**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»,  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 г. |

**Выпускная квалификационная работа**

на тему **ПРОГРАММА РАЗРАБОТКИ МАРШРУТОВ И СОСТАВЛЕНИЯ ГРАФИКОВ ДОСТАВКИ ТОВАРОВ КОМПАНИИ АВТОМОБИЛЬНЫМ ТРАНСПОРТОМ**

по направлению подготовки 09.03.04 «Программная инженерия»

|  |  |
| --- | --- |
| Научный руководитель  профессор департамента  программной инженерии  Должность, место работы  \_\_\_\_\_\_ к.т.н. \_\_\_\_\_\_\_\_\_  ученая степень, ученое звание  \_\_\_С.М.Авдошин\_\_\_\_\_  И.О. Фамилия  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Оценка  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись, Дата | Выполнил  студент группы \_БПИ122\_  4 курса бакалавриата  образовательной программы «Программная инженерия»  \_\_\_Д.С. Лянге\_\_\_\_\_\_\_\_\_\_\_\_\_  И.О. Фамилия  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись, Дата |

**Москва 2016**

# Реферат

Проблема составления маршрутов транспортных средств – комбинаторная задача оптимизации, касающаяся логистической сферы услуг компаний по доставке.

Приложение для решения проблемы распределения заказов, описанное в этой работе, нацелено на обеспечение лучшей эффективности решения задачи распределения и доставки заказов, учитывая реальные параметры компании-поставщика и используя эвристические алгоритмы. Эта проблема является важной из-за влияния, оказываемого стоимостью транспортировки на общие затраты на поставленные товары.

В работе приведена спецификация проблемы. Кроме того, описаны используемая модель, алгоритмы и критерии оптимизации решения. Предложенное решение проблемы распределения реализовано в форме приложения.

Работа содержит 45 страниц, 3 главы, 10 рисунков, 2 таблицы, 7 схем, 39 источников, 4 приложения.

***Ключевые слова:*** *проблема маршрутизации транспортных средств, проблема распределения и доставки, метаэвристические алгоритмы.*

# Abstract

The Vehicle Routing Problem is a combinatorial optimization problem which concerns the logistics service of a delivery company.

The application to resolve the distribution problem, described in this paper, is aimed at providing the best efficient performance while operating on real life parameters of goods distribution and delivery service company using the heuristic algorithms. This issue is important because of the influence transportation has on costs of delivered goods.

In this paper the specification of the problem is provided. Moreover, used model, algorithms and criteria of optimization are described. Proposed solution of distribution problem is developed in the form of an application.

The paper contains 45 pages, 3 chapters, 10 illustrations, 2 tables, 7 schemes, 39 bibliography items, 4 appendices.

***Keywords:*** *vehicle routing problem, distribution and delivery problem, metaheuristics.*

# Основные определения, термины и сокращения

**Задача комбинаторной оптимизации** – переборная задача, имеющая своей целью поиск оптимального решения.

**Эвристический алгоритм** (**эвристика**) – алгоритм решения, не гарантирующий правильность решения во всех случаях своего применения, но отрабатывающий правильно в большинстве случаев.

**Метаэвристический** **алгоритм** (**метаэвристика**) – высокоуровневая эвристика.

**Класс NP** (от non-deterministic polynomial) – класс задач разрешимости, которые можно за полиномиальное время решить на недетерминированной машине Тьюринга.

**NP-трудная задача** - задача из класс NP, к которой можно свести любую другую задачу из этого класса за полиномиальное время.

**TSP** (**Travelling Salesman Problem**; **задача коммивояжера**) - задача комбинаторной оптимизации, заключающаяся в отыскании самого выгодного маршрута, проходящего через указанные точки хотя бы по одному разу с последующим возвратом в исходную точку.

**VRP** (**Vehicle Routing Problem**; **классическая VRP-задача; задача маршрутизации**) – задача, являющаяся обобщением задачи коммивояжера, отвечающая на вопрос выбора оптимального набора маршрутов посещения различных локаций для множества транспортных средств.

Модификации VRP (насыщенные модели):

**VRPTW** (**Vehicle Routing Problem with Time Windows**) – добавляются временные ограничения на доставку заказов.

**VRPsTW** (**Vehicle Routing Problem with soft Time Windows**) – временные ограничения на доставку заказов допустимо нарушать, но для таких решений начисляется штраф.

**CVRP** (**Capacitated Vehicle Routing Problem**) – добавляются ограничения на грузоподъемность транспортных средств.

**VRPMT** (**Vehicle Routing Problem with Multiple Trips**) – добавляется возможность совершать несколько маршрутов для каждого из транспортных средств.

**OVRP** (**Open Vehicle Routing Problem**) – транспортные средства не обязаны возвращаться в начальную точку в конце маршрута.

**MDVRP** (**Multiple Depot Vehicle Routing Problem**) – транспортные средства могут стартовать с различных складов.

**SPDVRP** (**Simultaneous Pickup-and-Delivery VRP**) – транспортные средства не начинают движение, загруженные товарами, а могут останавливаться и загружаться в течение маршрута.

**API** (**интерфейс программирования приложений**) – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

**Маршрутный лист** – пронумерованный список заказов, которые должен доставить определенный курьер на транспортном средстве.

Оглавление

[Реферат 2](#_Toc451818810)

[Abstract 3](#_Toc451818811)

[Основные определения, термины и сокращения 4](#_Toc451818812)

[Введение 8](#_Toc451818813)

[Глава 1. Решения VRP-задач 10](#_Toc451818814)

[1.1. Обзор решений классической модели VRP 10](#_Toc451818815)

[1.1.1. “Vrphlibrary” 10](#_Toc451818816)

[1.1.2. “Vroom” 10](#_Toc451818817)

[1.1.3. “OpenVRP” 10](#_Toc451818818)

[1.1.4. “Jsprit” 10](#_Toc451818819)

[1.1.5. “SYMPHONY” 11](#_Toc451818820)

[1.1.6. “vrpmhsolver” 11](#_Toc451818821)

[1.1.7. “OscarLib” 11](#_Toc451818822)

[1.1.8. “VRP Spreadsheet Solver” 11](#_Toc451818823)

[1.1.9. “OptaPlanner” 11](#_Toc451818824)

[1.1.10. “Fast VRP” 11](#_Toc451818825)

[1.1.11. “VRP solver” 11](#_Toc451818826)

[1.2. Обзор решений насыщенных моделей VRP 12](#_Toc451818827)

[1.2.1. “GraphHopper” 12](#_Toc451818828)

[1.2.2. “Open Door Logistics” 13](#_Toc451818829)

[1.2.3. “Муравьиная логистика” 14](#_Toc451818830)

[1.2.4. “Optimo route” 14](#_Toc451818831)

[1.2.5. “Routyn” 15](#_Toc451818832)

[1.2.6. “Routific” 15](#_Toc451818833)

[1.2.7. “RouteXl” 16](#_Toc451818834)

[1.3. Анализ решений VRP 17](#_Toc451818835)

[Выводы по главе 18](#_Toc451818836)

[Глава 2. Задача построения маршрутов доставки товаров 19](#_Toc451818837)

[2.1. Классическая модель VRP 19](#_Toc451818838)

[2.2. Выбранная модель VRP 19](#_Toc451818839)

[2.3. Постановка задачи 20](#_Toc451818840)

[2.4. Используемые алгоритмы 24](#_Toc451818841)

[2.4.1. Hill Climbing Algorithm 25](#_Toc451818842)

[2.4.2. Tabu Search 27](#_Toc451818843)

[2.4.3. Simulated Annealing 27](#_Toc451818844)

[2.4.4. Late Acceptance Hill Climbing Algorithm 29](#_Toc451818845)

[2.4.5. Step Counting Hill Climbing Algorithm 29](#_Toc451818846)

[2.4.6. Genetic Algorithm 31](#_Toc451818847)

[2.5. Анализ алгоритмов решения VRP-задач 33](#_Toc451818848)

[2.6. Анализ полученных результатов применения реализованных алгоритмов решения VRP-задач 35](#_Toc451818849)

[Выводы по главе 36](#_Toc451818850)

[Глава 3. Разработка программы 37](#_Toc451818851)

[3.1. Инструменты разработки 37](#_Toc451818852)

[3.2. Использование API сторонних сервисов 37](#_Toc451818853)

[3.3. Модель спроектированной базы данных 38](#_Toc451818854)

[3.4. Структура Excel-документов 40](#_Toc451818855)

[3.5. Особенности прокладывания маршрутов доставки заказов в городе Москве 40](#_Toc451818856)

[Выводы по главе 41](#_Toc451818857)

[Заключение 42](#_Toc451818858)

[Список использованных источников 43](#_Toc451818859)

# Введение

Логистическое планирование доставки товаров было проблемой для человечества со времен начала потребности в транспортировке грузов в древнем мире. Современная глобализация и постоянно увеличивающееся число способов доставлять товары только усложнили принятие решения в этой области предпринимательства, создавая потребность в централизованном поиске и внедрении решения вопросов эффективности доставки для современных компаний.

