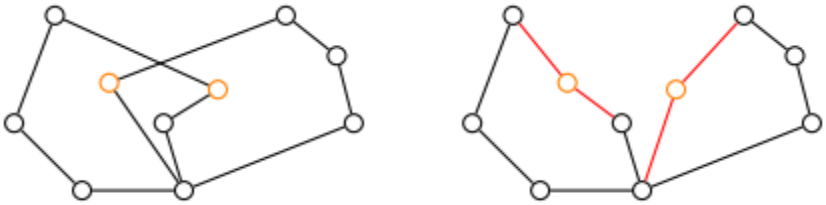

Рисунок 7 – Алгоритм обмена

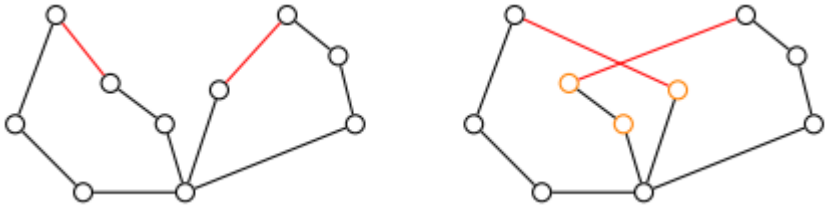

Рисунок 8 – Алгоритм пересечения

1.2.3 Метаэвристические алгоритмы

Метаэвристические алгоритмы, т.е. методы оптимизации, разработанные в соответствии с стратегиями, изложенными в метаэвристической структуре, - как следует из названия, всегда эвристичны по своей природе. Этот факт отличает их от точных методов, которые приходят с доказательством того, что оптимальное решение будет найдено в конечном (хотя и часто непомерно большом) количестве времени. Поэтому метаэвристика разработана специально для того, чтобы найти решение, которое «достаточно хорошо» в вычислительном времени, которое «достаточно мало». В результате они не подвергаются комбинаторному взрыву – явлению, в котором вычислительное время, необходимое для нахождения оптимального решения NP-трудных задач, возрастает как экспоненциальная функция размера задачи.

Научное сообщество продемонстрировало метаэвристику как жизнеспособную и часто превосходящую альтернативу более традиционным (точным) методам смешанной цельной оптимизации, таким как отраслевое и связанное и динамическое программирование. Специально для сложных проблем, метаэвристика часто может предложить лучший компромисс между качеством решения и вычислительным временем. Более того, метаэвристика более гибкая, чем точные, двумя важными способами. Во-первых, 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. Во-

вторых, метаэвристика не предъявляет никаких требований к формулировке задачи оптимизации (например, требование ограничения или целевых функций должно выражаться как линейные функции переменных решения). Однако эта гибкость сопряжена с затратами на значительную адаптацию к конкретной проблеме для достижения хорошей производительности.

Список из самых часто используемых метаэвристических алгоритмов:

1) муравьиный алгоритм (Ant Algorithm);
2) детерминированный отжиг (Deterministic Annealing);
3) генетические алгоритмы (Genetic Algorithms);
4) поиск с запретами (Tabu Search).

Так как цель бакалаврской работы состоит в применении эвристического алгоритма, нужно сделать грамотный выбор в пользу конкретно описанного в данной главе алгоритма.

Он должен обладать такими свойствами, как:

1) эффективностью;
2) простоте написания этого алгоритма;
3) приемлимой времени работы.

Проведя анализ по эвристическим алгоритмам, основываясь на различных исследованиях и статьях, использованных в данной работе, был сделан вывод, что для решения задачи маршрутизации транспорта чаще всего используют двухфазные и конструктивные алгоритмы. В качестве альтернативного варианта, в данной работе поиск эвристического алгоритма для решения задачи маршрутизации транспорта осуществлялся из категории улучшающих алгоритмов.

В таблице 1 проведен сравнительный анализ по критерию асимпотической вычислительной сложности улучшающих эвристических алгоритмов. Прежде чем сделать вывод по данной таблице, дадим определение асимпотической вычислительной сложности.

В теории сложности вычислений, асимптотическая вычислительная сложность является использование асимптотического анализа для оценки

вычислительной сложности алгоритмов и вычислительных задач, обычно связанных с использованием в большой нотации O.

Что касается вычислительных ресурсов, асимптотическая сложность времени и асимптотическое пространства сложности обычно оцениваются. Другое асимптотически оцениваемое поведение включает сложность схемы и различные меры параллельного вычисления, такие как количество (параллельных) процессоров.

Так как в оригинальной работе 1965 года Юриса Хартманиса и Ричарда Э. Стернса [25], а также книги Майкла Гарея и Дэвида С. Джонсона от 1979 года о NP-полноте [26], термин «вычислительная сложность» алгоритмов стал обычно называемой асимптотической вычислительной сложностью.

Кроме того, если не указано иное, термин «вычислительная сложность» обычно относится к верхней оценке асимптотической вычислительной сложности алгоритма или задачи, которая обычно записывается в терминах большой нотации O, например $O(n^3)$. Другие типы (асимптотических) оценок вычислительной сложности – это нижние границы (обозначение «Большая Омега», например, $\Omega(n)$) и асимптотически плотные оценки, когда асимптотическая верхняя и нижняя границы совпадают (записаны с использованием «большой теты», например, $\Theta\ (n\ log\ n)$).

Таблица 1 – Асимпотическая вычислительная сложность

Эвристические алгоритмы улучшения	Асимпотическая вычислительная сложность
Relocation	$O(n^2)$
Exchange	$O(n^2)$
Cross	$O(n^2)$
2-opt	$O(n^2)$
3-opt	$O(n^2)$
Or-opt	$O(n^3)$

В связи с тем что, в рамках данной работы количество вершин в входных данных будет небольшим, для выполнения цели бакалаврской работы подойдет любой алгоритм с асимпотической вычислительной сложностью $O\ n^2$.Таким образом, будет реализован алгоритм перемещения, в силу его простоты и эффективности.

