САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕОРИИ СИСТЕМ УПРАВЛЕНИЯ ЭЛЕКТРОФИЗИЧЕСКОЙ АППАРАТУРОЙ

Поборчий Игорь Всеволодович

Выпускная квалификационная работа бакалавра

Исследование эвристических методов решения задачи коммивояжера

Направление 010900
Прикладные математика и физика

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Гончарова А. Б.

Санкт-Петербург
2016

Содержание

Введение

В жизни современных предприятий самого разного рода существенное место занимают транспортные потоки: для каждой компании актуален вопрос о своевременной доставке товара потребителям в кратчайшие сроки. Для осуществления этого руководство компании, по сути, занимается решением так называемой задачи коммивояжера.

Представим себе ситуацию, в том или ином виде возникающую на любом предприятии, которое специализируется на доставке товаров и грузов. Допустим, курьеру компании необходимо развезти продукцию в определенное число мест и вернуться в офис. Задача формулируется следующим образом: определить в каком порядке курьер должен посещать клиентов, чтобы дорога заняла наименьшее время (все пункты посещаются один раз).

Задача коммивояжера в такой постановке является NP-трудной. Это означает, что не существует алгоритма, который находил бы точное решение задачи коммивояжера за полиномиальное время. Единственным же алгоритмом, который в принципе может гарантировать нахождение точного решения, является метод полного перебора. Однако работа программы, реализующей этот алгоритм, занимает адекватное время только при очень малой размерности входных данных (при числе пунктов < 20).

В связи с отсутствием эффективных точных методов решения задачи коммивояжера, становится необходимым использование методов эвристических. Эвристическими называют методы, которые будучи основанными на некой эвристике (правиле), не всегда следующей из строгих математических принципов, в подавляющем большинстве случаев дают решение, близкое к точному.

Таким образом, ставится следующая цель: исследовать различные эвристические методы решения задачи коммивояжера и оценить качество

полученного решения для задач с различным числом посещаемых пунктов.

Для реализации поставленной в работе цели ставятся следующие задачи:

- изучить единственный точный метод решения задачи коммивояжера – метод полного перебора,
- исследовать некоторые эвристические алгоритмы,
- программно реализовать их,
- проанализировать время и результаты работы программных реализаций алгоритмов,
- улучшить алгоритмы, если это возможно,
- разработать способ учета изменяющейся дорожной обстановки в методах,
- разработать способ генерации входных данных для задачи коммивояжера с учетом специфики постановки задачи,
- организовать возможность параллельной работы методов на разных экземплярах задач,
- сравнить время и результаты работы программных реализаций усовершенствованных алгоритмов с исходными.

Глава 1. Обзор транспортных задач. Актуальность задачи коммивояжера

1.1. Транспортные задачи

Характерным признаком любого современного мегаполиса является развитый рынок товаров и услуг. Перед компаниями, предлагающими эти компоненты потребительского рынка своим клиентам, стоит задача грамотно организовать работу, имеющихся в их распоряжении транспортных средств.

В связи с актуальностью проблемы, возникло большое число задач, которые можно объединить в общий класс «транспортных задач» [4,10,27,28]. В этот раздел входит огромное многообразие различных задач, которые связаны тем, что целевая функция в них носит тот или иной экономический смысл [2]. Серьезными работами в этом разделе математики отметились А.В. Левитин [11] и группа ученых во главе с Т.Х. Корменом [7]. Основательную классификацию транспортных задач в 2010 году осуществили Е.М. Бронштейн и Т.А. Заико в статье [2].

В статье [2] отмечены наиболее известные транспортные задачи. Так, например, классическая транспортная задача [1,2] заключается в развозке товара несколькими транспортными средствами определенному количеству потребителей. Теоретическая задача о загрузке рюкзака [16,20,24], имеющая своей целью наиболее выгодное наполнение ограниченного пространства, успешно применяется для оптимальной загрузки автотранспорта. Немаловажны и задачи теории расписаний [18], роль которых в современных реалиях нельзя недооценивать.

Особое место в классе транспортных задач занимает задача коммивояжера [13], заключающаяся в нахождении самого выгодного маршрута, проходящего через заданные пункты. Помимо огромной актуальности практического применения задача коммивояжера имеет серьезный теоретический смысл: она используется в качестве модели для

разработки эвристических алгоритмов различных оптимизационных задач. Особое место в данной области отводится программе Concorde TSP Solver, разработанной в 1990-х годах группой ученых для решения задачи коммивояжера с огромным числом вершин [22,23]. В настоящее время огромное внимание уделяется возможностям решения задачи коммивояжера и иных транспортных задач с помощью геоинформационных систем [4, 14].

Задачи коммивояжера различаются по типу графов, которые лежат в их основании. Так, существуют симметричные [12,17] и асимметричные [6] задачи коммивояжера. Также задачи коммивояжера делятся на статические и динамические, в зависимости от того, могут ли добавляться заказы в процессе работы алгоритма [2]. Решения в задаче могут искаться в виде гамильтоновых [6] или негамильтоновых [3] циклов.

1.2. Постановка задачи коммивояжера

Рассматривается следующая формулировка задачи [6]. Курьер должен доставить товар в определенное число мест и вернуться в пункт отправления. Определить, в каком порядке он должен объехать клиентов, чтобы дорога заняла наименьшее время (все пункты посещаются один раз).

В связи с тем, что данная работа посвящена решению задачи коммивояжера в условиях городского цикла, в ее основе лежит полный ориентированный граф. Это означает, что ребра существуют между каждой парой пунктов и «затраты» на проезд между двумя пунктами зависят от направления движения.

Под затратами в данной и в других работах автора подразумевается время. Термин «затраты» был использован в соответствии с распространенным в литературе термином «матрица затрат», определяющим матрицу, эквивалентную графу в задаче коммивояжера [12].

Как уже было отмечено ранее, решение задачи ищется среди гамильтоновых циклов, то есть циклов, в которых не происходит повторного посещения пунктов.

Сеть дорог для задачи представима графом $G = (V, E)$, где V – вершины графа, E– ребра графа. Вес ребра $C_{ij} > 0$ эквивалентен времени проезда между смежными вершинами графа.

Для удобства программной реализации граф представляется в виде матрицы, размерность которой соответствует количеству вершин в графе. Смысл значения элемента матрицы d_{ij} идентичен смыслу веса ребра C_{ij} в графе, оно определяет время проезда между пунктами i и j. При $i = j$, $d_{ij} = M$. Элемент M символизирует бесконечность, запрещая переход «в себя».

Глава 2. Алгоритмы решения задачи коммивояжера

2.1. Метод полного перебора

Метод полного перебора, по-другому именуемый методом грубой силы (англ. brute force), является простым, логичным и широко используемым математическим методом [7,11]. Он применим во многих, если не во всех, областях математики: задача коммивояжера также не является исключением.

Идея brute force предельно проста: перебираются всевозможные решения и из них выбирается решение (или множество решений) отвечающее условию задачи.

В задаче коммивояжера, соответственно, требуется из всевозможных вариантов объезда пунктов выбрать маршрут, занимающий кратчайшее время (или минимальный по стоимости маршрут).