Продвижение науки в прошлых веках позволило математически сформулировать определение проблемы и таким образом создать способ структурировать алгоритмы, подходящие для ее решения. VRP-задача используется для нахождения оптимального набора маршрутов для достижения всех клиентов имеющимся парком транспортных средств. Обобщая TSP, VRP - одна из самых известных комбинаторных проблем оптимизации, и хотя она не имеет оптимального решения, будучи NP-трудной, многие дифференцирующиеся в подходах, числе параметров и затраченных для подсчета ресурсах, алгоритмы могут быть применены, чтобы получить результаты с необходимой точностью и эффективностью. Детальное изучение моделей VRP-задач, начиная с введения термина [8], привело к увеличению разнообразия среди VRP-моделей и сужению рамок исследования и разработке алгоритмов решения для отдельных моделей. Общая тенденция исследования VRP сосредотачивается на более практических проблемах доставки, нежели описанных в стандартных моделях базовых параметрах [15].

Важность низких расходов на транспортировку и их последующее влияние на стоимость доставки товаров многократно увеличивают продуктивность и сбережения компании при интеграции решения VRP, тем самым искореняя все сомнения в актуальности исследования VRP, несмотря на длительность существования этой проблемы.

В рамках данной квалификационной работы была поставлена цель решить конкретную практическую проблему распределения и составления маршрутов доставки. Цель исследования состоит в том, чтобы внедрить различные алгоритмы для решения VRP, переведя алгоритмы из теоретической модели в географическую карту города и оценить эффективность их работы с различными наборами реальных данных и трудностями, возникающими в сфере повседневных услуг доставки, чтобы создать приложение, применяющее реализованные алгоритмы для создания легких для чтения и применения маршрутных листов и карт доставки товаров.

Для достижения этой цели были поставлены и решены следующие задачи:

1. Выбрать VRP-модель, описывающую поставленную задачу доставки заказов;
2. Изучить алгоритмы и существующие подходы решения VRP-задач, выбрать несколько алгоритмов из различных классов для реализации;
3. Выбрать критерии оптимизации для реализуемых алгоритмов;
4. Реализовать выбранные алгоритмы для применения их на выбранной модели;
5. Разработать программу, решающую VRP-задачу для выбранной модели при помощи реализованных алгоритмов;
6. Провести тестирование алгоритмов, сравнить и обосновать полученные результаты;
7. Разработать пользовательский интерфейс управления программой;
8. Разработать техническую документацию.

Работа организована следующим образом: в первой главе производится обзор и сравнение основных групп аналогов, решающих VRP-задачи. Во второй главе описываются классическая и выбранная VRP-модели, приводится математическая постановка решаемой задачи, а также описываются и сравниваются используемые и существующие алгоритмы решения. В третьей главе описаны особенности программной реализации разработанного приложения.

# Глава 1. Решения VRP-задач

В данной главе проводится обзор и сравнение ключевых групп приложений, а также их наиболее заметных представителей, занимающихся решением VRP-задач.

Все приложения и инструменты, которые можно найти в сети Интернет, делятся на два основных типа: предназначенные для теоретического решения классической VRP-задачи и для практического решения насыщенных VRP-моделей.

## 1.1. Обзор решений классической модели VRP

Проекты первого типа стремятся объединить большое число известных эвристик для решения классической проблемы VRP. Эта цель достигается либо посредством создания крупных интернет-библиотек, содержащих набор эвристик и метаэвристик, либо посредством создания приложений с возможностью визуализации полученных решений на простых графических моделях VRP.

Примерами проектов первого типа являются:

### 1.1.1. “Vrphlibrary”

Библиотека, реализованная на языке C++, позволяющая использовать эвристики и метаэвристики для решения классической и двух самостоятельно разработанных авторам моделей VRP, а также подключать сторонние библиотеки для создания двумерных графиков. Не предоставляет возможность проверить оптимальность выбранной эвристики [18].

### 1.1.2. “Vroom”

Библиотека, реализованная на языке Java, позволяющая моделировать классическую задачу VRP, а также применять к ней две реализованные эвристики [24].

### 1.1.3. “OpenVRP”

Библиотека, реализованная на языке Lisp, позволяющая моделировать классическую задачу VRP, а также применять к ней одну реализованную эвристику. Находится в состоянии разработки.

### 1.1.4. “Jsprit”

Библиотека, реализованная на языке Java, решающая классическую модель VRP-задачи посредством одной реализованной эвристики и позволяющая учитывать дополнительные параметры для развозимых товаров.

### 1.1.5. “SYMPHONY”

Библиотека, реализованная на языке С, моделирующая решение [27] классической задачи VRP.

### 1.1.6. “vrpmhsolver”

Библиотека, позволяющая решать классическую задачу VRP посредством применения трех различных эвристик.

Примерами второго типа проектов можно назвать:

### 1.1.7. “OscarLib”

Библиотека, реализованная на языках Java/Scala, предназначенная для решения сложных оптимизационных проблем, в том числе и проблему маршрутизации.

### 1.1.8. “VRP Spreadsheet Solver”

Библиотека, реализованная на языке Visual Basic и предназначенная для интеграции с Microsoft Excel, позволяющая решать задачу маршрутизации [31] для этой платформы.

### 1.1.9. “OptaPlanner”

Приложение, реализованное на языке Java, оптимизирующее управление бизнес-ресурсами, позволяющее в том числе решать и классическую VRP-задачу.

### 1.1.10. “Fast VRP”

Приложение, реализованное под GAE, с удобным 2D интерфейсом нанесения промежуточных точек для графического решения VRP-задачи для нескольких транспортных средств.

### 1.1.11. “VRP solver”

Приложение, позволяющее решать классическую задачу VRP и предоставляющее решение в графическом и текстовом видах.

Преимуществом теоретического подхода можно назвать возможность аккумуляции большого числа эвристик и метаэвристик, использующих основные базовые параметры классической модели VRP и отвечающих только на вопрос оптимизации времени доставки всех заказов.

Общим недостатком такого теоретического типа решений в рамках поставленной в этой работе задачи разработки маршрутов доставки товаров является отсутствие возможности применить реализованные эвристики на реальных данных любой сложности, поскольку классические реализации эвристик не предполагают использование дополнительных параметров или ограничений сверх уже имеющихся в оригинальной постановке задачи VRP. Также ограниченные средства визуализации, если таковые имеются, не позволяют интерпретировать входящие и выходящие данные на географическую местность. Возможна интерпретация результатов в виде графов рис. 1. Возможна работа только с ограниченно узким набором VRP-моделей, близких к базовой (CVRP, VRPTW).

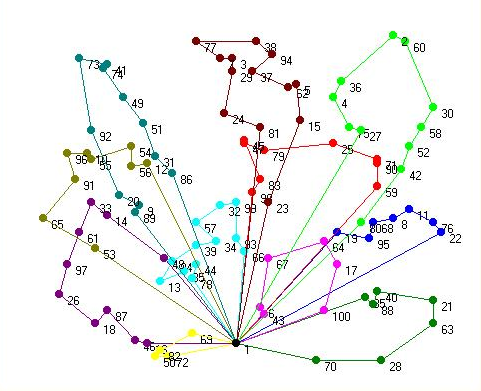


Рисунок 1. Стандартный вид визуализации решения классической задачи VRP [35]

## 1.2. Обзор решений насыщенных моделей VRP

Проекты второго типа ставят своей целью применение изученных решений VRP на современные логистические проблемы. Для достижения данной цели создаются наборы инструментов и библиотеки для решения сложных и насыщенных моделей VRP-задач, которые после используются для составления карт маршрутов.

### 1.2.1. “GraphHopper”

Дорожная библиотека маршрутизации, использующаяся как механизм для создания интерфейса GraphHopper Maps, сервиса составления маршрутов на карте. Приложение используется для создания маршрутов, имеющих сильно удаленные промежуточные точки, находящиеся, к примеру, в различных городах, как показано на рис. 2, а также строит свои карты на базе карт OpenStreetMap, не всегда отличающихся своей детальностью ввиду открытого и непринудительного заполнения рядовыми пользователями.