1.3 Ограничения в задачах маршрутизации транспорта

На ЗМТ могут быть наложены определенные ограничения. Данные ограничения можно классифицировать так, как это предложено в статье [4]:

1. Задача маршрутизации транспорта с ограничением по грузоподъемности (ЗМТУГ). Это проблема маршрутизации транспортных средств, в которой парк идентичных транспортных средств с ограниченной грузоподъемностью, и расположенных на центральном складе, должен обслуживать набор клиентов с известными требованиями. Цель состоит в том, чтобы определить набор поездок транспортных средств с минимальной общей стоимостью (суммой расстояния или времени поездки), чтобы каждый автомобиль начинал и заканчивался на складе, каждый клиент посещался ровно один раз, и общий спрос, обрабатываемый любым транспортным средством, не превышаел его мощности.

2. Задача маршрутиазции транспорта с временными окнами. Цель состоит в том, чтобы свести к минимуму парк транспортных средств и сумму времени в пути с временем ожидания, необходимое для обслуживания всех клиентов в требуемые часы.

3. Задача маршрутизации транспорта с погрузкой и доставкой. В данной задаче необходимо свести к минимуму парк транспортных средств и сумму времени поездки с ограничением на то, что транспортное средство должно иметь достаточную грузоподъемность для транспортировки товаров, которые должны быть доставлены, и тех, которые были выбраны заказчиками для их возвращения в депо.

4. Задача маршрутиазции транспорта с обратной сетью. Задачи с возможностью возврата товара после завершения обслуживания всех клиентов. Эта задача является 3 модификацией предыдущей задачи. При этом подразумевается, что каждый транспорт сначала развозит все товары, а затем начинает принимать товар от клиентов.

5. Задача маршрутизации транспорта с разделением доставки. Для этой задачи допускается, чтобы один и тот же клиент мог обслуживаться различными транспортными средствами, при условии что это уменьшит общие затраты.

6. Периодическая ЗМТ. Задачи с возможностью доставки грузов в течение длительного периода. В традиционнной ЗМТ обычный период планирования – один день. В задачах с периодической маршрутизацией применяется увеличенный промежуток планирования на некоторое количество суток.

7. Стохастическая ЗМТ. Задачи, в которых с целью описания действия покупателей или автотранспортных средств могут быть применены случайные переменные. В данном случае один либо ряд переменных (покупатели, запросы или время) имеют все шансы принимать случайное значение.

8. ЗМТ с несколькими депо. Задачи с возможностью применения автотранспортных средств с нескольких депо. Для того, чтобы совладать с растущим числом запросов на доставку товаров, компании, производящие доставку товаров, не только приумножают парк, но и формируют несколько складов (депо). Как правило, каждый склад обладает собственным парком и самостоятельно решает задачу маршрутизации автомобильного транспорта. Однако в совокупном случае допускается рассматривать версию, когда для погрузки каждый автомобиль компании может использовать любой склад, обладающим необходимым товаром.

В данной работе исследуется эвристический алгоритм с учетом грузоподъемности, то есть будет реализован эвритический алгоритм перемещения с ограничением по грузоподъемности.

1.4 Математическая модель задачи маршрутизации транспорта

Следуя из статьи [5], постановка математической модели ЗМТ может звучать следующим образом.

Задача маршрутизации транспорта определена на полном неориентированном графе $G = (V, E)$. Множество $V = \{0, \dots, n\}$ является множеством вершин. Каждая вершина $i \in V / \{0\}$ представляет собой клиент, имеющий неотрицательный спрос q_i, а вершина 0 соответствует депо, как это представлено на рисунке 9. Каждому ребру $e \in E = \{(i, j): i, j \in V, i < j\}$ соответствует стоимость поездки c_e или c_{ij}. На складе имеется фиксированный парк из двух одинаковых транспортных средств, каждый из которых имеет емкость Q. ЗМТ требует определения набора маршрутов m, стоимость которых в целом сводится к минимуму и такая, что:

1) каждый клиент посещается ровно один раз на один маршрут;
2) каждый маршрут запускается и заканчивается на складе;
3) общая потребность клиентов, обслуживаемых маршрутом, не превышает пропускную способность Q;
4) длина каждого маршрута не превышает установленный предел L.

Обычно принято предполагать постоянную скорость, чтобы расстояния, время проезда и расходы на проезд считались одинаковыми. Решение можно рассматривать как набор из m циклов, разделяющих общую вершину на складе.

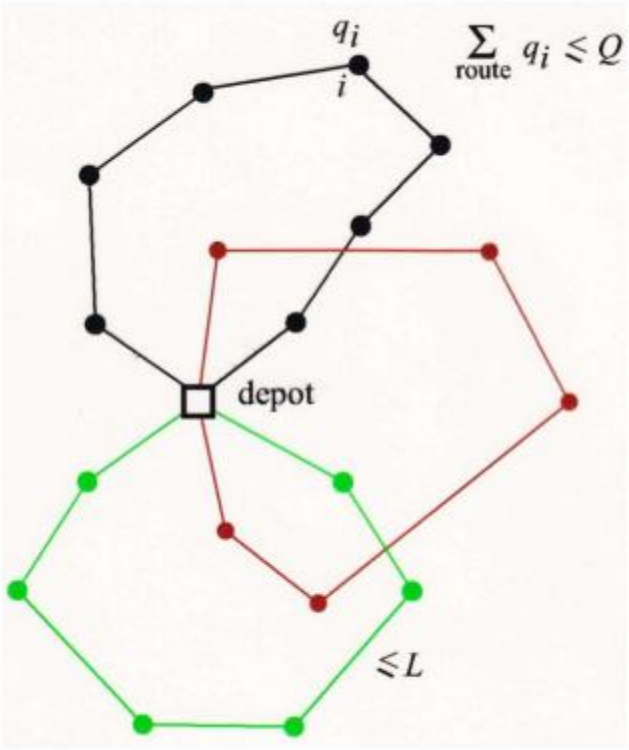

Рисунок 9 – Граф задачи маршрутизации транспорта

Далее следует целочисленная формулировка линейного программирования *ЗМТУГ*, где для каждого ребра $e \in E$ целочисленная переменная X_e указывает количество пересечений ребра *e* в решении.