Огромным преимуществом метода полного перебора перед другими методами решения задачи коммивояжера является гарантированность нахождения наилучшего маршрута. Другие методы советуют лишь «хороший» маршрут, который совсем не обязательно является лучшим. Кроме того, к достоинствам метода относится простота его программной реализации.

Однако, в связи с наличием огромного недостатка, метод полного перебора крайне редко используется на практике. Этим недостатком является временная сложность алгоритма. Асимметричная задача коммивояжера с n посещаемых пунктов требует при полном переборе рассмотрения (n-1)! туров, а факториал, как можно увидеть из таблицы 2.1.1, растет невероятно быстро:

Таблица 2.1.1. Примерные значения факториала

Факториал числа	5!	10!	15!	20!	25!	30!	40!	50!
Значение	$\sim 10^{2}$	$\sim 10^{6}$	$\sim 10^{12}$	$\sim 10^{18}$	$\sim 10^{25}$	$\sim 10^{32}$	$\sim 10^{47}$	$\sim 10^{64}$

Поэтому метод полного перебора может применяться только для задач малой размерности (при рассмотрении до двух десятков посещаемых пунктов).

Для реализации метода полного перебора необходимо научиться производить генерацию всех перестановок заданного числа элементов. Сделать это можно несколькими способами, но самый распространенный (самый используемый в иных переборных алгоритмах) – это перебор в лексикографическом порядке.

Пусть имеется некоторый алфавит и наборы символов этого алфавита (букв) – слова. Буквы в алфавите упорядочены. Например, в русском алфавите порядок букв следующий: $а \rightarrow$ б $\rightarrow$ я (символ «→» используется здесь для обозначения предшествования). Если упорядочены буквы алфавита, то можно упорядочить и слова. Например, дано слово $u = (u_1, u_2, \dots, u_m)$, состоящее из букв $u_1, u_2, \dots, u_m$, и слово $v = (v_1, v_2, \dots, v_b)$. Тогда если $u_1 \rightarrow v_1$, то и $u \rightarrow v$. Если же $u_1 = v_1$, то сравнивают вторые буквы и т. д. Такой порядок слов и называется лексикографическим. Поэтому в словарях слово «тополь» стоит раньше слова «топор». Пробел считается символом, предшествующим любой букве алфавита, поэтому слово «пар» стоит раньше слова «парк».

Рассмотрим в качестве примера перебор перестановок из пяти элементов, обозначенных цифрами 1…5. Лексикографически первой перестановкой будет являться 1-2-3-4-5, второй – 1-2-3-5-4, …, последней – 5-4-3-2-1. Для непосредственной реализации генерации перестановок нужно определить общий алгоритм преобразования любой перестановки в следующую.

Алгоритм выглядит следующим образом. Пусть дана перестановка 1-3-5-4-2. Нужно двигаться по перестановке справа налево, пока впервые не будет обнаружено число, меньшее, чем предыдущее («3» после «5»). Предположим, что найденное число располагается на позиции P_{i-1}. Меняем найденное число местами с наименьшим из больших чисел, которые расположены правее позиции P_{i-1} (такое, очевидно, будет). После чего, числа правее позиции P_{i-1} необходимо упорядочить по возрастанию. В результате получается непосредственно следующая перестановка: в примере это

1-4-2-3-5, за ней следует 1-4-2-5-3 и т. д.

После генерации перестановки в основной программе считается сумма проезда между пунктами, результат записывается в файл. После того, как сгенерированы все перестановки и посчитаны суммы проезда для них, выбирается маршрут, соответствующий минимальному времени объезда пунктов.

2.2. Метод ближайшего соседа

Метод ближайшего соседа относится к так называемым жадным алгоритмам [3,19]. Жадный алгоритм (англ. Greedy algorithm) подразумевает под собой принятие локально оптимальных решений, допуская что конечное решение также окажется оптимальным. Существует довольно большое количество задач, для которых жадные алгоритмы дают оптимальные решения. К таким задачам относятся, например, задача о распределении заявок, задача о размене монет. Однако для ряда задач, относящихся к классу NP, жадные алгоритмы не дают оптимального решения. Однако в таких задачах, в том числе и в задаче коммивояжера, они дают весьма неплохое приближенное решение.

Идею алгоритма ближайшего соседа основана на простом эвристическом правиле: если мы будем посещать ближайший пункт на каждом шаге, то маршрут получится довольно хорош в целом. Перед коммивояжером ставится задача посещать ближайший из еще не посещенных пунктов. В алгоритме существуют два важных ограничения:

1. Недопущение повторного заезда в пункт. Оно связано с необходимостью (по условию задачи) нахождения гамильтонова цикла, то есть цикла, в котором все пункты посещаются единожды.

2. Недопущение возврата преждевременного возврата в исходный пункт. Этот запрет вводится для предотвращения преждевременного зацикливания маршрута, которое повлечет за собой неправильную работу алгоритма.

Схема алгоритма изображена на рис. 2.2.1.

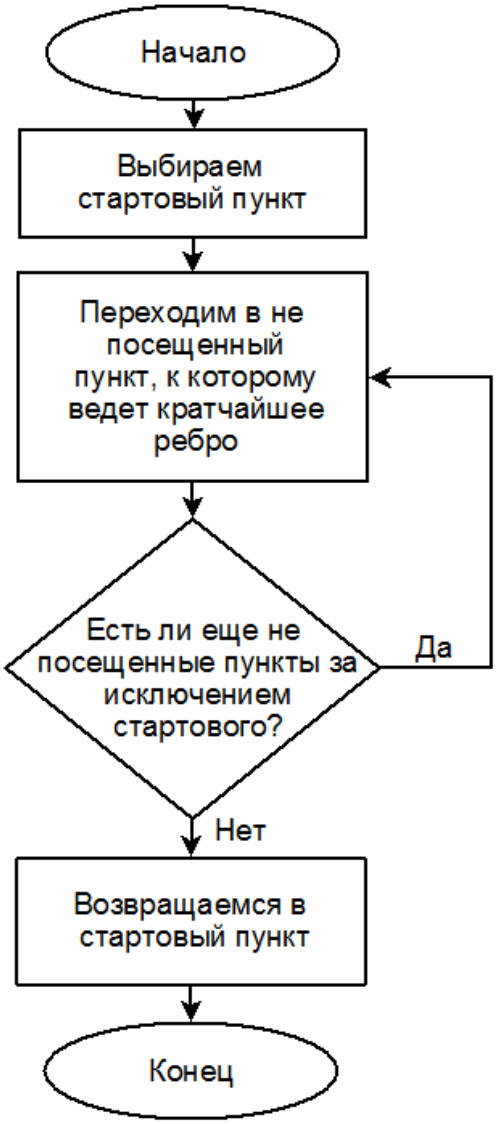

Рис. 2.2.1. Схема алгоритма ближайшего соседа

Для демонстрации работы метода приведена реальная матрица времен проезда между транспортными узлами г. Санкт-Петербург (см. таблицу 2.2.1).

Таблица 2.2.1. Матрица времен для демонстрации метода.

Наименование	№	0	1	2	3	4
Московский вкз.	0	M	12	21	19	9
м. Василеостровская	1	10	M	22	17	16
м. Автово	2	21	24	M	36	29
м. Старая Деревня	3	24	23	39	M	30
м. Ладожская	4	11	21	30	27	M

Значения времен проезда на автомобиле между парами пунктов были найдены с помощью сервиса https://2gis.ru/spb. Время, затрачиваемое на проезд между пунктами, дано в минутах.