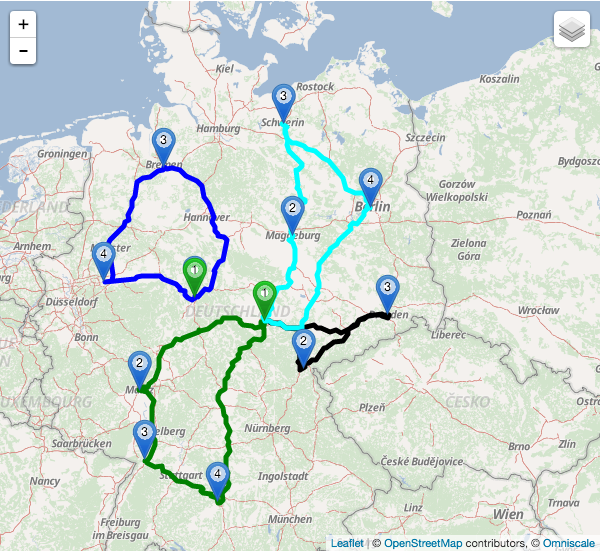


Рисунок 2. Построение маршрутов приложением “GraphHopper Maps” [17]

### 1.2.2. “Open Door Logistics”

Сервис маршрутизации, позволяющий считывать данные из xls-таблиц и учитывающий многие параметры транспортных средств в насыщенных VRP-задачах для построения маршрутов на карте рис. 3 и выдачи маршрутных листов.

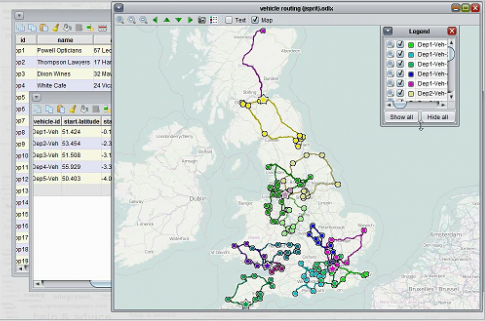


Рисунок 3. Интерфейс построения маршрутов приложением “Open Door Logistics” [33]

### 1.2.3. “Муравьиная логистика”

Русскоязычный сервис планирования маршрутов рис. 4, принимающий в учет ограниченное число дополнительных параметров, описывающих транспортные средства и временные рамки доставки заказов.

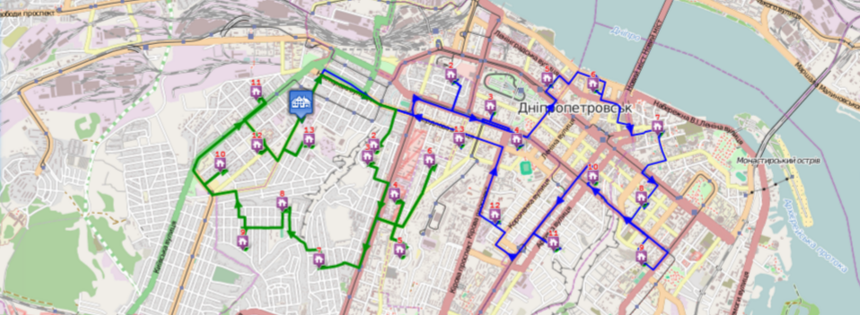


Рисунок 4. Маршруты, проложенные приложением “Муравьиная логистика” [37]

### 1.2.4. “Optimo route”

Сервис маршрутизации, использующий open source карты для прокладывания маршрутов рис. 5.

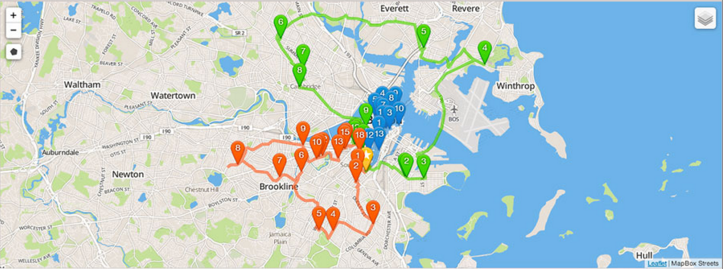


Рисунок 5. Построение маршрутов приложением “Optimo route” [23]

### 1.2.5. “Routyn”

Сервис, позволяющий отслеживать рис. 6 и управлять передвижением флота транспортных средств.

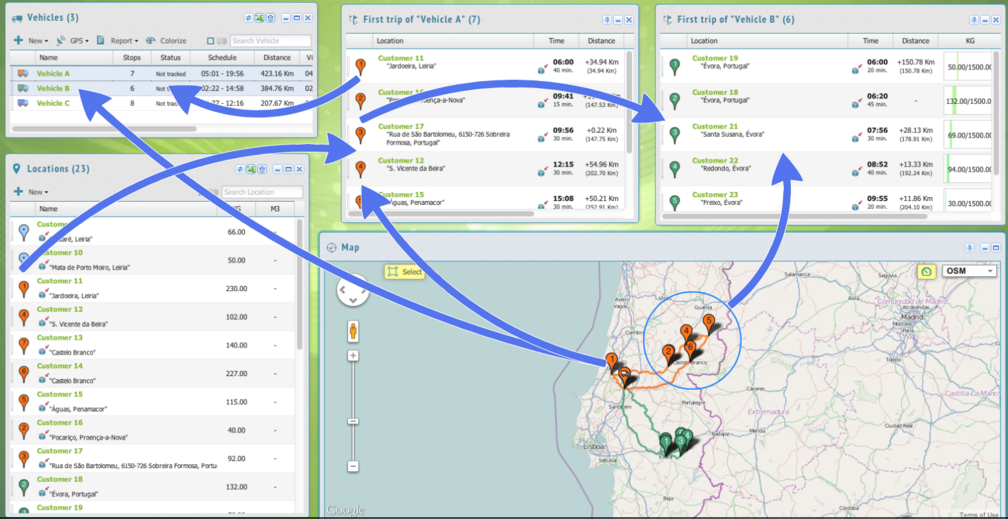


Рисунок 6. Интерфейс отслеживания маршрутов приложения “Routyn” [26]

### 1.2.6. “Routific”

Сервис, позволяющий отслеживать рис. 7 передвижение флота транспортных средств.

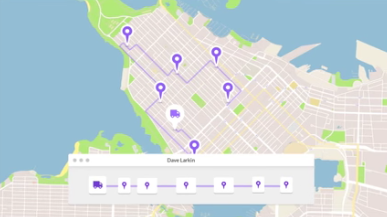


Рисунок 7. Интерфейс отслеживания маршрутов приложения “Routific” [25]

### 1.2.7. “RouteXl”

Сервис, строящий маршрут по заданному набору адресов.

Проекты второго типа ставят своей целью применение изученных решений VRP на современные логистические проблемы. Для достижения данной цели создаются наборы инструментов и библиотеки для решения сложных и насыщенных моделей VRP-задач, которые после используются для составления карт маршрутов.

Преимуществом проектов такого подхода можно назвать возможность применения решений для реальных картографических данных, возможность использования в качестве входных данных несколько форматов данных и наглядность предоставления результатов (в виде маршрутных листов и карт).

Основными недостатками являются невозможность выбора эвристик, применяемых к поставленной задаче маршрутизации, ограниченное число параметров для всех участвующих в решении задачи VRP сущностей, отсутствие возможности настройки критериев оптимизации найденных результатов и ограниченность моделей предоставленных для решения задач VRP.

## 1.3. Анализ решений VRP

Общее сравнение библиотек и приложений решений VRP-задач (табл. 1), подробно описанных ранее в тексте главы, производилось по нескольким критериям:

1. используемая VRP-модель;
2. возможность выбора эвристики для решения VRP-задачи (столбец 2 в табл. 1);
3. возможность визуализации решения в виде графов (столбец 3 в табл. 1);
4. возможность визуализации решения на географической карте (столбец 4 в табл. 1);
5. возможность представления полученного решения в виде текстовых таблиц (столбец 5 в табл. 1);
6. возможность бесплатного использования приложения/библиотеки (столбец 6 в табл. 1);
7. возможность ввода данных из различных типов источников (столбец 7 в табл. 1);
8. наличие нескольких (больше одного) критериев оптимизации построения маршрутов и возможность переключения между ними (столбец 8 в табл. 1).

Таблица 1.