Пусть *r (S)* обозначает минимальное количество транспортных средств, необходимых для обслуживания клиентов подмножества *S* клиентов. Значение *r (S)* можно определить, разрешив связанную проблему упаковки бинов (BPP) с набором элементов S и бинов емкости *Q*. Наконец, для $S \subset V$ пусть $\delta(S) = \{(i, j): i \in S, j \notin S \text{ или } i \notin S, j \in S\}$. Если $S = \{i\}$, тогда мы просто будем писать $\delta(i)$. Формула ЗМТУГ, предложенная Laporte et al. (1985):

$$\text{minimize} \quad \sum_{e \in E} C_e X_e \tag{4}$$

$$\sum_{e \in \delta(i)} X_e = 2, \; i \in V \backslash \{0\}, \tag{5}$$

$$\sum_{e \in \delta(0)} X_e = 2m, \tag{6}$$

$$\sum_{e \in \delta(S)} X_e \geq 2r(S), \; S \subseteq V \backslash \{0\}, S \neq 0, \tag{7}$$

$$X_e \in \{0,1\}, \; e \notin \delta\{0\}, \tag{8}$$

$$X_e \in \{0,1,2\}, \; e \in \delta\{0\}. \tag{9}$$

Ограничения степени (5) указывают, что каждый клиент посещается ровно один раз, тогда как ограничение степени депо (6) означает, что m маршрутов создаются. Ограничения пропускной способности (7) накладывает как связность решения, так и требования к емкости транспортного средства, заставляя достаточное количество ребер войти в каждое подмножество вершин. Наконец, ограничения (8) и (9) говорят, что каждое ребро между двумя клиентами проходит не чаще одного раза, и каждое ребро, инцидентное депо, проходит не более двух раз. В последнем случае автомобиль выполняет маршрут, посещая одного клиента.

ГЛАВА 2 РЕАЛИЗАЦИЯ ЭВРИСТИЧЕСКОГО АЛГОРИТМА ДЛЯ РЕШЕНИЯ ЗМТ

Применение эвристического алгоритма перемещения в условиях ограничений по грузоподъемности можно разделить на несколько этапов:

1) обоснование выбора технических средств;
2) подготовка исходных данных;
3) реализация алгоритма и архитектуры приложения;
4) проведение вычислительного эксперимента.

Готовый эвристический алгоритм перемещения, схематично изображенный на рисунке 10 [27], должен будет решать задачу маршрутизации транспорта с учетом ограничения по грузоподъемности.

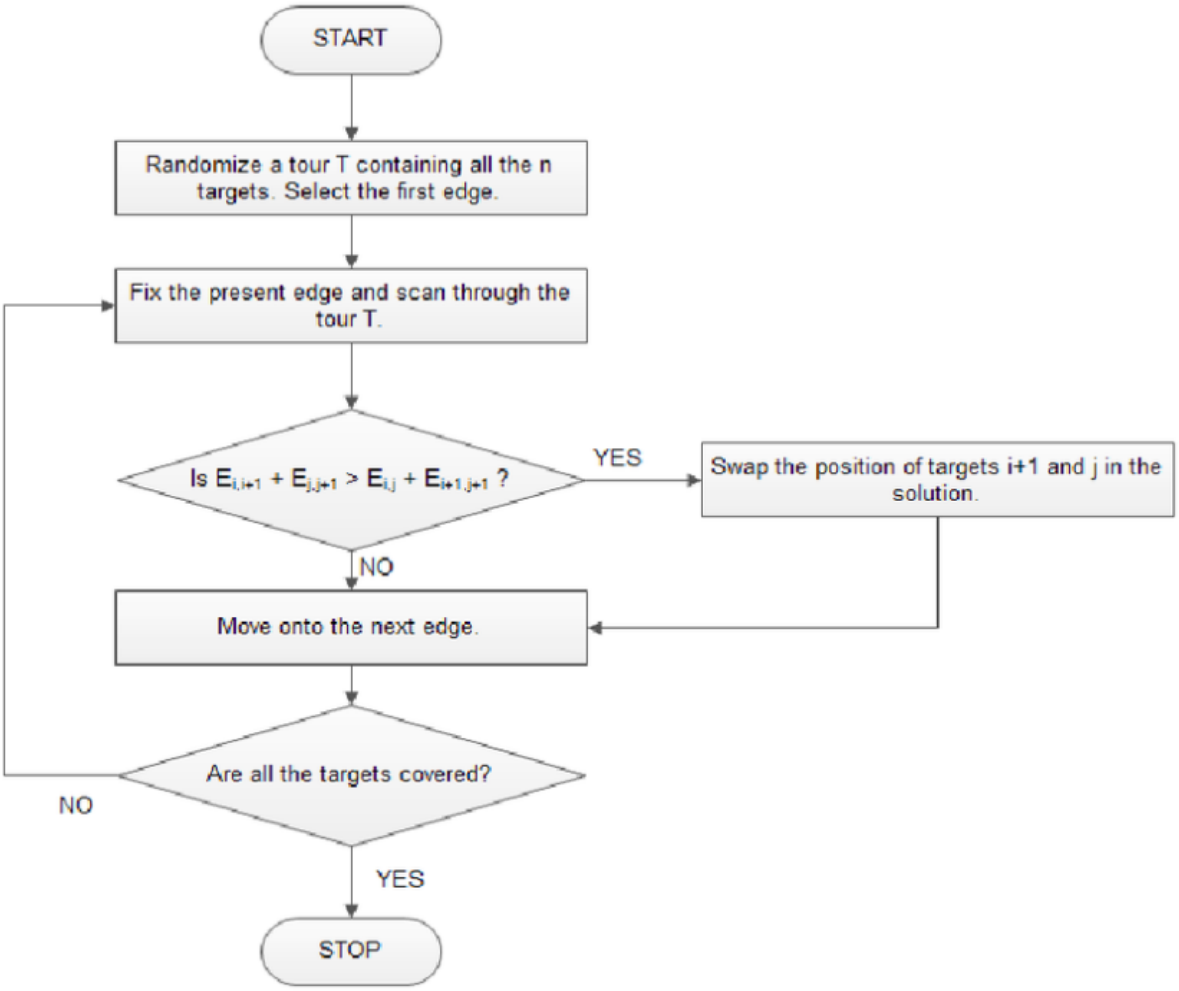

Рисунок 10 – Блок-схема алгоритма перемещения

2.1 Обоснование выбора технических средств

Язык программирования является формальным языком, который указывает на набор инструкций, которые могут быть использованы для производства различных видов продукции. Языки программирования обычно состоят из инструкций для компьютера. Языки программирования могут использоваться для создания программ, реализующих определенные алгоритмы.