На первом и последующих шагах курьеру необходимо выбирать ближайший из еще не посещенных пунктов. Также необходимо помнить, что

возврат в исходный пункт раньше времени невозможен.

Таблица 2.2.2. Шаг первый.

Наименование	№	0	1	2	3	4
Московский вкз.	0	M	12	21	19	9
м. Василеостровская	1	10	M	22	17	16
м. Автово	2	21	24	M	36	29
м. Старая Деревня	3	24	23	39	M	30
м. Ладожская	4	11	21	30	27	M

На первом шаге выбор производится без ограничений, так как повторное посещение пока не возможно, и возвращение в стартовый пункт даже теоретически не представляется реальным из-за нахождения в нём. То есть находясь в базовом пункте 10 (Московский вокзал) курьер выберет для посещения пункт 4 (ст. м. «Ладожская»), время проезда до которого (9 минут) минимально. Смотри таблицу 2.2.2.

Таблица 2.2.3. Шаг второй.

Наименование	№	0	1	2	3	4
Московский вкз.	0	M	12	21	19	9
м. Василеостровская	1	10	M	22	17	16
м. Автово	2	21	24	M	36	29
м. Старая Деревня	3	24	23	39	M	30
м. Ладожская	4	~~11~~	21	30	27	M

Оказавшись в 4-ом пункте для последующего посещения курьер выберет 1-ый пункт. Возможность возврата в «наиболее выгодный» стартовый пункт будет проигнорирована ввиду того, что нельзя вернуться на базу, не обслужив всех клиентов.

Таблица 2.2.4. Шаг третий.

Наименование	№	0	1	2	3	4
Московский вкз.	0	M	12	21	19	9
м. Василеостровская	1	~~10~~	M	22	17	~~16~~
м. Автово	2	21	24	M	36	29
м. Старая Деревня	3	24	23	39	M	30
м. Ладожская	4	11	21	30	27	M

Из 1-го пункта курьер последует в 3-й, игнорируя возврат на базу и повторное посещение 4-го пункта.

Таблица 2.2.5. Шаг четвертый.

Наименование	№	0	1	2	3	4
Московский вкз.	0	M	12	21	19	9
м. Василеостровская	1	10	M	22	17	17
м. Автово	2	21	24	M	36	29
м. Старая Деревня	3	~~24~~	~~23~~	39	M	~~30~~
м. Ладожская	4	11	21	30	27	M

На этом этапе так же, как и раньше будет проигнорирован преждевременный возврат в стартовый пункт и повторные посещения 1-го и 4-го пунктов. В результате чего курьер последует в единственный до сих пор не посещенный пункт 2.

Таблица 2.2.6. Шаг пятый.

Наименование	№	0	1	2	3	4
Московский вкз.	0	M	12	21	19	9
м. Василеостровская	1	10	M	22	17	17
м. Автово	2	21	~~24~~	M	~~36~~	~~29~~
м. Старая Деревня	3	24	23	39	M	30
м. Ладожская	4	11	21	30	27	M

Как только все клиенты обслужены, курьер может возвращаться в базовый пункт. Это и происходит на последнем этапе.

В итоге курьер проследовал по маршруту 0-4-1-3-2-0, затратив на объезд 9+21+17+39+21 = 107 минут.

Метод является очень быстрым ввиду чрезвычайно малого числа операций, требуемых для осуществления его работы. Однако результаты в сравнении с, например, методом ветвей и границ он показывает довольно скромные.

Время работы алгоритма для различного числа пунктов приведено в таблице 2.2.7.

Таблица 2.2.7. Время работы программной реализации метода ближайшего соседа.

Число пунктов	10	25	50	75	100	125	150
Время работы, с	Менее 0,1 с						

2.3. Усовершенствованный метод ближайшего соседа

Для того, чтобы метод давал более точные результаты, можно ввести в алгоритм небольшое усовершенствование, которое позволит выбирать решение из большего числа маршрутов для одной матрицы времен.

Возможность этого усовершенствования возможна в связи с тем, что объектом исследования является задача коммивояжера, решение которой подразумевает нахождение замкнутого цикла. В задачах, посвященных нахождению незамкнутого маршрута объезда пунктов такое усовершенствование алгоритма невозможно.

Идея заключается в том, чтобы запускать алгоритм, циклически изменяя начальный пункт (начальный для алгоритма, реальный же стартовый пункт остается прежним).

Для демонстрации работы метода была сгенерирована матрица 7-го порядка. Способ генерации будет описан в последующих главах.

Таблица 2.3.1. Матрица времен для демонстрации усовершенствованного метода.

№	0	1	2	3	4	5	6
0	M	75	64	69	60	63	29
1	66	M	35	59	96	80	74
2	58	30	M	26	110	111	57
3	64	53	21	M	14	15	86
4	51	94	109	8	M	45	95
5	57	70	106	6	35	M	76
6	22	71	55	76	92	67	M

Обычный алгоритм ближайшего соседа для данного экземпляра задачи предлагает следующий маршрут 0-6-2-3-4-5-0, объезд которого займет 305 минут.

Сформируем новую цепочку с условным стартовым пунктом 1 (вместо 0). Для этого воспользуемся методом ближайшего соседа с входными данными из таблицы 2.3.1, подразумевая первый пункт начальным. В результате получим цепочку 1-2-3-4-5-0-6-1, время объезда которой займет всего 277 минут. Легко убедится, что эта цепочка является решением для исходной задачи с реальным стартовым пунктом 0. В связи с замкнутостью маршрута, нам нужно лишь передвинуть стартовый пункт. Таким образом,

1-2-3-4-5-0-6-1 преобразуется в 0-6-1-2-3-4-5-0.

Те же действия повторяются и для остальных фиктивных стартовых пунктов (см. таблицу 2.3.2).

Таблица 2.3.2. Образование новых цепочек.

Новые цепочки	Время
1-2-3-4-5-0-6-1 → 0-6-1-2-3-4-5-0	277
2-3-4-5-0-6-1-2 → 0-6-1-2-3-4-5-0	277
3-4-5-0-6-2-1-3 → 0-6-2-1-3-4-5-0	289
4-3-5-0-6-2-1-4 → 0-6-2-1-4-3-5-0	290
5-3-4-0-6-2-1-5 → 0-6-2-1-5-3-4-0	265
6-0-4-3-5-1-2-6 → 0-4-3-5-1-2-6-0	267

Можно увидеть, что для приведенного экземпляра задачи маршрут, предложенный модифицированным методом, занимает на 40 минут меньше (относительный выигрыш во времени – 15 %). Усовершенствованный метод ближайшего соседа по результатам тестирования на различных матрицах размерностями от 10 до 150 дает результаты в среднем на 16% лучше обычного метода.

При малых размерностях (до 10-15 рассматриваемых в задаче пунктов) метод может не давать улучшения, ввиду малого количества рассматриваемых цепочек. При размерностях матрицы от 50 относительный выигрыш стабилизируется в окрестности 16%.