Таблица сравнения библиотек эвристик и приложений, решающих VRP-задачи

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название приложения | Модели VRP-задач | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Vrphlibrary | VRP, custom VRP | - | - | - | - | + | - | - |
| Vroom | VRP | + | - | - | - | + | - | - |
| OpenVRP | VRP | - | - | - | - | + | - | - |
| Jsprit | CVRP | + | - | - | - | + | - | - |
| SYMPHONY | VRP | + | - | - | - | + | - | - |
| vrpmhsolver | VRP | + | - | - | - | + | - | - |
| OscarLib | CVRP | + | + | - | - | + | - | - |
| VRP Spreadsheet Solver | VRP | + | + | - | - | + | - | - |
| OptaPlanner | CVRPTW | + | + | - | + | + | + | - |
| Fast VRP | VRP | - | + | - | - | + | - | - |
| VRP solver | VRP | - | + | - | + | + | - | - |
| GraphHopper | CMDVRPTW | + | - | + | + | - | - | - |
| Open Door Logistics | COVRPTW | - | - | + | + | - | + | - |
| Муравьиная логистика | CMDVRPTW | - | - | + | + | - | + | + |
| Optimo route | CVRPTW | - | - | + | - | - | + | + |
| Routyn | CMDVRPTW | - | - | + | + | - | - | - |
| Routific | CVRPTW | - | - | + | - | - | - | - |
| RouteXl | CMDOVRPTW | - | - | + | - | - | - | - |

Как показал анализ выбранных библиотек и приложений, решения, оперирующие на классической и близких к ней моделях VRP, обычно реализуют несколько эвристик для решения поставленной задачи маршрутизации, используют только один критерий оптимизации и не имеют возможности решать задачу на географических данных. Решения же, оперирующие на более сложных моделях VRP и ставящие своей целью решение логистических задач на действительных географических данных, в большинстве своем не предоставляют возможности выбора используемого для нахождения решения алгоритма и используют всего один критерий оптимизации. Также решения такого типа, ориентированные на практическое решение VRP-задач, несмотря на платность использования собственных сервисов, накладывают довольно сильные ограничения на входные данные (число маршрутов/транспортных средств/развозимых заказов), что сильно затрудняет настраивание приложения под нужды и требования каждого отдельного пользователя.

Целью приложения, описываемого в этой работе, по итогам анализа найденных при сравнении существующих приложений трудностей стала реализация решения задачи VRP на насыщенной модели CMDOVRPTW, позволяющая выбирать используемые эвристики и критерии оптимизации, используя реальные данные для вывода построенных маршрутов на географическую карту. Полезными были также признаны критерий возможности выбора формата входных данных и вывода полученных результатов в виде маршрутных листов.

## Выводы по главе

В первой главе был проведен обзор и анализ основных библиотек и приложений, занимающихся решением VRP-задач. По итогам анализа были выбраны ключевые критерии реализации приложения (насыщенная CMDOVRPTW-модель решения задачи, выбор эвристик и критериев оптимизации решения), а также выработаны требования к числу форматов входных и выходных данных.

# Глава 2. Задача построения маршрутов доставки товаров

Данная глава посвящена описанию классической (рис. 8) и выбранной VRP- моделей; в ней также приводятся постановка задачи, описываются и сравниваются алгоритмы, выбранные для программной реализации.

## 2.1. Классическая модель VRP

Классическую модель отличают следующие особенности:

* наличие всего одного склада, с которого начинают движение транспортные средства;
* все транспортные средства являются гомогенными, т.е. обладают одинаковыми характеристиками;
* транспортные средства возвращаются в начальную точку маршрута;
* маршруты составляются на один день.

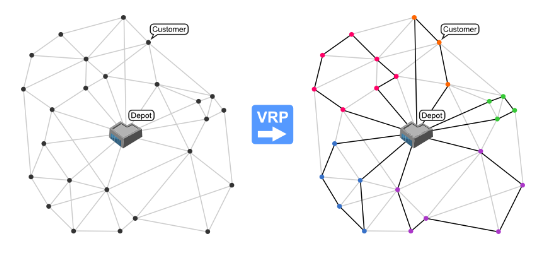


Рисунок 8. Модель классической задачи VRP (слева) и ее решение (справа) [34]

## 2.2. Выбранная модель VRP

CMDOVRPTW-модель описывают следующие особенности, дополняющие классическую VRP-модель:

* существует несколько складов, с которых транспортные средства могут начинать свои маршруты; для всех заказов известны склады, на которых они располагаются;
* транспортные средства имеют ограничения на грузоподъемность;
* транспортные средства не обязаны возвращаться в начальную точку маршрута;
* каждому транспортному средству присваиваются временные рамки, в которые оно может осуществлять свою работу;
* каждому заказу также присваиваются временные рамки, в которые оно может быть доставлено.

## 2.3. Постановка задачи

Описываемая в этой работе задача представляет собой решение насыщенной модели CMDOVRPTW (Capacitated Multiple Depot Open Vehicle Routing Problem with Time Windows) для города Москвы. Необходимо решить задачу распределения имеющегося набора заказов на один день по набору автомобилей и составить маршрутные листы для каждого из курьеров. Обозначим через

Пусть - множество автомобилей.

Зададим на множестве индексацию

,

где .

Будем использовать обозначение для элемента из множества , для которого

.

Поставим в соответствие каждому автомобилю вектор атрибутов, описывающих его:

,

где – грузоподъемность -го автомобиля, кг;

– максимальный вес для одного товара в заказе, помещающегося в -ый автомобиль, кг;

– вместительность -го автомобиля в длину, м;

– максимальная длина для одного товара в заказе, помещающегося в -ый автомобиль, м;

- скорость передвижения -го автомобиля, км/ч;

– время появление курьера на -ом автомобиле на складе (время начала работы курьера), ч/м/с;

– время доставки курьером на -ом автомобиле последнего по времени заказа (время завершения работы курьера), ч/м/с.

Пусть - множество заказов.

Зададим на множестве индексацию

,

где .

Будем использовать обозначение для элемента из множества , для которого

.

Поставим в соответствие каждому заказу вектор атрибутов, описывающих его:

,

где – общий вес всех товаров в -ом заказе, кг;

– максимальный вес одного товара в -ом заказе , кг;

– общая длина всех товаров в -ом заказе, м;

– максимальная длина одного товара в -ом заказе, м;

– адрес склада размещения -го заказа, координаты;

– адрес доставки -го заказа, координаты;

– верхняя временная граница доставки -го заказа, ч/м/с;

– нижняя временная граница доставки -го заказа, ч/м/с.

Пусть - множество планов доставки заказов на выбранный день. План доставки заказов состоит из множества различных листов доставки заказов, описывающих маршрут каждого из автомобилей.

,

Пусть – множество раличных листов доставки заказов.

Элемент этого множества представляет собой множество маршрутных листов.

,

где маршрутный лист представляет собой упорядоченный набор заказов, удовлетворяющих соотношению

Здесь - множество заказов в маршрутном листе .

Обозначим через множество заказов маршрутного листа .

Данные множества обладают следующими свойствами:

1. Все заказы должны быть распределены по маршрутным листам так, что индекс одного заказа должен встречаться не больше одного раза среди всех маршрутных листов.

2. Для каждого из автомобилей суммарный вес всех заказов этого автомобиля не должен превышать грузоподъемность автомобиля.

3. Для каждого из автомобилей суммарная длина всех заказов этого автомобиля не должна превышать установленную общую длину заказов автомобиля.

4. Для каждого заказа каждого из автомобилей максимальный вес любого товара не должен превосходить допустимый показатель веса одного товара этого автомобиля.

5. Для каждого заказа каждого из автомобилей максимальная длина любого товара не должна превосходить допустимый показатель длины одного товара этого автомобиля.

Введем функцию, подсчитывающую расстояние, пройденное автомобилем , у которой наступила очередь развозить заказ под номером r в маршрутном листе (заказ ). Расстояние высчитывается как сумма расстояний, пройденного автомобилем от склада до адреса первого заказа, и суммы попарных очередных расстояний между адресами последующих развезенных заказов до текущего заказа под номером r.

,

где - функция расстояния между двумя адресами, описываемыми координатами.

6. Для каждого автомобиля рассчитанное время доставки последнего заказа не должно превышать установленное время работы данного курьера.

7. Для каждого заказа каждого из автомобилей рассчитанное время доставки заказа должно находиться в границах интервала времени доставки, установленного для этого заказа.

Постановки задачи могут быть описаны такими способами в зависимости от применяемого критерия оптимизации:

1. Наименьшее время доставки всех заказов:

Введем функцию K, подсчитывающую максимальное время, потраченное каждым из курьеров для развоза всех заказов из своего маршрутного листа:

В таком случае, требуется найти такой набор маршрутных листов , что ,

2. Наименьшее число курьеров, использованных для доставки заказов:

Введем функцию , определяющую отсутствие заказов в маршрутном листе:

Введем функцию K, подсчитывающую число курьеров, незадействованных в развозе заказов (имеющих пустые маршрутные листы):

Требуется найти такой набор маршрутных листов , что ,

3. Минимальное суммарное пройденное расстояние:

Введем функцию K, подсчитывающую суммарное расстояние, затрачиваемое всеми курьерами для развоза всех маршрутных листов из одного набора:

Требуется найти такой набор маршрутных листов , что ,

## 2.4. Используемые алгоритмы

Все используемые в работе метаэвристики делятся на локальные и глобальные алгоритмы поиска оптимального решения.