Для разработки алгоритмов для решения задач маршрутизации транспорта применяются различные языки программирования. Рассмотрим основные из них:

Java – высокоуровненный, объектно-ориентированный язык программирования.

Преимущества Java:

1. возможность использования объектно-ориентированных парадигм;
2. богатая библиотека;
3. относительная простота языка;
4. большая безопасность;
5. мультиплатформность;
6. имеет множество удобных инструментов для автоматизации сборок.

Недостатки Java:

1. ресурсоемкость.

C++ – высокоуровневый язык программирования.

Преимущества C++:

1. поддерживает различные стили и технологии;
2. возможность применения шаблонов для написания обобщенных контейнеров;
3. возможность имитации расширения языка для поддержки парадигм, которые не поддерживаются компиляторами напрямую;
4. имеется возможность работы на низком уровне с памятью, адресами.

Недостатки C++:

1. сложность и избыточность.

Matlab – высокоуровневый язык программирования предназначенный для технических расчетов.

Преимущества Matlab:

1. прост для изучений;
2. упрощенная разработка новых алгоритмов;
3. ежегодные обновления модулей;
4. наличие готовых модулей для работы с нейронными сетями различных типов.

Недостатки Matlab:

1. низкая вычислительная производительность;
2. перегруженность операторами и функциями.

Описав преимущества и недостатки представленных языков, составим таблицу 2 и занесем необходимые критерии для выбора языка программирования.

Таблица 2 – Критерии для выборы языка программирования

	Java	C++	Matlab
Производительность	+	+	-
Простота понимания	+	-	+
Автоматизация сборки	+	-	-

Для разработки было принято решение воспользоваться языком Java, так как его преимущества хорошо подходят для цели данной работы.

2.2 Исходные данные

Под исходными данными в задачах маршрутизации транспорта, или как их еще называют, тестами производительности, понимается набор входных

параматеров, таких как количество клиентов, их координаты, требования к проблеме и так далее. Служат они не только, чтобы убедиться с работоспособности программы, но и для того, чтобы сравнивать полученые решения (маршруты, время работы программы) с другими.

Приведем список из исходных данных для задачи маршрутизации транспорта с ограничением по грузоподъемности.

1. Set A (Augerat, 1995)

Состоит из 27 экземпляров, количество транспортных средств варьируется от 5 до 10, максимальная вместимость транспорта 100 усл. ед. показан на рисунке 11.

Benchmark	Instance	*n*	*K*	*Q*	UB
	A-n32-k5	31	5	100	784
	A-n33-k5	32	5	100	661
	A-n33-k6	32	6	100	742
	A-n34-k5	33	5	100	778
	A-n36-k5	35	5	100	799
	A-n37-k5	36	5	100	669
	A-n37-k6	36	6	100	949
	A-n38-k5	37	5	100	730
	A-n39-k5	38	5	100	822
	A-n39-k6	38	6	100	831
	A-n44-k6	43	6	100	937
	A-n45-k6	44	6	100	944
	A-n45-k7	44	7	100	1146
	A-n46-k7	45	7	100	914
	A-n48-k7	47	7	100	1073
	A-n53-k7	52	7	100	1010
	A-n54-k7	53	7	100	1167
	A-n55-k9	54	9	100	1073
	A-n60-k9	59	9	100	1354
	A-n61-k9	60	9	100	1034
	A-n62-k8	61	8	100	1288
	A-n63-k9	62	9	100	1616
	A-n63-k10	62	10	100	1314
	A-n64-k9	63	9	100	1401
	A-n65-k9	64	9	100	1174
	A-n69-k9	68	9	100	1159
	A-n80-k10	79	10	100	1763

Рисунок 11 – Set A

2. Set B (Augerat, 1995)

Состоит из 23 экземпляров, количество транспортных средств варьируется от 5 до 10, максимальная вместимость транспорта 100 усл. ед., количество вершин от 30 до 77, показан на рисунке 12.

Benchmark	Instance	n	K	Q	UB
	B-n31-k5	30	5	100	672
	B-n34-k5	33	5	100	788
	B-n35-k5	34	5	100	955
	B-n38-k6	37	6	100	805
	B-n39-k5	38	5	100	549
	B-n41-k6	40	6	100	829
	B-n43-k6	42	6	100	742
	B-n44-k7	43	7	100	909
	B-n45-k5	44	5	100	751
	B-n45-k6	44	6	100	678
	B-n50-k7	49	7	100	741
	B-n50-k8	49	8	100	1312
	B-n51-k7	50	7	100	1032
	B-n52-k7	51	7	100	747
	B-n56-k7	55	7	100	707
	B-n57-k7	56	7	100	1153
	B-n57-k9	56	9	100	1598
	B-n63-k10	62	10	100	1496
	B-n64-k9	63	9	100	861
	B-n66-k9	65	9	100	1316
	B-n67-k10	66	10	100	1032
	B-n68-k9	67	9	100	1272
	B-n78-k10	77	10	100	1221

Риснок 12 – Set B

3. Set E (Christofides and Eilon, 1969)

Состоит из 13 экземпляров, количество транспортных средств варьируется от 4 до 14, максимальная вместимость транспорта 112-6000 усл. ед., количество вершин от 12 до 100, показан на рисунке 13.

Benchmark	Instance	n	K	Q	UB
	E-n13-k4	12	4	6000	247
	E-n22-k4	21	4	6000	375
	E-n23-k3	22	3	4500	569
	E-n30-k3	29	3	4500	534
	E-n31-k7	30	7	140	379
	E-n33-k4	32	4	8000	835
	E-n51-k5	50	5	160	521
	E-n76-k7	75	7	220	682
	E-n76-k8	75	8	180	735
	E-n76-k10	75	10	140	830
	E-n76-k14	75	14	100	1021
	E-n101-k8	100	8	200	815
	E-n101-k14	100	14	112	1067

Рисунок 13 – Set E

4. Set F (Fisher, 1994)

Состоит из 3 экземпляров, количество транспортных средств варьируется от 4 до 7, максимальная вместимость транспорта 2010-300000 усл. ед., количество вершин от 30 до 77, показан на рисунке 14.