Для того, чтобы дополнительно увеличить вероятность получения более выгодного маршрута, можно рассматривать также обратные к цепочкам маршрутам (просто изменяя направление обхода цикла). В рамках данной работы это улучшение не производилось.

В связи с ничтожным временем работы исходного метода увеличение количества операций в 2n раз на итоговом времени работы программы не сказывается критически.

Время работы алгоритма для различного числа пунктов приведено в таблице 2.3.3.

Таблица 2.3.3. Время работы программной реализации усовершенствованного метода ближайшего соседа.

Число пунктов	10	25	50	75	100	125	150
Время работы, с	0,02	0,14	0,5	2,0	4,5	9	15

2.4. Метод ветвей и границ

Метод ветвей и границ (англ. branch and bound) — общий алгоритмический метод, широко применяющийся для решения задач комбинаторной оптимизации [8,25,26].

По сути, метод является усовершенствованным методом полного перебора с последовательным отсевом решений, кажущихся невыгодными. Вследствие того, что в процессе работы метода некоторые решения не рассматриваются, метод ветвей и границ не может гарантировать нахождения точного решения задачи. Отброшенное на начальном этапе «невыгодное» решение может оказаться в конце концов лучшим.

Метод впервые был предложен Лендом и Дойгом в 1960 году для решения задач целочисленного программирования [25]. В 1963 году группой авторов (Дж. Литтл, К. Мурти, Д. Суини, К. Кэрол) была предложена модификация метода ветвей и границ, разработанная специально для решения задачи коммивояжера [26]. Впоследствии этот алгоритм получил название «алгоритм Литтла».

В основе метода ветвей и границ лежит идея последовательного разбиения множества решений путем ветвления и нахождения оценок (границ). Схема метода приведена на рисунке 2.4.1.

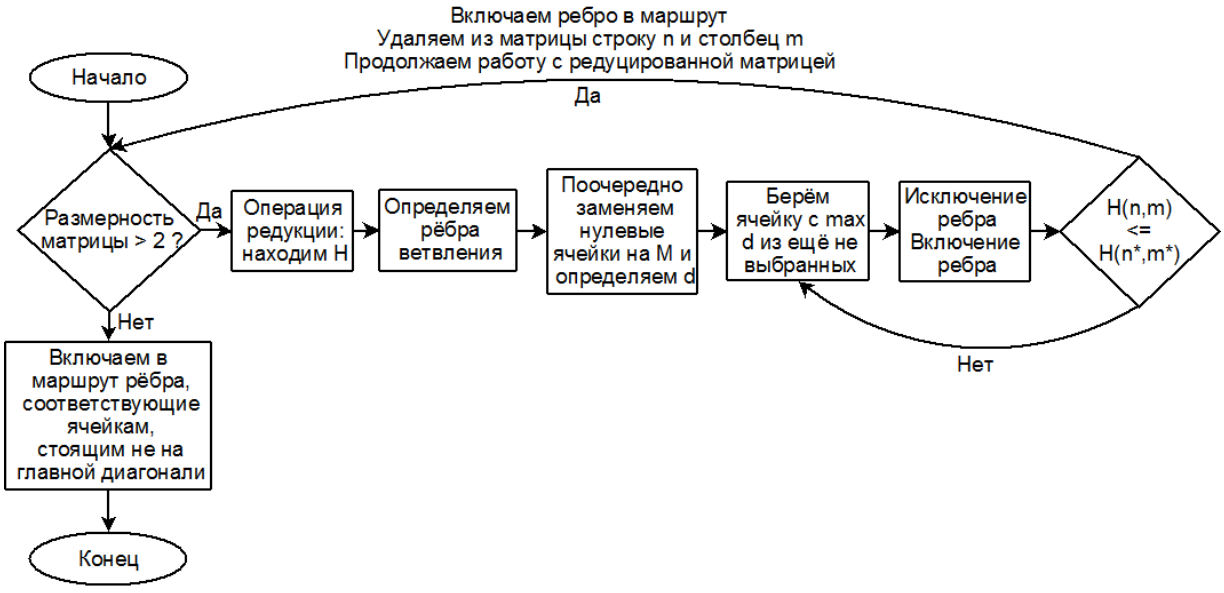

Рис. 2.4.1. Схема метода ветвей и границ.

Метод ветвей и границ считается универсальным методом, так как он хорошо подходит для решения антисимметричной задачи коммивояжера. В то время как другие методы приспособлены в основном для решения симметричных задач.

Алгоритм состоит из двух этапов:

Первый этап (предварительный):

Операция приведения матрицы и вычисление нижней оценки стоимости маршрута H.

1. Нахождение минимальных элементов в каждой строке (константы приведения d_i.

$$d_i = \min_i d_{ij}$$

Таблица 2.4.1. Шаг первый.

№	0	1	2	3	4	di
0	M	12	21	19	9	9
1	10	M	22	17	16	10
2	21	24	M	36	29	21
3	24	23	39	M	30	23
4	11	21	30	27	M	11

2. Из элементов строк матрицы вычитаются константы приведения d_i.

Таблица 2.4.2. Шаг второй.

№	0	1	2	3	4
0	M	3	12	10	0
1	0	M	12	7	6
2	0	3	M	15	8
3	1	0	16	M	7
4	0	10	19	16	M

3. Нахождение минимального элемента в каждом столбце (константы приведения d_j).

$$d_j = \min_j d_{ij}$$

Таблица 2.4.3. Шаг третий.

№	0	1	2	3	4
0	M	3	12	10	0
1	0	M	12	7	6
2	0	3	M	15	8
3	1	0	16	M	7
4	0	10	19	16	M
dj	0	0	12	7	0

4. Из элементов столбцов матрицы вычитаются константы приведения d_j.

Таблица 2.4.4. Шаг четвертый.

№	0	1	2	3	4
0	M	3	0	3	0
1	0	M	0	0	6
2	0	3	M	8	8
3	1	0	4	M	7
4	0	10	7	9	M

В результате получаем приведенную матрицу, в которой в каждой строке и в каждом столбце имеется хотя бы один нулевой элемент.

5. Вычисление H (нижней границы) как сумму констант приведения d_i и d_j.

$$H = \sum d_i + \sum d_j$$

$$H = 9 + 10 + 21 + 23 + 11 + 0 + \ 0 + 12 + 7 + 0 = 93$$

Второй этап (основной):

1. Вычисление штрафа за неиспользование для каждого нулевого элемента приведенной матрицы затрат.

Штраф за неиспользование элемента с индексом (i, j) означает, что соответствующее ребро не будет включено в маршрут, а значит минимальная стоимость неиспользования этого ребра равна сумме минимальных элементов в строке i и столбце j.

а) Ищутся все нулевые элементы в приведенной матрице

Таблица 2.4.5. Шаг пятый.

№	0	1	2	3	4
0	M	3	0	3	0
1	0	M	0	0	6
2	0	3	M	8	8
3	1	0	4	M	7
4	0	10	7	9	M

б) Нулевые элементы поочередно заменяются на M (бесконечность). Каждому нулевому элементу сопоставляется штраф (сумма констант приведения) за его неиспользование.

Таблица 2.4.6. Шаг шестой.