Алгоритмы локального поиска (Local Search Algorithms) заключаются в случайной инициализации начального решения и постепенного превращения данного решения во все более и более оптимальное. На каждом цикле выполнения алгоритма берется текущее решение и рассматривается набор соседних решений (решения в окрестности, neighborhood solutions), в которые текущее решение может быть переведено посредством одного элементарного изменения (движение, move). Из движений выбирается наиболее оптимальное (дающее максимальное значение оценочной функции, соответствующей выбранному критерию оптимизации – минимальное время доставки заказов, минимальное число задействованных транспортных средств, минимальное покрытое расстояние). Если оно удовлетворяет критериям (является приемлемым, acceptable), присущим конкретному алгоритму локальной оптимизации, то текущее решение заменяется на выбранное оптимальное из окрестности (шаг, step) и цикл выполнения алгоритма запускается заново для нового решения. Если же критерии не удовлетворяются, то выполнение алгоритма прекращается (termination state) и текущее решение на момент остановки выполнения алгоритма считается оптимальным. Критерии могут описывать как параметры сравнения найденного в окрестности решения и предыдущих решений, так и, к примеру, общее число циклов выполнения алгоритма [3] [29]. Конкретные критерии для каждого из используемых в данной работе алгоритмов локального поиска будут приведены в описании самих алгоритмов.

Решением в данной постановке задачи CMDOVRPTW считается набор маршрутных листов, каждый из которых содержит набор заказов. Если решение является полным, то есть все имеющиеся заказы развозятся в срок, то движением (элементарным изменением) будем называть перемещение заказа из одного маршрутного листа в другой так, чтобы получившийся набор маршрутных листов все также являлся полным. Если же решение является неполным (существуют заказы, которые невозможно развести при накладываемых временных ограничениях и имеющимся флотом транспортных средств), то для таких изначально частично разрешимых случаев принимаются движения, не ухудшающие ситуацию (не уменьшающие число всех развозимых заказов).

Локальные алгоритмы имеют общий вид, описанный в схеме 1.

**Схема 1**. Local Search Algorithm



Для инициализации первичного решения используется эвристический алгоритм First Fit (поиск первого подходящего) [32], заключающийся в постепенном распределении всех имеющихся заказов по маршрутным листам так, чтобы минимизировать выбранный для основного алгоритма поиска решения критерий оптимизации (наименьшее число курьеров – максимальное заполнение маршрутных листов заказами; наименьшее время доставки заказов – максимально равномерное заполнение маршрутных листов; наименьшее расстояние – минимальные расстояния между заказами в одном маршрутном листе). При использовании данной эвристики решение не всегда получается полным из-за “жадности” алгоритма (выбор лучшей возможной позиции для каждого конкретного заказа не гарантирует оптимальности для общей картины распределения заказов), поэтому оно и используется благодаря своей высокой скорости получения для последующего улучшения при помощи метаэвристических алгоритмов локальной оптимизации.

### 2.4.1. Hill Climbing Algorithm

Hill Climbing Algorithm (Simple Local Search Algorithm) является локальным оптимизационным алгоритмом максимизации оценочной функции. Данный алгоритм берет все шаги из текущей окрестности, выбирает из них оптимальный (ведущий к решению с максимальным счетом) и выполняет шаг, если получаемое решение имеет не меньшее значение оценочной функции, нежели текущее. Если оптимальных шагов несколько, то выбирается любой из них. Далее процесс повторяется до тех пор, пока не будет достигнуто решение, для которого все возможные шаги ведут к меньшему результату оценочной функции [28].

Псевдокод алгоритма приведен в схеме 2.

**Схема 2**. Hill Climbing Algorithm



Поскольку Hill Climbing Algorithm всегда выбирает оптимальный шаг для текущего решения, данный алгоритм легко может завершить свою работу решением в локальном максимуме (рис. 9).

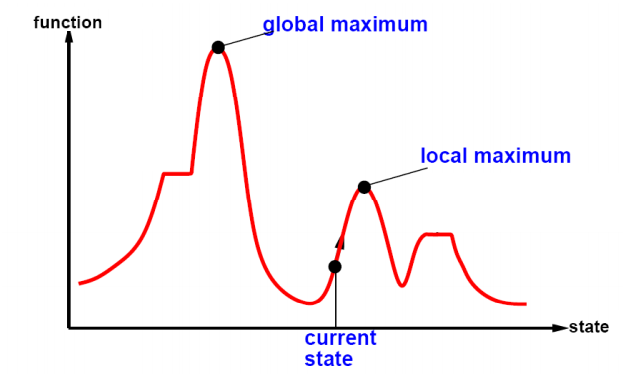


Рисунок 9. Ситуация попадания в локальный максимум при использовании Hill Climbing [22]

Описанные далее локальные алгоритмы оптимизации (Late Acceptance Hill Climbing Algorithm, Step Counting Hill Climbing Algorithm, Tabu Search, Simulated Annealing) являются улучшением Hill Climbing Algorithm, каждый по-своему решая проблему локальных экстремумов.

### 2.4.2. Tabu Search

Tabu Search (метод табу-поиска) так же, как и Hill Climbing Algorithm, ищет оптимальное значение оценочной функции на всех решениях из окрестности текущего, но исключает из них те, движения к которым затрагивают объекты, содержащиеся в табу-списке [12] [13]. Табу-список представляет собой набор объектов, не имеющих право в течение определенного времени после попадания в табу-список участвовать в движении. По истечении срока нахождения объекта в табуированном списке, движения с его участием снова становятся доступными. В список объекты попадают после выбора соответствующего шага, затрагивающего их, на текущем цикле выполнения алгоритма. Объектом может быть как само движение, так и элементы решения или само решение. В данной реализации алгоритма в виде объектов используются решения.

Псевдокод алгоритма приведен в схеме 3.

**Схема 3**. Tabu Search



### 2.4.3. Simulated Annealing

В отличие от двух предыдущих алгоритмов, Simulated Annealing рассматривает не все движения в окрестности, что значительно повышает скорость выполнения этого алгоритма. Просмотр движений прерывается при нахождении первого подходящего движения, которое и становится оптимальным шагом на этом цикле. Движение считается подходящим, если оно приводит к решению, не уменьшающему значение оценочной функции относительно текущего решения, или же уменьшающему, но проходящему случайную проверку. Шанс и частота выбора таких ухудшающих ситуацию шагов со временем уменьшается в связи с ростом декремента счета текущего решения и течением времени, представляемым температурными параметрами алгоритма, делая Simulated Annealing на поздних стадиях все более похожим на Hill Climbing Algorithm, принимая только улучшающие движения [1] [7].

Псевдокод алгоритма приведен в схеме 4.

**Схема 4**. Simulated Annealing



### 2.4.4. Late Acceptance Hill Climbing Algorithm

Late Acceptance Hill Climbing Algorithm так же, как и Simulated Annealing, рассматривает только часть движений в окрестности. Движение может быть выбрано как шаг в случае, если оно приводит к решению, не уменьшающему значение оценочной функции для текущего оптимального решения, или же к решению с не меньшим значением счета, чем у оптимального решения определенного числа циклов назад [4].

Псевдокод алгоритма приведен в схеме 5.

**Схема 5**. Late Acceptance Hill Climbing Algorithm



### 2.4.5. Step Counting Hill Climbing Algorithm

Step Counting Hill Climbing Algorithm отличается от Late Acceptance Hill Climbing Algorithm тем, что вместо сравнивания возможных движений со счетом оптимального решения определенное число циклов назад данный алгоритм хранит одну пороговую оценку на протяжении нескольких циклов, выбирая последующие движения только в том случае, если они не понижают данный порог [5].

Псевдокод алгоритма приведен в схеме 6.

**Схема 6**. Step Counting Hill Climbing Algorithm



Описанный далее генетический алгоритм оптимизации отличается от предыдущих алгоритмов тем, что является глобальным, а не локальным алгоритмом оптимизации.

Генетический алгоритм развивает поколение (набор) индивидуумов, представленных в виде хромосом, создавая новые поколения потомков посредством внесения изменений в текущее, до тех пор, пока не будут достигнуты определенные критерии терминирования алгоритма. К таким критериям в описываемой реализации алгоритма относится значение максимального числа поколений. По завершению выполнения алгоритма, лучшая произведенная хромосома в поколении расшифровывается и выбирается в качестве искомого решения [19] [14].