Benchmark	Instance	*n*	*K*	*Q*	UB
	F-n45-k4	44	4	2010	724
	F-n72-k4	71	4	30000	237
	F-n135-k7	134	7	2210	1162

Рисунок 14 – Set F

5. Set M (Christofides, Mingozzi and Toth, 1979)

Состоит из 5 экземпляров, количество транспортных средств варьируется от 7 до 17, максимальная вместимость транспорта 200 усл. ед., количество вершин от 100 до 199, показан на рисунке 15.

Benchmark	Instance	*n*	*K*	*Q*	UB
	M-n101-k10	100	10	200	820
	M-n121-k7	120	7	200	1034
	M-n151-k12	150	12	200	1015
	M-n200-k16	199	16	200	1274
	M-n200-k17	199	17	200	1275

Рисунок 15 – Set M

6. Set P (Augerat, 1995)

Состоит из 23 экземпляров, количество транспортных средств варьируется от 2 до 15, максимальная вместимость транспорта 35-3000 усл. ед., количество вершин от 15 до 100 показан на рисунке 16.

Benchmark	Instance	*n*	*K*	*Q*	UB
	P-n16-k8	15	8	35	450
	P-n19-k2	18	2	160	212
	P-n20-k2	19	2	160	216
	P-n21-k2	20	2	160	211
	P-n22-k2	21	2	160	216
	P-n22-k8	21	8	3000	603
	P-n23-k8	22	8	40	529
	P-n40-k5	39	5	140	458
	P-n45-k5	44	5	150	510
	P-n50-k7	49	7	150	554
	P-n50-k8	49	8	120	631
	P-n50-k10	49	10	100	696
	P-n51-k10	50	10	80	741
	P-n55-k7	54	7	170	568
	P-n55-k10	54	10	115	694
	P-n55-k15	54	15	70	989
	P-n60-k10	59	10	120	744
	P-n60-k15	59	15	80	968
	P-n65-k10	64	10	130	792
	P-n70-k10	69	10	135	827
	P-n76-k4	75	4	350	593
	P-n76-k5	75	5	280	627
	P-n101-k4	100	4	400	681

Рисунок 16 – Set P

7. Christofides, Mingozzi and Toth (1979)

Состоит из 14 экземпляров, количество транспортных средств варьируется от 5 до 18, максимальная вместимость транспорта 140-200 усл. ед., количество вершин от 50 до 100 показан на рисунке 17.

Benchmark	Instance	*n*	*K*	*Q*	UB
	CMT1	50	5	160	524.61
	CMT2	75	10	140	835.26
	CMT3	100	8	200	826.14
	CMT4	150	12	200	1028.42
	CMT5	199	17	200	1291.29
	CMT6	50	6	160	555.43
	CMT7	75	11	140	909.68
	CMT8	100	9	200	865.95
	CMT9	150	14	200	1162.55
	CMT10	199	18	200	1395.85
	CMT11	120	7	200	1042.12
	CMT12	100	10	200	819.56
	CMT13	120	11	200	1541.14
	CMT14	100	11	200	866.37

Рисунок 17 – Christofides, Mingozzi and Toth (1979)

8. Li et al. (2005)

Состоит из 12 экземпляров, количество транспортных средств варьируется от 10 до 20, максимальная вместимость транспорта 900-2500 усл. ед., количество вершин от 560 до 1200, показан на рисунке 18.

Benchmark	Instance	*n*	*K*	*Q*	UB
	Li_21	560	10	1200	16212.83
	Li_22	600	15	900	14499.04
	Li_23	640	10	1400	18801.13
	Li_24	720	10	1500	21389.43
	Li_25	760	19	900	16668.51
	Li_26	800	10	1700	23977.73
	Li_27	840	20	900	17372.64
	Li_28	880	10	1800	26566.03
	Li_29	960	10	2000	29154.34
	Li_30	1040	10	2100	31742.64
	Li_31	1120	10	2300	34330.94
	Li_32	1200	11	2500	37159.41

Рисунок 18 – Li et al

Каждый из представленных исходных данных различаются набором входных параметров, например вместимостью транспорта, количеством маршрутов и самих транспортных средств.

В качестве экземпляра теста производительности был выбран Christofides, Mingozzi and Toth (1979).

2.3 Реализация алгоритма

Разработка программы для каждодневного использования в логистических задачах является завершающей стадией работы. Она дает возможность внедрить академические достижения на практике с получением финансового эффекта.

Основные этапы решения задача маршрутизации транспорта (ЗМТ):

1) загрузка исходных данных из текстового файла;

2) обрабатывание данных и создание несколько замкнутых маршрутов, проходящих через установленное множество целевых вершин. Через каждую вершину должен пролегать только лишь единственный путь;

3) вычисление матрицы стоимости перемещений;

4) пуск алгоритмов решения ЗМТ с целью расчета маршрутов;

5) вывод полученных маршрутов.

Требования к программной реализации ЗМТ:

1) все маршруты обязаны следовать через депо;

2) единая цена объезда маршрутов обязана быть минимальной.

Кроме основного вида, на задачу накладывается дополнительное ограничение грузоподъёмности.

Были выделены и разработаны следующие Java-классы [Приложение А]. Проведем краткий обзор по самым выжным состовляющих программы:

1) класс Node – предназначен для хранения информации о клиенте: его координаты, максимальная грузоподъемность и порядковый номер;

2) класс Route – вспомогательный класс, хранящий в себе все точки-клиенты и дополнительные сведения для работы алгоритма;

3) класс Loader – предназнчен для загрузки исходных данных (тестов производительности);

4) класс Utils – содержит в себе набор методов для отображения полученных решений алгоритма;

5) класс Solution – класс, предназнченный для сбора полученных данных и распределяющий их по транспортным средствам;

6) класс Problem – служит для формирования проблемы, которую требуется решить;

7) класс Main – точка старта программы.