№	0	1	2	3	4	di
0	M	3	0 (0)	3	0 (6)	0
1	0 (0)	M	0 (0)	0 (3)	6	0
2	0 (3)	3	M	8	8	3
3	1	0 (4)	4	M	7	1
4	0 (7)	10	7	9	M	7
dj	0	3	0	3	6	

в) Выбирается элемент (из еще не выбранных), которому соответствует максимальный штраф.

Выбор элемента с максимальным штрафом обусловлен тем, что исключение из маршрута этого ребра приведет к максимальному увеличению стоимости оптимального маршрута, нахождение которого является нашей целью.

Наибольшая сумма констант приведения равна (7 + 0) = 7 для ребра $(4,0)$, следовательно, выбираем этот элемент.

2. Теперь исходное множество разбивается на два подмножества — $(i*, j*)$ (содержащее ребро (i, j)) и (i, j) (не содержащее ребро (i, j)). Проводится вычисление нижних границ для подмножеств.

В нашем случае множество разбивается на подмножества (4,0) и (4*,0*).

а) Исключение ребра $(i*, j*)$

Нижняя граница для подмножества $(i*, j*)$ равна сумме нижней границы H исходного множества и штрафа за неиспользование ребра (i, j).

Исключение ребра $(4,0)$ производится путем замены элемента $d_{40} = 0$ на M, после чего осуществляется очередное приведение матрицы расстояний для образовавшегося подмножества (4*,0*), в результате получим редуцированную матрицу.

Таблица 2.4.7. Шаг седьмой.

№	0	1	2	3	4	di
0	M	3	0	3	0	0
1	0	M	0	0	6	0
2	0	3	M	8	8	0
3	1	0	4	M	7	0
4	M	10	7	9	M	7
dj	0	0	0	0	0	

Нижняя граница гамильтоновых циклов этого подмножества:

$$H(4*,0*) = 93 + 7 = 100$$

б) Включение ребра (i, j)

При вычислении нижней границы для множества (i, j) необходимо принять во внимание, что раз ребро (i, j) входит в маршрут, то симметричное ребро (j, i) в маршрут входить не может, поэтому в матрице этот элемент заменяется на M.

А в связи с тем, что из пункта i мы «уже ушли», а в пункт j мы «уже пришли», то ни одно ребро, выходящее из i, и ни одно ребро, приходящее в j, уже использоваться не могут, поэтому вычеркиваем из матрицы строку i и столбец j.

Проводим операцию приведения для получившейся матрицы. Нижняя граница для (i, j) будет равна нижней границе H исходного множества плюс сумма констант приведения для получившейся редуцированной матрицы.

Включение ребра $(4,0)$ проводится путем исключения всех элементов 4-й строки и 0-го столбца, в которой элемент d_{04} заменяется на M, для исключения образования негамильтонова цикла.

В результате получается другая сокращенная матрица (4×4), которая подлежит операции приведения.

После операции приведения сокращенная матрица будет иметь вид:

Таблица 2.4.8. Шаг восьмой.

№	1	2	3	4	di
0	3	0	3	M	0
1	M	0	0	6	0
2	3	M	8	8	3
3	0	4	M	7	0
dj	0	0	0	6	

Сумма констант приведения сокращенной матрицы:

$$\sum d_i + \sum d_j = 9$$

3. Если $(i, j) \leq (i *, j *)$, то ребро включается в маршрут, и дальнейшая работа метода происходит с матрицей меньшей размерности. В противном случае происходит возврат к пункту 1в, и рассматривается элемент с со следующим по убыванию штрафом за неиспользование.

Нижняя граница подмножества $(4,0)$ равна:

$$H(4,0) = 93 + 9 = 102 > 100$$

Поскольку нижняя граница этого подмножества $(4,0)$ больше, чем подмножества $(4 *, 0 *)$, то ребро $(4,0)$ на данный момент в маршрут не включается. Происходит откат к пункту 1в, выбирается элемент со следующим по убыванию штрафом.

Процесс ветвления путем включения и исключения ребер продолжается до тех пор, пока в рассмотрении не окажется матрица размерности 2. Из нее добавление ребер в маршрут происходит тривиальным образом.

Замечание. При решении задачи коммивояжера методом ветвей и границ необходимо перед каждой итерацией алгоритма запрещать для посещения ребра, которые могут привести к преждевременному зацикливанию алгоритма. Делается это путем присвоения соответствующему

элементу матрицы значения M (бесконечность).

В рассматриваемом случае алгоритмом в конце работы был предложен следующий маршрут: 0-4-3-1-2-0. Время объезда маршрута, предложенного методом ветвей и границ, займет 102 минуты.

Среднее время работы программной реализации алгоритма для различного числа пунктов представлено в таблице.

Таблица 2.4.9. Время работы программной реализации метода ветвей и границ.

Число пунктов	10	25	50	75	100	125	150
Время работы, с	0,03	0,2	3,4	35	85	202	436

Глава 3. Моделирование загруженности транспортной сети

3.1. Дорожные заторы (пробки)

Задача коммивояжера несомненно является очень интересным объектом изучения с точки зрения математики. Но не стоит забывать, что она занимает немаловажное место и в семействе прикладных задач. Однако, статическая задача коммивояжера мало интересна для практики ввиду недостаточного соответствия реальности, где время проезда между парой пунктов не постоянно [2,4].

Для того чтобы учесть эту составляющую задачи было принято решение реализовать модель динамического изменения дорожной ситуации в городе [14,15]. Для учета этих влияний, символизирующих дорожные заторы (пробки), было решено воспользоваться балльной системой.

Систему баллов, демонстрирующую влияние пробок на дорожную ситуацию в городе, было решено разработать в соответствии с аналогичной системой сервиса «Яндекс.Пробки», предлагающей следующее ранжирование:

1 — Дороги свободны
2 — Дороги почти свободны
3 — Местами затруднения
4 — Местами затруднения
5 — Движение плотное
6 — Движение затрудненное
7 — Серьезные пробки
8 — Многокилометровые пробки
9 — Город стоит
10 — Пешком быстрее

Необходимо отметить, что приведенные выше баллы соответствуют лишь общей (средней) загруженности дорог в городе, то есть они не могут служить эквивалентами констант, на которые будут «домножены» конкретные значения времени перемещения между парой пунктов в матрице времен.

Для того, чтобы модель максимально отражала реальное влияние пробок на изменение времени перемещения между пунктами, был проведен

небольшой социологический опрос автолюбителей Санкт-Петербурга. По результатам которого были сформированы диапазоны изменения времени перемещения между пунктами для пробок, эквивалентных конкретному значению (баллу).

В таблице приведены эти диапазоны.

Таблица 3.1.1. Диапазоны домножения входных данных.

Балл	1	2	3	4	5	6	7	8	9	10
min	1.0	1.0	1.0	1.0	1.2	1.4	1.6	1.8	2.0	2.5
max	1.0	1.1	1.3	1.6	3.0	5.0	7.0	9.0	10.0	12.0

То есть генерация влияния пробок на идеальную дорожную ситуацию происходит следующим образом:

1. Функция probki вызывается с определенным значением от 1 до 10.
2. Увеличение времени проезда по конкретному маршруту эквивалентно значению из диапазона, соответствующего числовому эквиваленту дорожного затора. Значение выбирается из диапазона случайным образом для каждого из маршрутов.