### 2.4.6. Genetic Algorithm

Для решения задачи VRP при помощи генетического алгоритма оптимизации была выбрана модель представления каждого индивидуума в поколении (обозначающего одно решение) только одной хромосомой, где каждая хромосома выглядит как цепочка заказов и транспортных средств. Транспортные средства представляют собой разделители между маршрутными листами, а заказы, расположенные после таких разделителей, показывают свою принадлежность к определенному маршрутному листу [20].

Создание нового поколения включает в себя три стадии:

селекция части индивидуумов из текущего поколения с наилучшими показателями оценочной функции; случайный выбор набора пар индивидуумов из текущего поколения и рекомбинация генов в хромосомах (операция кроссинговера) выбранных родителей внутри пары для получения двух потомков; мутация генов потомков для увеличения диверсификации индивидуумов (решений).

Описанные стадии создают из старого поколения новое, с одной стороны сохраняя качество в виде набора лучших хромосом, а с другой – добавляя операцией кроссинговера и мутацией диверсификацию в набор решений нового поколения.

Первое поколение получается аналогично стартовому решению для алгоритмов локального поиска, описанных ранее, при помощи эвристики First Fit, запускаемой определенное число раз для списков заказов в различных последовательностях.

Псевдокод алгоритма приведен в схеме 7.

**Схема 7**. Genetic Algorithm [21]



## 2.5. Анализ алгоритмов решения VRP-задач

Общее сравнение рассмотренных эвристик и метаэвристик для решения VRP-задач (табл. 2), описанных ранее в тексте главы, производилось по нескольким критериям:

1. тип алгоритма: эвристика (Heuristics, H), метаэвристика(Metaheuristics, MH), исчерпывающий поиск (Exhaustive Search, ES) (столбец 1 в табл. 2);
2. уязвимость к попаданию в локальные экстремумы (столбец 2 в табл. 2);
3. относительное время поиска решения (относительное в сравнении представленных алгоритмов друг с другом) (столбец 3 в табл. 2);
4. Область поиска решений для каждого цикла алгоритма: локальная (Local) или глобальная (Global) (столбец 4 в табл. 2).

Также в таблицу сравнения были включены некоторые популярные алгоритмы, используемые для решения VRP-задач, но не реализованные в конечном счете в приложении. Некоторые из них представляют собой интерес для дальнейшего изучения и внедрения в приложение, другие не имеют серьезных отличий от реализованных и поэтому были опущены. Из-за обилия эвристик и метаэвристик для решения VRP-задач, а также из-за существующего числа модификаций для каждого из алгоритмов, были выбраны базовые, но наиболее подходящие к выбранной модели и для решения поставленной задачи алгоритмы.

Таблица 2.

Таблица сравнения алгоритмов, решающих VRP-задачи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название алгоритма | 1 | 2 | 3 | 4 | Примечание |
| Реализованные: | | | | | |
| First Fit | H | - | Низкое | - | Используется в приложении для получения стартового решения для других алгоритмов |
| Hill Climbing Algorithm (Simple Local Search Algorithm) | MH | Наиболее уязвим | Среднее | Local | Из-за “жадности” при выборе каждого последующего решения, наиболее уязвим к локальным максимумам |
| Tabu Search | MH | Уязвим | Среднее | Local | Являясь в той или иной степени улучшением Simple Local Search Algorithm, также уязвимы к локальным максимумам, но используют различные способы оценки выбора следующего решения для достижения глобального максимума |
| Simulated Annealing | MH | Уязвим | Среднее | Local |
| Late Acceptance Hill Climbing Algorithm | MH | Уязвим | Среднее | Local |
| Step Counting Hill Climbing Algorithm | MH | Уязвим | Среднее | Local |
| Genetic Algorithm | MH | Неуязвим | Высокое | Global | Единственный алгоритм глобального поиска среди реализованных, отличается максимально долгим временем нахождения оптимального решения |
| Рассмотренные: | | | | | |
| Random Restart Hill Climbing Algorithm | MH | Наиболее уязвим | Среднее | Local | Отличается от обычного Hill Climbing Algorithm тем, что пытается избежать локального максимума, перезапуская алгоритм из случайных начальных решений, что не гарантирует нахождение глобального максимума, но значительно увеличивает время работы алгоритма |
| Local Beam Search | MH | Наиболее уязвим | Среднее | Local | Похож на Random Restart Hill Climbing Algorithm, но не реализует набор перезапусков алгоритма, а сразу генерирует набор случайных решений и параллельно выполняет на них Hill Climbing Algorithm |
| Brute Force | ES | Неуязвим | - | Global | Исчерпывающий поиск всегда находит оптимальное решение, создавая и оценивая все возможные решения (Brute Force) или наиболее обещающие ветви решений (Branch and Bound). Оба алгоритма практически неприменимы для решения реальных задач ввиду экспоненциального роста числа решений при росте числа распределяемых заказов [11] [2] |
| Branch and Bound | ES | Неуязвим | - | Global |
| Branch and Cut | ES | Неуязвим | - | Global |
| К дальнейшему изучению и внедрению: | | | | | |
| Great Deluge Algorithm | MH | Уязвим | Среднее | Local | Алгоритм локального поиска, описываемый авторами как улучшение алгоритма Simulated Annealing [9] [10] |

## 2.6. Анализ полученных результатов применения реализованных алгоритмов решения VRP-задач

Сравнение результатов применения реализованных алгоритмов для решения VRP-задач, описанных ранее в тексте главы, может быть описано следующим образом:

1. Hill Climbing Algorithm

Алгоритм показал наиболее быстрое получение результата среди остальных алгоритмов, но результаты получились худшими по качеству по причине неспособности алгоритма обходить локальные максимумы.

1. Tabu Search - Simulated Annealing - Late Acceptance Hill Climbing Algorithm - Step Counting Hill Climbing Algorithm

Результаты тестирования данных алгоритмов объединены по причине схожести их работы, так как все являются локальными алгоритмами оптимизации, сосредоточенными на решении проблемы избегания локальных максимумов и поиска глобального максимума значения оценочной функции. Из-за дополнительных условий выбора очередного решения в качестве оптимального на каждом цикле алгоритмы работали дольше, нежели простой алгоритм локального поиска, но и полученные результаты оказались намного эффективнее по значениям оценочной функции. Лучшие результаты по значению оценочной функции были показаны алгоритмом Simulated Annealing (но этому алгоритму требовалось больше итераций для нахождения решения, что выражалось в большей длительности работы), результаты же остальных трех алгоритмов несильно отличались друг от друга по эффективности и скорости при повторных запусках на одних и тех же данных.

1. Genetic Algorithm

Результаты применения генетического алгоритма оптимизации оказались хуже результатов всех остальных алгоритмов, за исключением Hill Climbing Algorithm. Отразилось это на большем времени поиска решения, что объясняется необходимостью оценивать и проверять на каждом цикле смены популяции большое число решений.

Все полученные результаты показали значительное улучшение во времени по сравнению с ручной системой распределения заказов по курьерам при отсутствии падения эффективности самих решений. Также у пользователей новой системы автоматизированного распределения появилась возможность осмысленно выбирать критерии оптимизации, вместо использовавшегося ранее условного смешанного критерия минимизации числа курьеров и времени доставки.

Полученные результаты сравнения можно легко соотнести и подтвердить при помощи видимых результатов выбора алгоритмов для решения задачи маршрутизации крупными приложениями, описанными во второй главе данной работы. Если про используемые алгоритмы на сайте приложений предоставляется информация, то реализованными оказываются в основном также алгоритмы локальной оптимизации, умеющие отсеивать локальные максимумы и стремиться к глобальному.

## Выводы по главе

Во второй главе были приведены описания классической (VRP) и используемой для решения поставленной задачи (CMDOVRPTW) VRP-моделей. Далее была сформулирована математическая постановка задачи и перечислены реализованные алгоритмы. По итогам сравнения реализованных алгоритмов были выявлены их уязвимости и относительные временные рамки работы. В конце главы представлен результат сравнения решений, полученных с использованием реализованных алгоритмов, с решениями, составленными вручную менеджерами до внедрения реализованной системы в компанию.

# Глава 3. Разработка программы

Данная глава посвящена описанию инструментов и методов, использованных при разработке приложения. Описаны использованные методы API сторонних сервисов и подробно показан формат входных данных, расписанный в виде модели спроектированной базы данных и используемых Excel-документов. В конце главы приведены особенности построения маршрутов доставки заказов в городе Москве.