Следующим шагом, после определения и реализации классов, стало реализация самого алгоритма перемещения. Для начала был сформулирован псевдокод, представленный на рисунке 19, для того чтобы понять будущую структуру алгоритма.

Input: Множество городов $C = \{c_1, \ldots, c_n\}$, $n \times n$ матрица D расстояний.
Output: Перестановка $T = (c_{\pi_1}, \ldots, c_{\pi_n})$ городов.
begin
  $\tau :=$ исходный тур τ_0;
  Пусть $Q = \{(i,j) \mid i,j \in \{1,\ldots,n\}$ и $i \neq j\}$;
  $N := Q$;
  repeat
    Пусть $\tau = (c_{i_1}, \ldots, c_{i_n})$;
    Выбрать пару индексов $(p,q) \in N$;
    $N := N - \{(p,q)\}$;
    $\tau' := (c_{i_1}, \ldots, c_{i_{p-1}}, c_{i_q}, c_{i_{q-1}}, \ldots, c_{i_{p+1}}, c_{i_p}, c_{i_{q+1}}, \ldots, c_{i_n})$;
    if $l(\tau') < l(\tau)$ **then**
    begin
      $\tau := \tau'$;
      $N := Q$
    end
  until $N = \emptyset$;
  return τ
end.

Рисунок 19 – Псевдокод алгоритма перемещения

На основе псевдокода, составлялся программный код на языке программирования Java, в классе RelocateOptimizer. Фрагмент кода представлен на рисунке 20.

```java
private boolean relocate(List<Vehicle> aList) {
    double gain;
    double bestGain = 0;

    Vehicle bestSourceVehicle = null;
    Vehicle bestDestVehicle = null;
    Pair<Integer, Integer> bestRelocate = new Pair<>();

    for (Vehicle b : aList) {
        for (Vehicle a : aList) {

            System.out.println("expanding {" + a + "}, shrinking {" + b + "}");

            float remainingCapacity = a.getCapacity() - a.getRoute().demand();

            for (int i = 1; i < b.getRoute().size() - 1; i++) {    // iterate on customers (no depots!)
                if (b.getRoute().demand() < remainingCapacity) {    // customer i can fit into route a
                    System.out.println("node {" + i + "} fit into route {" + a + " }");
                    /*
                     * for each possible position in route a
                     * find a position that decrement the amount cost of the two routes
                     */
                    for (int j = 0; j < a.getRoute().size() - 1; j++) {
                        gain = calculateGain(a.getRoute(), b.getRoute(), j, i);
                        System.out.println("node {" + i + "}, between {" + j + "} {" + (j + 1) + "}, gain = {" + gain + "}");

                        if (gain > bestGain) {
                            bestSourceVehicle = b;
                            bestDestVehicle = a;
                            bestRelocate.set(i, j);        // save indexes and move at the end
                            bestGain = gain;            // update best gain
                        }
                    }
                }
            }
        }
    }
```

Рисунок 20 – Фрагмент алгоритма в программной коде Java

Для обеспечения демонстрации работы приложения, был реализован консольный вывод информации. На первоначальном этапе выводятся исходные данные в удобном для просмотра форме, как на рисунке 21.

```
line { 0 } : { [120, 200, 720, 50] }
line { 1 } : { [10, 45] }
line { 2 } : { [25, 1, 25] }
line { 3 } : { [25, 3, 7] }
line { 4 } : { [31, 5, 13] }
line { 5 } : { [32, 5, 6] }
line { 6 } : { [31, 7, 14] }
line { 7 } : { [32, 9, 5] }
line { 8 } : { [34, 9, 11] }
line { 9 } : { [46, 9, 19] }
line { 10 } : { [35, 7, 5] }
line { 11 } : { [34, 6, 15] }
line { 12 } : { [35, 5, 15] }
line { 13 } : { [47, 6, 17] }
line { 14 } : { [40, 5, 13] }
line { 15 } : { [39, 3, 12] }
```

Рисунок 21 – Консольный вывод начальных значений

В результате работы алгоритма выводятся оптимизированные маршруты. Данные маршруты можно наблюдать на рисунке 22.

```
number of customers: 50
vehicle capacity: 160.0
solution:
[0, 12, 5, 49, 9, 38, 46, 0] (demand=99.0) {cost=55.04878606977536}
[0, 1, 28, 31, 26, 8, 48, 27, 0] (demand=94.0) {cost=79.45534162757446}
[0, 11, 2, 20, 35, 36, 3, 22, 32, 0] (demand=136.0) {cost=93.86146681253084}
[0, 4, 17, 15, 45, 33, 39, 10, 30, 34, 50, 21, 29, 16, 0] (demand=158.0) {cost=142.4532549575219}
[0, 37, 44, 42, 19, 40, 41, 13, 25, 14, 0] (demand=153.0) {cost=114.04452792191931}
[0, 23, 7, 43, 24, 6, 0] (demand=71.0) {cost=77.84653205966423}
[0, 47, 18, 0] (demand=66.0) {cost=32.26106194058855}
cost: 594.9709713895746
```

Рисунок 22 – Решение проблемы ЗМТУГ

2.4 Проведение вычислительного эксперимента

Сравним результаты теста производительности Christofides, Mingozzi and Toth(CMT) [16] по каждому экземпляру, с результатами разработанного эвритического алгоритма. Полученные данные запишем в таблицу 3. В качестве результативного параметра принимается общая стоимость проезда автомобиля из всех оптимизированных маршрутов. В первой колонке представлены экземпляры теста производительности, в количестве четырнадцати штук. Во второй колонке находятся лучшие результаты теста, в третьей – результаты полученные при помощи реализованного алгоритма перемещения.

Таблица 3 – Результаты работы алгоритма

	Лучшие значения стоимости маршрутов, полученные на основе исходных данных Christofides, Mingozzi and Toth(CMT)	Значения стоимости маршрутов, полученные разработанным алгоритмом
CMT1	524,61	594,97
CMT2	835,26	903,90
CMT3	826,14	961,39
CMT4	1028,42	1228,09
CMT5	1291,29	1625,01
CMT6	555,43	594,97

CMT7	909,68	903,90
CMT8	865,95	961,39
CMT9	1162,55	1228
CMT10	1395,85	1625
CMT11	1042,12	1438,90
CMT12	819,56	967,81
CMT13	1541,14	1438
CMT14	866,37	967

Проведя анализ по таблице 3, можно сказать, что полученные значения, разработанные эвритическим алгоритмом перемещения, близки к лучшим результатам данного теста производительности, которые были получены за последние 30 лет.