Ниже приведен фрагмент этой функции:

```
case 3: // если c = 3
{
f << "\n" << "3 – Местами затруднения" << endl << "\n" ;
      for (int i = 1; i < str; i++)
      for (int j = 1; j < stl; j++)
          if (i != j)
          {
               probka = random (1.0, 1.3);
               out.mas[i][j] = out.mas[i][j] * probka;
          }
break;
}
```

Предложенный способ был успешно реализован на языке C++ для метода ветвей и границ. Примеры работы предложенного способа представлены в таблицах 3.1.2 – 3.1.4.

Таблица 3.1.2. Матрица времен для демонстрации метода. Результат генерации 1-балльной пробки.

№	1	2	3	4	5	6
1	M	12	21	19	9	20
2	10	M	22	17	17	27
3	21	24	M	36	29	19
4	24	23	39	M	30	41
5	11	21	30	27	M	29
6	17	28	18	36	26	M

Таблица 3.1.3. Результат генерации 5-балльной пробки.

№	1	2	3	4	5	6
1	M	31,2	54,6	55,1	27	38
2	12	M	48,4	37,4	42,5	51,3
3	44,1	31,2	M	90	63,8	22,8
4	36	55,2	39	M	70,2	57,4
5	24,2	25,2	42	32,4	M	52,2
6	42,5	50,4	25,2	43,2	46,8	M

Таблица 3.1.3. Результат генерации 8-балльной пробки.

№	1	2	3	4	5	6
1	M	86,4	172,2	49,4	50,4	50
2	22	M	63,8	120,7	83,3	59,4
3	62	52,8	M	104,4	243,6	121,6
4	79,2	131,1	292,5	M	66	303,4
5	341	159,6	126	218,7	M	162,4
6	78,2	176,4	100,8	226,8	189,8	M

В первом случае программой был предложен маршрут 1-5-4-2-3-6-1, объезд которого занял 117 минут. В случае 5-балльной пробки программой был предложен маршрут 1-5-4-3-6-2-1, объезд которого занял 183,6 минуты. В случае 8 баллов программа предложила маршрут 1-4-5-3-2-6-1, его объезд занял 431,8 минуты.

Можно увидеть, что помимо серьезного изменения времен объезда пунктов, пробки ввиду своего относительно случайного характера влияют и на сами маршруты. Предложенный способ адекватно моделирует загруженность транспортной сети.

Предполагается, что пользователь (курьер) перед выездом на задание, запускает программу. Перед запуском алгоритмической части программы

вызывается функция probki, моделирующая дорожную ситуацию. С учетом влияния этой функции основная программа предлагает маршрут следования. В каждом новом пункте курьер повторно запускает программу, размерность входных данных в данном случае будет меньше на единицу. Функция генерирует новую дорожную ситуацию, в соответствии с которой предлагает измененный маршрут следования.

Автором была предложена модель генерирования измененной дорожной ситуации. Для использования метода на практике необходимо реализовать функцию сбора данных об изменении дорожной ситуации со спутниковых систем.

Глава 4. Сравнительный анализ эвристических методов решения задачи коммивояжера

4.1. Разработка скрипта для параллельных генерации и вычислений алгоритмов

Одной из главных целей данной выпускной квалификационной работы является сравнение и анализ целесообразности применения алгоритмов решения задачи коммивояжера для конкретных постановок задачи. Для того чтобы делать такие выводы, необходимо располагать широкой выборкой экземпляров задач.

Однако генерация и последующая работа алгоритма даже на одном экземпляре задачи, например, для метода ветвей и границ может занимать длительное время. В этой связи возникает необходимость ускорить процесс генерации выборки. Параллельное вычисление различных экземпляров задач позволяет в разы сократить время на создание выборки.

В рамках учебной практики, проводимой на базе Учебно-научной лаборатории высокопроизводительных вычислении факультета ПМ-ПУ, автору была поставлена задача разработать скрипт, который бы позволил одновременно производить генерацию нескольких экземпляров задачи и выполнение алгоритма для них.

Поставленная задача была выполнена. Был успешно разработан скрипт (см. приложение), с помощью которого может осуществляться генерация и последующая работа метода ветвей и границ на нескольких экземплярах задачи в фоновом режиме. Сравнение времен выполнения последовательного алгоритма и алгоритма, запускаемого в фоновом режиме приведено в таблице 4.1.1.

Таблица 4.1.1. Сравнение времени работы программы на кластере с использованием скрипта и суммарного времени работы программы для 8 экземпляров задачи на 8 потоках.

Число пунктов	Скрипт, время работы	Последовательно, время работы
100	53 с	7 мин 57 с
125	6 мин 4 с	29 мин 58 с
150	15 мин 47 с	1 ч 17 мин
200	29 мин 7 с	3 ч 50 мин

Как видно из таблицы, использование кластера для параллельных вычислений позволило выполнять необходимые действия в среднем в 5-10 раз быстрее.

В случае доступности большего числа нод кластера, можно дополнительно оптимизировать выполнение задачи за счет параллельного вычисления большего числа экземпляров задачи.

4.2. Способ генерации матриц времен для задачи коммивояжера в условиях городского цикла

Для удобства реализации и для чистоты проводимых в рамках сравнительного анализа численных экспериментов разработана небольшая функция, генерирующая матрицы времен специально для асимметричной задачи коммивояжера.

При разработке были проанализированы матрицы времен, соответствующие реальным городским пунктам (например, смотри таблицу 2.2.1.). Значения времен проезда соответствуют городским реалиям, времена проезда между симметричными пунктами генерируются в зависимости друг от друга. Листинг функции приведен ниже.

```
void matrix::vvod_rand()
{
     srand( time( 0 ) );

     for (int i = 0; i < str; i++) mas[i][i] = 0;

     for (int i = 0; i < str; i++)
          for (int j = 0; j < stl; j++)
               if ((i != j) && (mas[i][j] == 0))
               {
                    mas[i][j] = 11 + rand() % 101;
                    mas[j][i] = mas[i][j] - 10 + rand() % 10;
               }
}
```

Матрица из таблицы 2.3.1. в числе прочих была сгенерирована описанным выше способом.

4.3. Сравнительный анализ эвристических методов решения задачи коммивояжера

Для качественного сравнения алгоритмов в соответствии со способом, приведенным выше, было сгенерировано по 5 матриц размерностей 10, 25, 50, 75, 100, 125 и 150. Времена работы программных реализаций алгоритмов приведены в таблице 4.3.1. без учета времени на генерацию матриц.

По итогам работы программных реализаций в файл записывалось время работы программы и суммарное время объезда, предложенного программой оптимального маршрута. Сам оптимальный маршрут в файл не записывался в связи с отсутствием наглядности в представлении маршрутов для задач большой размерности (возможность записи маршрута в файл в программных реализациях методов имеется).

Времена и результаты работы каждого алгоритма для матриц каждой размерности были усреднены. На основании результатов были сформированы три таблицы, каждой из которой соответствует график.

Сравнение времен работы. В таблице 4.3.1. сравниваются времена работы программных реализаций алгоритмов и демонстрируется зависимость времени работы алгоритмов от размерности входных данных.

Таблица 4.3.1. Сравнение времен работы программных реализаций алгоритмов.