## 3.1. Инструменты разработки

Для написания программы были использованы языки PHP, JavaScript и язык разметки HTML. Выбор обоснован спецификой данных языков, направленной на создание web-приложений, что являлось основным требованием по созданию интерфейса приложения. Также важную роль сыграли реализации API внешних систем, с которыми приложению требовалось работать, а именно Яндекс.Карты и геокодер Яндекс, имеющие реализацию как раз для вышеописанных языков.

Для написания приложения использовались php-библиотеки “curl” [6] и “SimpleXML” [30], обеспечивающие работу с HTTP-запросами и xml-форматом данных соответственно. HTTP-запросы использовались для работы с методами известного API, а xml-формат использовался для работы с Excel-документами.

## 3.2. Использование API сторонних сервисов

Для решения задачи распределения и доставки заказов требовалось перевести адреса, хранящиеся в системе текстовом виде, в вид географических координат для дальнейшего их использования в приложении. Для получения географических координат из адресов рассматривалось два наиболее популярных сервиса – Google Maps [16] и Яндекс Карты [36]. Окончательный выбор пал на сервис геокодирования Яндекса, отличающийся большей точностью определения географических наименований на русском языке и позволяющий задавать до 25000 запросов в день к своей функции геокодирования.

Для решения задачи отображения построенных маршрутов на карте также производился выбор двумя выше обозначенными сервисами, а также сервисом построения карт OpenStreetMap. По итогам сравнения оказалось, что OpenStreetMap, как и Народная Карта Яндекса, не дает достаточной точности в отображении маршрута, зачастую имеет устаревшие участки карты и не показывает дополнительной информации о построенном по точкам маршруте, кроме как общей траектории движения. При дальнейшем же изучении документации Яндекс. Карт и Google Maps выяснилось, что второй сервис имеет более строгие ограничения на число запросов к функциям своего API, а также не позволяет, в отличие от Яндекс.Карт, использовать статические карты, что значительно увеличивает размер передаваемых сервисом данных, увеличивая время ответа и общее время ожидания пользователем результатов работы приложения.

Итоговым выбранным стали статические карты Яндекс Карт, по ранее названным причинам представляющие собой сочетание оптимальной скорости отображения результата и приемлемые ограничения на число запросов в день (достаточное для поставленной задачи, где на день распределяется и доставляется по городу Москве не более 50 заказов) [39].

## 3.3. Модель спроектированной базы данных

Для описания поставленной задачи была спроектирована база данных, схема которой приведена на рис.10.

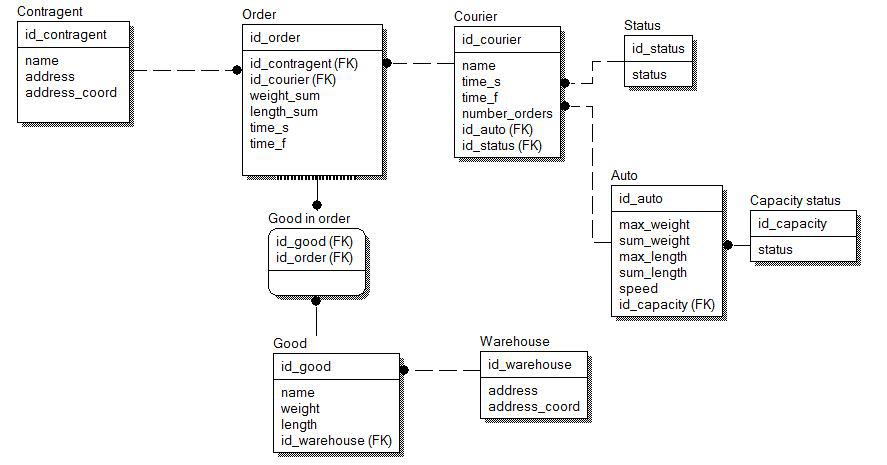


Рисунок 10. Модель базы данных

Описание сущностей:

Contragent: сущность, хранящая информацию о контрагенте, которому нужно доставить заказ.

* id\_contragent – идентификационный номер контрагента (первичный ключ);
* name – наименование контрагента;
* address – адрес в текстовом формате, на который нужно доставить заказы, предназначенные данному контрагенту;
* address\_coord – адрес, преобразованный в координаты.

Order: сущность, хранящая информацию о заказе.

* id\_order – идентификационный номер контрагента (первичный ключ);
* id\_contragent (FK) – внешний ключ, связывающий заказ с контрагентом;
* id\_courier (FK) – внешний ключ, связывающий заказ с курьером;
* weight\_sum – суммарный вес товаров в заказе;
* length\_sum – суммарная длина товаров в заказе;
* time\_s – верхняя временная граница доставки заказа;
* time\_f – нижняя временная граница доставки заказа.

Courier: сущность, хранящая информацию о заказе.

* id\_ courier – идентификационный номер курьера (первичный ключ);
* id\_auto (FK) – внешний ключ, связывающий курьера с автомобилем;
* id\_status (FK) – внешний ключ, связывающий курьера со статусом;
* time\_s – время начала работы курьера;
* time\_f – время окончания работы курьера;
* number\_orders – число заказов, которые курьер готов развести за день.

Status: сущность, хранящая информацию о статусе курьера.

* id\_ status – идентификационный номер статуса (первичный ключ);
* status – статус занятости курьера в процессе доставки заказов на выбранный день.

Auto: сущность, хранящая информацию об автомобиле.

* id\_ auto – идентификационный номер автомобиля (первичный ключ);
* id\_capacity (FK) – внешний ключ, связывающий автомобиль со статусом грузоподъемности;
* max\_length – максимальная длина товара, помещающегося в автомобиль;
* sum\_length – максимальная суммарная длина всех товаров, помещающихся в автомобиль;
* max\_weight – максимальный вес товара, помещающегося в автомобиль;
* sum\_weight – максимальный суммарный вес всех товаров, помещающихся в автомобиль;
* speed – скорость передвижения автомобиля.

Capacity status: сущность, хранящая информацию о статусе грузоподъемности автомобиля.

* id\_ capacity – идентификационный номер статуса грузоподъемности (первичный ключ);
* status – статус грузоподъемности автомобиля в определенное время суток для въезда в различные зоны города Москвы.

Good: сущность, хранящая информацию о товаре.

* id\_ good – идентификационный номер товара (первичный ключ);
* id\_warehouse (FK) – внешний ключ, связывающий товар со складом;
* name – наименование товара;
* length – длина товара;
* weight – вес товара.

Warehouse: сущность, хранящая информацию о товаре.

* id\_ warehouse – идентификационный номер склада (первичный ключ);
* address – адрес в текстовом формате, по которому находится склад;
* address\_coord – адрес, преобразованный в координаты.

Good in Order: агрегированная сущность, отражающая связь между заказами и товарами.

* id\_good (FK) – внешний ключ, связывающий сущность с товаром;
* id\_order (FK) – внешний ключ, связывающий сущность с заказом;

Средой развертывания данной модели стал MS SQL Server 2014, используемый компанией-заказчиком в силу своей простоты в обслуживании и поддержке, а также легкости развертывания и масштабируемости.

## 3.4. Структура Excel-документов

Структура используемых компанией для считывания входных данных Excel-документов аналогична структуре базы данных с тем отличием, что весь набор данных на определенный промежуток времени хранится в одном документе Excel, где отдельные листов представляют собой реализацию таблиц базы данных.

## 3.5. Особенности прокладывания маршрутов доставки заказов в городе Москве

Для учета ограничений, накладываемых на перемещения крупногабаритных транспортных средств по городу Москве [38], для которого решается задача маршрутизации, было принято решение разделить карту на 3 кольцевых сектора, где самый внешний сектор ограничивается снаружи Московским Малым Кольцом (ММК) и изнутри Московской Кольцевой Автомобильной дорогой (МКАД); средний сектор представляет собой часть города, расположенную между МКАД и Третьим Транспортным Кольцом (ТТК); третий сектор заключает собой всю центральную часть города внутри ТТК. ММК, МКАД и ТТК задаются как выпуклые многоугольники, заданные вершинами при помощи координат, полученных из геокодера Яндекса, где число вершин каждого из многоугольников определяется числом отрезков - прямых линий, которые можно провести по данным магистралям. Поскольку обслуживание заказов компанией происходит внутри ММК, случайно попавшие во входные данные заказы с адресами доставки вне ММК не учитываются при распределении. Область между ММК и МКАД не имеет никаких ограничений на грузоподъемность транспортных средств, в то же время как два других сектора – между МКАД и ТТР и внутри ТТР – накладывают ограничения на транспортные средства, которые могут развозить заказы по данным частям города. Таким образом транспортные средства, неподходящие под предъявляемые ограничения, не участвуют в процессе распределения заказов из данных областей Москвы и в данных временных рамках по курьерам.