ЗАКЛЮЧЕНИЕ

В данной работе был рассмотрен ряд видов задачи маршрутизации транспорта, подробно описаны распространенные алгоритмы решения ЗМТ, такие как точные, эвристические и метаэвристические алгоритмы. Среди них был эвристический алгоритм перемещения, который был реализован на языке программирования Java. В качестве теста производильности использовался тест Christofides, Mingozzi and Toth. В конце работы проведен вычислительный эксперимент, в результате которого был сделан вывод, что разработанный алгоритм стремится к лучшим результатам данного теста.

В рамках бакалаврской работы все цели и задачи выполнены.

Итогом проделанной работы является:

1) изучена литература, посвященная задаче маршрутизации транспорта;

2) рассмотрены известные методы решения задачи маршрутизации транспорта;

3) проведен анализ по эвристическим алгоритмам;

4) реализован эвристический алгоритм перемещения для решения задачи маршрутизации транспорта;

5) разработана программа, способная решать задачу маршрутизации транспорта с использованием эвристического алгоритма в условиях ограничений по грузоподъемности.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Dantzig G. B., Ramser J. H., The Truck Dispatching Problem, [Электронный ресурс] / G. B. Dantzig, J. Ramser // Management Science, Vol. 6, No. 1, 1959. – 91.

2. Lin S., Computer solutions of the traveling salesman problem [Электронный ресурс] / S. Lin // Bell System Technical Journal. — 1965. — № 44. — P. 2245–2269.

3. Ferrucci, Francesco, Pro-active Dynamic Vehicle Routing [Книга] / Ferrucci, Francesco // Operations Research. — 2013г. — P. 280.

4. Cordeau Jean-François, Laporte Gilbert, W.P. Savelsbergh Martin, Vigo Daniele, Vehicle Routing [Электронный ресурс] / Jean-François Cordeau, Gilbert Laporte, W.P. Savelsbergh Martin, Daniele Vigo // USA.: Handbook in OR & MS, Vol. 14, 2007. – 62c.

5. Hoskins M., Lobit H., Pillac V., Vidal T., Vigo D., Mendoza J.E., Guéret C., VRP-REP: the vehicle routing community repository. Third meeting of the EURO Working Group on Vehicle Routing and Logistics Optimization [Электронный ресурс] / M. Hoskins, H. Lobit, V. Pillac, T. Vidal, D. Vigo, J.E. Mendoza, C. Guéret // Oslo, Norway – 2014.— 89p.

6. Toth Paolo, Vigo Daniele, The Vehicle Routing Problem [Электронный ресурс] / Paolo Toth, Daniele Vigo // Society for Industrial and Applied Mathematics, 2002. – 367c.

7. Caric Tonci, Gold Hrvoje, Vehicle Routing Problem / Tonci Caric, Hrvoje Gold // Austria.: In-Teh is Croatian branch of I-Tech Education and Publishing KG, 2008. – 152c.

8. Fisher M., Jaikumar R., A generalized assignment heuristic for vehicle routine / M. Fisher, R. Jaikumar // Net-works, 1981. – P. 109-124.

9. Clark G., Write J. W., Scheduling of vehicles from central depot to a number delivery points / G. Clark, J. W. Write// Oper. Res. Quarts, 1964. – 568-581c.

10. Bach Lukas, Hasle Geir, Wøhlk Sanne, A lower bound for the Node, Edge, and Arc Routing Problem [Text] / Lukas Bach, Geir Hasle, Sanne Wøhlk // Department of Applied Mathematics, Norway – 2013. – PP. 943-952.

11. Gromicho J., Haneyah S., Kok A.L., Solving a Real-Life VRP with Inter-Route and Intra-Route Challenges [Text] / J. Gromicho, S. Haneyah, A.L. Kok // VU University Amsterdam, Netherlands – 2015. – PP. 30.

12. Gromicho J., van Hoorn J.J., Kok A.L., Schutten J.M.J., Restricted dynamic programming: A flexible framework for solving realistic VRPs [Text] / J. Gromicho, J.J.van Hoorn, A.L. Kok, J.M.J. Schutten // Vrije Universiteit, Amsterdam, The Netherlands – 2011. – P. 902-909.

13. Tsouros Michael, Pitsiava Magda, Grammenidou Karin, Routing-Loading Balance Heuristic Algorithms for a Capacitated Vehicle Routing Problem [Text] / Michael Tsouros, Magda Pitsiava, Karin Grammenidou // Faculty ofEngineering, Aristotle University of Thessaloniki, Greece – 2006. – P. 943-952.

14. Mavroidis Ioannis, Papaefstathiou Ioannis, Pnevmatikatos Dionisios, Hardware Implementation of 2-Opt Local Search Algorithm for the Traveling Salesman Problem [Text] / Ioannis Mavroidis, Ioannis Papaefstathiou, Dionisios Pnevmatikatos // Technical University of Crete (TUC), Brazil – 2007. – PP. 43-52.

15. Kinderwater G.A.P, Vehicle routing: Handling edge exchanges. In E.H.L. Aarts, J.K. Lenstra, editors / G.A.P. Kinderwater, M.W.P. Savelsbergh [Электронный ресурс] // Local Search in Combinatorial Optimization. — Wiley, Chichester, 1997. — P. 337–360.

16. Christofides N., The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, C. Sandi, editors [Text] / N. Christofides, A. Mingozzi, P. Toth // Combinatorial Optimization. — Wiley, Chichester, 1979. — P. 315–338.

17. Пожидаев М. С., Сбалансированная эвристика для решения задачи маршрутизации транспорта с учетом грузоподъемности [Text] / Ю. Л. Костюк, М. С. Пожидаев // Вестник ТГУ. УВТиИ. — 2010. — № 3.