Число пунктов	Время работы программной реализации, с		
	Программная реализация метода ветвей и границ	Программная реализация метода ближайшего соседа	Программная реализация усовершенствованного метода ближайшего соседа
10	0,03	0,01	0,02
25	0,24	0,04	0,14
50	3,44	0,03	0,55
75	35,47	0,05	2,00
100	84,91	0,03	4,57
125	201,62	0,09	8,71
150	436,28	0,10	15,12

Из таблицы видно, что при числе пунктов более 75 метод ветвей и границ может работать несколько минут. Однако, стоит отметить, что, к примеру, пятиминутное время ожидания результата программы приемлемо для пользователя, которому требуется объехать порядка ста пунктов. Время работы метода ближайшего соседа составляет менее 0,1 с для задач приведенных размерностей, с дальнейшим увеличением числа пунктов, существенного увеличения времени работы не произойдет. Усовершенствованный метод ближайшего соседа работает приемлемое время, однако при дальнейшем увеличении числа пунктов время работы алгоритма будет расти существеннее.

В соответствии с таблицей 4.3.1 был построен график (см. рис. 4.3.1.)

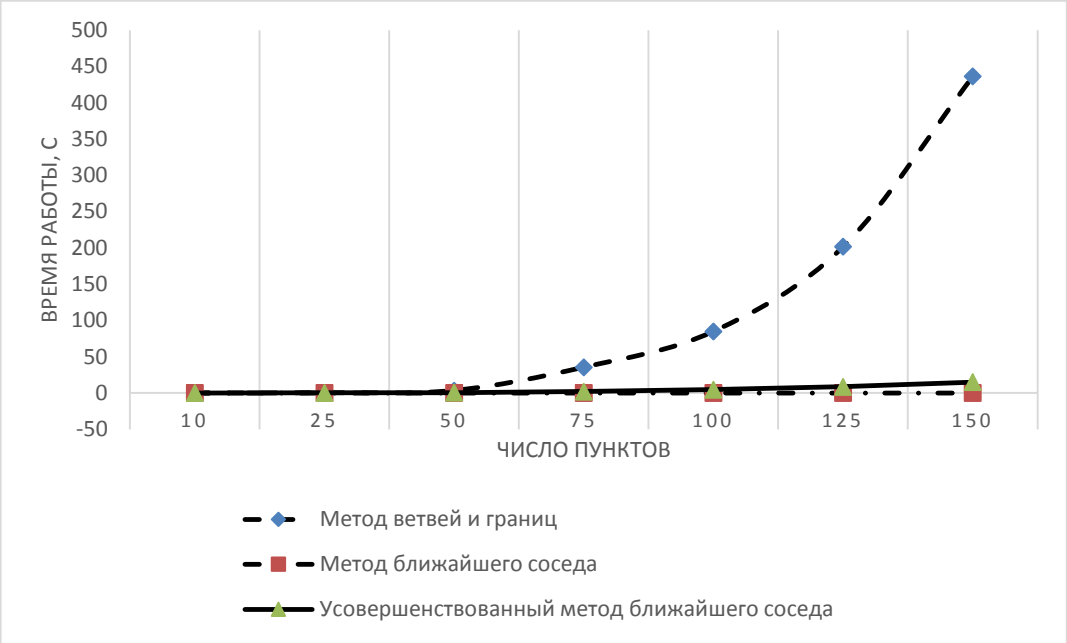

Рис. 4.3.1. График зависимости времени работы методов от числа пунктов.

Из графика видно, что зависимость времени работы метода ветвей и границ от числа пунктов близка к экспоненциальной. Зависимость времен работы методов ближайшего соседа близка к линейной, однако, в случае усовершенствованного метода возрастание времени работы при существенном увеличении числа пунктов будет расти полиномиально.

Сравнение результатов работы. В следующей таблице результаты работы программных реализаций обычного и усовершенствованного методов ближайшего соседа сравниваются с результатами работы программной реализации метода ветвей и границ. В таблице приводится относительный проигрыш во времени объезда методов ближайшего соседа методу ветвей и границ в процентом отношении.

Таблица 4.3.2. Относительный проигрыш результатов работы программных реализаций исходного и усовершенствованного методов ближайшего соседа программной реализации метода ветвей и границ.

Число пунктов	Проигрыш, %	
	Программная реализация метода ближайшего соседа	Программная реализация усовершенствованного метода ближайшего соседа
10	31,08	9,98
25	29,77	9,98
50	44,13	23,07
75	39,01	21,97
100	37,86	23,16
125	43,17	23,62
150	45,83	26,56
Среднее	38,69	19,76

По данным таблицы 4.3.2. был построен график (см. рис. 4.3.2.)

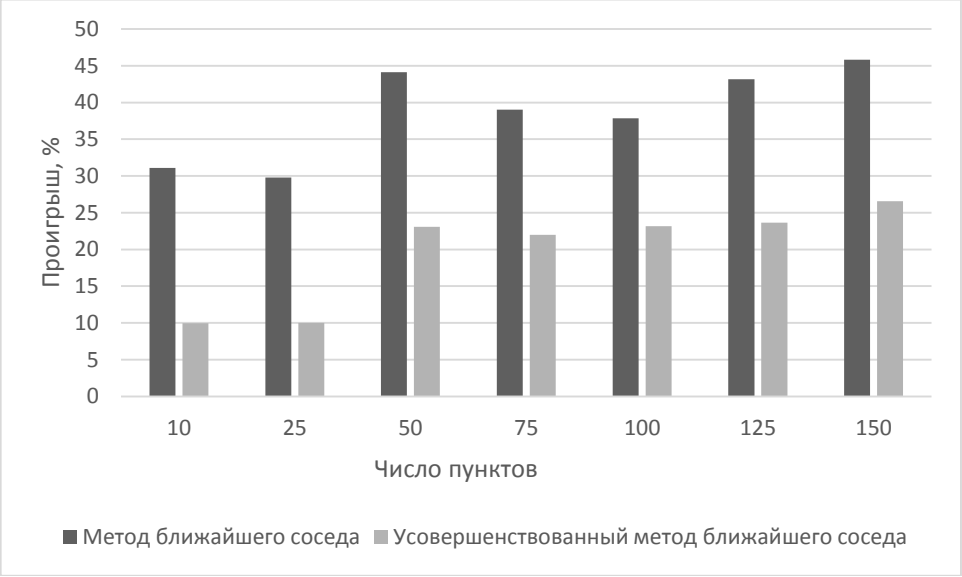

Рис. 4.3.2. График зависимости относительных проигрышей результатов работы программных реализаций исходного и усовершенствованного методов ближайшего соседа программной реализации метода ветвей и границ от числа рассматриваемых в задаче пунктов.

Из приведенных таблицы и графика видно, что оба метода ближайшего достаточно существенно проигрывают методу ветвей и границ. Однако проигрыш усовершенствованного метода не так существенен. Интересно, что для одного из экземпляров задач (при 25 посещаемых пунктах) усовершенствованный метод ближайшего соседа предложил более выгодный маршрут, чем метод ветвей и границ. Также стоит отметить тенденцию к увеличению относительного проигрыша методов ближайшего соседа с увеличением числа рассматриваемых в задаче пунктов.