## Выводы по главе

В третьей главе были приведены инструменты и методы, использованные при разработке приложения. Далее был изложен процесс выбора и использования API сторонних сервисов, и приведены модель используемых базы данных и Excel-документов. Также в данной главе описаны особенности доставки заказов в городе Москве.

# Заключение

Проблема разработки маршрутов доставки товаров – одна из самых сложных и востребованных логистических задач комбинаторной оптимизации, разнообразие подходов к решению которой порой затрудняет процесс выбора наиболее эффективного для конкретной задачи.

В рамках выполнения выпускной квалификационной работы было разработано приложение, позволяющее решать задачу распределения заказов и построения маршрутов доставки товаров автомобильным транспортом в городе Москве.

Для реализации программы была построена математическая постановка задачи и выбрана оптимальная модель VRP, а также были изучены и реализованы различные алгоритмы и критерии оптимизации решения VRP-задач.

В процессе выполнения дипломной работы были решены следующие задачи:

1. Выбрана VRP-модель, описывающая поставленную задачу доставки заказов;
2. Изучены алгоритмы и существующие подходы решения VRP-задач, выбрано и реализовано несколько алгоритмов из различных классов;
3. Выбраны критерии оптимизации для реализованных алгоритмов;
4. Разработана, протестирована и внедрена программа, решающая поставленную задачу распределения заказов и построения маршрутов доставки.

Направления дальнейшей работы включают в себя увеличение числа и разнообразия алгоритмов решения задачи маршрутизации, увеличение числа учитываемых при решении задачи параметров (таких как предпочитаемые курьерами округа для развоза заказов), а также расширении выбранной VRP-модели (к примеру, добавлением к текущей модели CMDOVRPTW возможности совершать одним транспортным средством несколько поездок в один день – CMDOVRPTW with Multiple Trips - CMDOVRPTWMT).

# Список использованных источников

1. Arbelaitz O., Rodriguez C. Low cost parallel solutions for the VRPTW optimization problem //International Journal of Computational Science and Engineering. – 2005. – Т. 1. – №. 2-4. – С. 175-182.
2. Blasum U., Hochstättler W. Application of the branch and cut method to the vehicle routing problem //Zentrum für Angewandte Informatik Köln Technical Report zpr2000-386. – 2000.
3. Bräysy O., Hasle G., Dullaert W. A multi-start local search algorithm for the vehicle routing problem with time windows //European Journal of Operational Research. – 2004. – Т. 159. – №. 3. – С. 586-605.
4. Burke E. K., Bykov Y. The late acceptance hill-climbing heuristic //University of Stirling, Tech. Rep. – 2012.
5. Bykov Y., Petrovic S. An initial study of a novel step counting hill climbing heuristic applied to timetabling problems //Proceedings of the 6th Multidisciplinary International Scheduling Conference: Theory & Applications (MISTA), Gent, Belgium. – 2013.
6. Client URL Library [Электронный документ] – Режим доступа: <http://php.net/manual/ru/book.curl.php>, свободный. (Дата обращения: 25.04.2016).
7. Czech Z. J., Czarnas P. Parallel simulated annealing for the vehicle routing problem with time windows //euromicro-pdp. – IEEE, 2002. – С. 0376.
8. Dantzig G. B., Ramser J. H. The truck dispatching problem //Management science. – 1959. – Т. 6. – №. 1. – С. 80-91.
9. Dueck G. New optimization heuristics: the great deluge algorithm and the record-to-record travel //Journal of Computational physics. – 1993. – Т. 104. – №. 1. – С. 86-92.
10. Dueck G., Scheuer T. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing //Journal of computational physics. – 1990. – Т. 90. – №. 1. – С. 161-175.
11. Fisher M. L. Optimal solution of vehicle routing problems using minimum k-trees //Operations research. – 1994. – Т. 42. – №. 4. – С. 626-642.
12. Glover F. Tabu search-part I //ORSA Journal on computing. – 1989. – Т. 1. – №. 3. – С. 190-206.
13. Glover F. Tabu search—part II //ORSA Journal on computing. – 1990. – Т. 2. – №. 1. – С. 4-32.
14. Goldberg D. E. et al. Genetic algorithms in search optimization and machine learning. – Reading Menlo Park : Addison-wesley, 1989. – Т. 412.
15. Golden B. L., Raghavan S., Wasil E. A. (ed.). The vehicle routing problem: latest advances and new challenges. – Springer Science & Business Media, 2008. – Т. 43.
16. Google Maps Geocoding API [Электронный документ] – Режим доступа: [https://developers.google.com/maps/documentation/geocoding/intro](https://developers.google.com/maps/documentation/geocoding/intro?hl=ru), свободный. (Дата обращения: 25.04.2016).
17. GraphHopper Maps [Электронный ресурс] – Режим доступа: <https://graphhopper.com/maps/> свободный. (Дата обращения: 25.04.2016).
18. Groer C. Parallel and serial algorithms for vehicle routing problems. – ProQuest, 2008.
19. Holland J. H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. – U Michigan Press, 1975.
20. Jaszkiewicz A., Kominek P. Genetic local search with distance preserving recombination operator for a vehicle routing problem //European Journal of Operational Research. – 2003. – Т. 151. – №. 2. – С. 352-364.
21. Liu B. Theory and practice of uncertain programming. – Heidelberg : Physica-verlag, 2002. – С. 78-81.
22. Local search algorithms [Электронный ресурс] – Режим доступа: https://www.cs.unc.edu/~lazebnik/fall10/lec06\_local\_search.pdf свободный. (Дата обращения: 25.04.2016).
23. Optimo route [Электронный ресурс] – Режим доступа: <https://optimoroute.com/> свободный. (Дата обращения: 25.04.2016).
24. Pillac V. Dynamic vehicle routing: solution methods and computational tools : дис. – Ecole des Mines de Nantes; universidad de los andes, 2012.
25. Routific [Электронный ресурс] – Режим доступа: <https://routific.com/> свободный. (Дата обращения: 25.04.2016).
26. Routyn [Электронный ресурс] – Режим доступа: <http://www.routyn.com/last-mile-edition/> свободный. (Дата обращения: 25.04.2016).
27. Schaus P., Hartert R. Multi-objective large neighborhood search //Principles and Practice of Constraint Programming. – Springer Berlin Heidelberg, 2013. – С. 611-627.
28. Selman B., Gomes C. P. Hill‐climbing Search //Encyclopedia of Cognitive Science. – 2006.
29. Shaw P. Using constraint programming and local search methods to solve vehicle routing problems //Principles and Practice of Constraint Programming—CP98. – Springer Berlin Heidelberg, 1998. – С. 417-431.
30. SimpleXML [Электронный документ] – Режим доступа: http://php.net/manual/en/book.simplexml.php, свободный. (Дата обращения: 25.04.2016).
31. Tas D. Time and reliability in vehicle routing problems : дис. – Technische Universiteit Eindhoven, 2013.
32. Thomas H.. Cormen et al. Introduction to algorithms. – Cambridge : MIT press, 2001. – Т. 6.
33. Vehicle routing & scheduling [Электронный ресурс] – Режим доступа: http://www.opendoorlogistics.com/software/vehicle-routing-scheduling/ свободный. (Дата обращения: 25.04.2016).
34. Vehicle Routing Problem [Электронный ресурс] – Режим доступа: <http://neo.lcc.uma.es/vrp/vehicle-routing-problem/> свободный. (Дата обращения: 25.04.2016).
35. VRP Solver [Электронный ресурс] – Режим доступа: https <http://coral.ie.lehigh.edu/~larry/software/vrp-solver/> свободный. (Дата обращения: 25.04.2016).
36. Геокодирование [Электронная страница] – Режим доступа: https://tech.yandex.ru/maps/doc/geocoder/desc/concepts/About-docpage/, свободный. (Дата обращения: 25.04.2016).
37. Муравьиная логистика [Электронный ресурс] – Режим доступа: <http://ant-logistics.com/features.html> свободный. (Дата обращения: 25.04.2016).
38. Правила проезда грузовых автомобилей в Москву [Электронный документ] – Режим доступа: http://www.sl-24.ru/pravila-proezda-Moscow.htm, свободный. (Дата обращения: 25.04.2016).
39. Условия использования сервиса “API Яндекс.Карты” [Электронный документ] – Режим доступа: <https://legal.yandex.ru/maps_api/>, свободный. (Дата обращения: 25.04.2016).