18. Haimovich M., Bounds and heuristics for capacitated routing problems [Электронный ресурс] / M. Haimovich, A.H.G. Rinnooy Kan // Mathematics of Operations Research. — 1985. — № 10. — P. 527–542.

19. Salhi S., Improvements to vehicle routing heuristics [Электронный ресурс] / S. Salhi, G.K. Rand. // Journal of the Operational Research Society. — 1987. — № 38. — P. 293–295.

20. Fischetti Matteo, Toth Paolo, Vigo Daniele, A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs [Text] / Matteo Fischetti, Paolo Toth, Daniele Vigo // Operation Research. – 1994. – Vol. 42, № 5. – P. 846-859.

21. John E. Mitchell, Branch-and-Cut Algorithms for Combinatorial Optimization Problems [Text] / E. Mitchell John // Mathematical Sciences Rensselaer Polytechnic Institute Troy, NY, USA. – 1999. – P.19.

22. Ryan D. M., Hjorring C., Glover F., Extensions of the Petal Method for Vehicle Routing [Text] / D. M. Ryan, C. Hjorring, F. Glover // Journal of the Operational Research Society. – 1993. – P.289-296.

23. GUlert B.E., Miller L.R., A heuristic algorithm for the vehicle dispatch Problem [Text] / B.E. GUlert, L.R. Miller // Operation Research. – 1974. – Vol.22 – P. 340-349.

24. Croes G.A., A Method for Solving Traveling-Salesman Problems [Text] / G.A. Croes // Operations Research. – 1958. – Vol. 6, № 6. – P. 791-812.

25. Hartmanis J., Stearns R. E., On the computational complexity of algorithms [Text] J. Hartmanis, R. E. Stearns // Transactions of the American Mathematical Society. – 1965. – P. 285-306.

26. Michael Garey, David S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness [Text]/ Garey Michael, S. Johnson David// New York: W. H. Freeman & Co. – 1979.

27. Sathyan Anoop, Boone Nathan, Cohen Kelly, Comparison of Approximate Approaches to Solving the Travelling Salesman Problem and its

Application to UAV Swarming [Электронный ресурс]/ Anoop Sathyan, Nathan Boone, Kelly Cohen // University of Cincinnati, USA. – 2015. – P. 17.

ПРИЛОЖЕНИЕ А
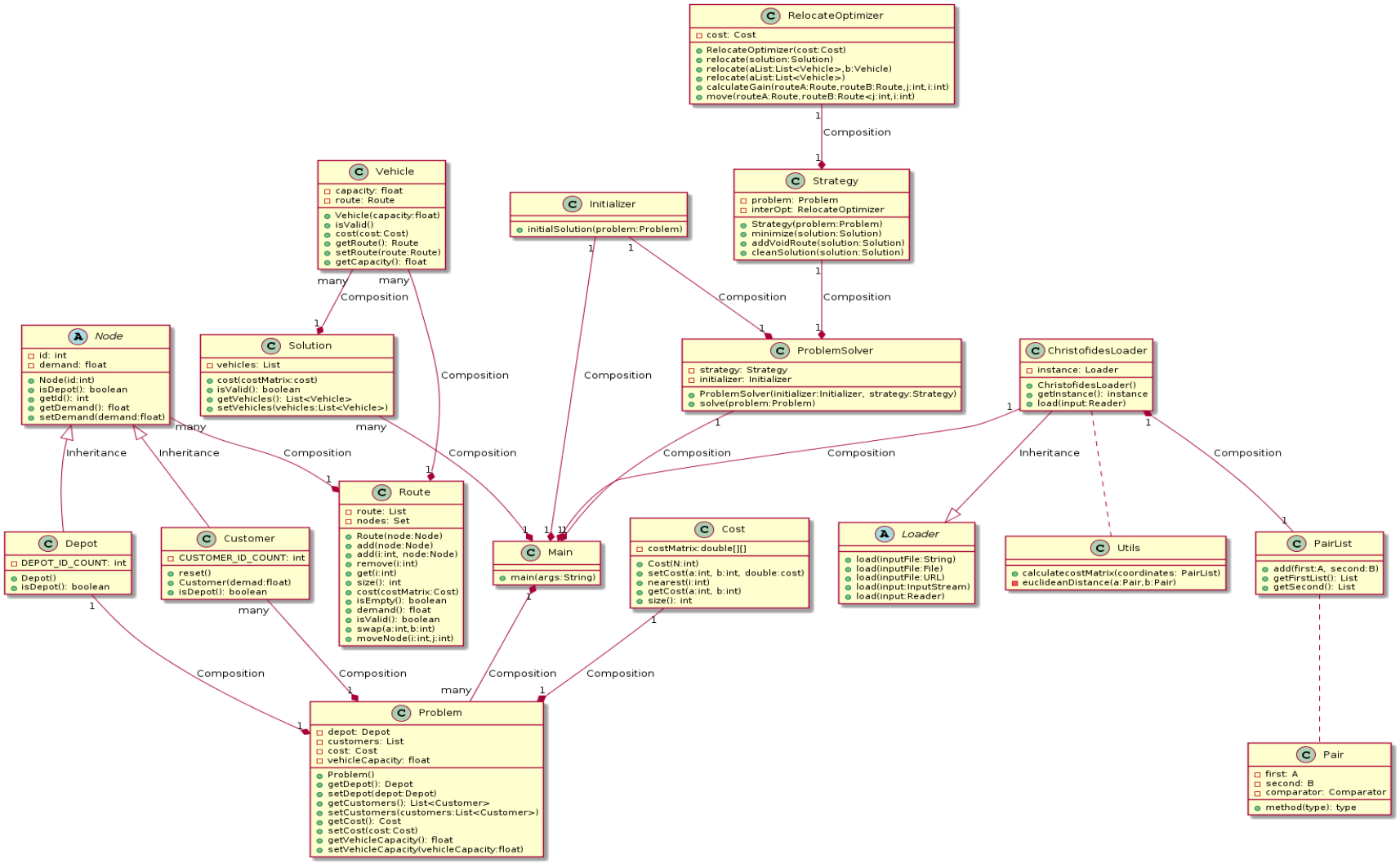
Диаграмма классов