В следующей таблице сравниваются усовершенствованный и исходный методы ближайшего соседа. Результаты представлены в виде проигрыша по времени объезда в процентном отношении обычного метода усовершенствованному.

Таблица 4.3.3. Относительный проигрыш результатов работы программной реализации исходного метода ближайшего соседа программной реализации усовершенствованного метода.

Число пунктов	Проигрыш, %
10	19,93
25	17,96
50	16,93
75	14,30
100	12,04
125	15,81
150	15,17
Среднее	16,00

В соответствии с таблицей 4.3.3 был построен график (см. рис. 4.3.3.).

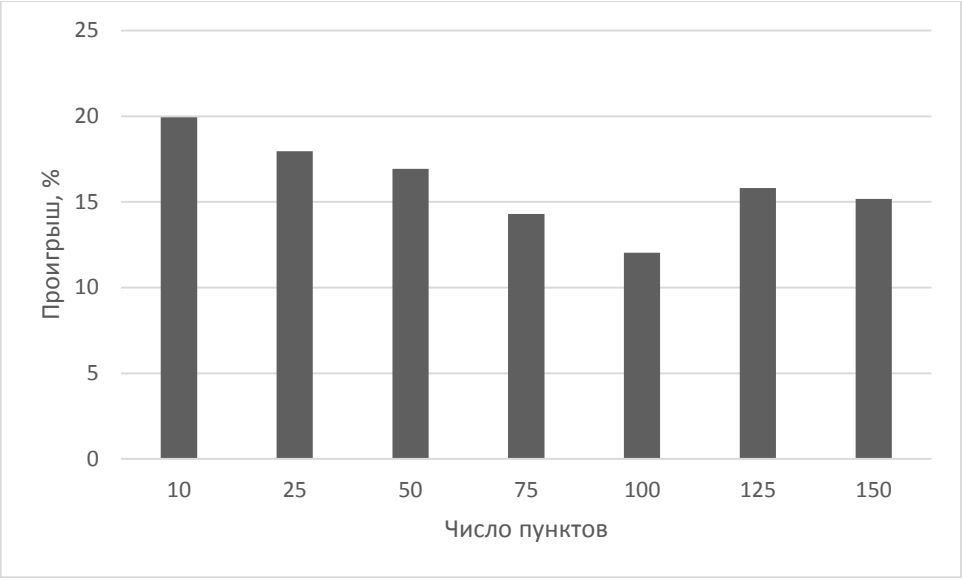

Рис. 4.3.3. График зависимости относительного проигрыша результатов работы программной реализации исходного метода ближайшего соседа программной реализации усовершенствованного метода от числа рассматриваемых в задаче пунктов.

Из приведенных таблицы и графика видно, что усовершенствованный метод ближайшего соседа дает более точные результаты, чем исходный метод. Также стоит отметить, что выигрыш усовершенствованного метода фактически не зависит от размерности задачи и составляет порядка 16 процентов.

Все программные реализации тестировались на переносном персональном компьютере Samsung RV511 со следующими техническими характеристиками:

Таблица 4.3.4. Технические характеристики переносного ПК.

Процессор	Intel Core i5 480M
Тактовая частота	2.66 ГГц
Количество ядер	2
Объем оперативной памяти	4 Гб
Тип оперативной памяти	DDR3
Частота памяти	1066 МГц

Выводы

Успешно разработан способ моделирования загруженности транспортной сети, который впоследствии может применяться и для программных реализаций иных транспортных задач, подразумевающих изменение дорожной ситуации.

Метод ветвей и границ подтвердил, что он является одним из самых успешных методов для решения асимметричной задачи коммивояжера. Тот факт, что метод работает несколько дольше своих соперников на практике существенно не скажется, так как время работы программы при большом числе пунктов ничтожно мало по сравнению с выигрышем во времени объезда, которое он дает.

Программная реализация метода ближайшего соседа, как и предполагалось, работает быстро, однако маршруты, предлагаемые методом, могут быть полезны только при чрезвычайно малых размерностях входных данных. Только в этих случаях маршрут, предлагаемый методом ближайшего соседа, проигрывает адекватное в абсолютном значении время. Также алгоритм метода ближайшего соседа может быть полезен для курьера в случае отсутствия программного обеспечения. Связано это с простотой построения маршрута этим методом и его удовлетворительных результатов.

Из результатов работы программной реализации усовершенствованного метода ближайшего соседа видно, что модификация действительно сделала метод более эффективным. Усовершенствованный метод дает примерно на 16% лучшие результаты и на некоторых экземплярах задачи может составить конкуренцию методу ветвей и границ. Время работы программной реализации метода делает его удобным в использовании мобильными пользователями.

Продемонстрировано, что эвристические методы позволяют успешно решать задачи любых размерностей.

Заключение

Поставленные в данной работе цели и задачи были успешно выполнены:

- изучена литература, посвященная задаче, отмечены основные направления современных исследований в данной области;
- изучен метод полного перебора; показано, что он не может успешно применяться на практике ввиду быстрого роста времени работы его программной реализации при увеличении количества посещаемых пунктов;
- подробно описаны метод ближайшего соседа и метод ветвей и границ, проиллюстрированы шаги работы алгоритмов, приведены их логические схемы, оба метода успешно реализованы на языке C++;
- на основе широкой выборки сгенерированных матриц построены сравнительные таблицы и графики, на основании которых были сделаны выводы о полезности применения алгоритмов на практике;
- произведено усовершенствование метода ближайшего соседа путем введения в рассмотрение фиктивных стартовых пунктов; на основании применения усовершенствованного метода к различным экземплярам задачи, сделан вывод о том, что эффективность метода повысилась;
- разработана модель учета изменения дорожной ситуации, проведена демонстрация работы модели на примере;
- разработан удобный способ генерации входных данных, обоснована его адекватность специфике рассматриваемой постановки задачи; разработан скрипт, позволяющий генерировать входные данные и осуществлять программное решение задачи одновременно для нескольких экземпляров задачи.

Результаты проделанной работы докладывались и публиковались на конференциях с международным участием, проводимых Санкт-

Петербургским государственным университетом и Санкт-Петербургским политехническим университетом Петра Великого, а так же на международной научно-исследовательской конференции «INFO-эксперт»-2016. Статья «Один из способов моделирования дорожной ситуации для решения задачи коммивояжера в условиях города» была отмечена дипломом победителя II-ой степени в номинации «Computer science – IT-эксперт» на международной научно-исследовательской конференции «INFO-эксперт»-2016.

Список литературы

1. Большакова Е. И., Мальковский М. Г., Пильщиков В. Н. Искусственный интеллект. Алгоритмы эвристического поиска [Электронный ресурс] // Учебная литература факультета ВМК МГУ. URL: http://recyclebin.ru/BMK/II/ii.html (дата обращения: 08.10.2015).
2. Бронштейн Е. М., Заико Т. А. Детерменированные оптимизационные задачи транспортной логистики // Автоматика и телемеханика, 2010. №10. С. 133-147.
3. Вишняков П.О. Планирование маршрутов с использованием модифицированного метода «ближайшего соседа» //Математические методы в технике и технологиях - ММТТ. 2014. № 6 (65). С. 63-67.
4. Гладков Л.А., Баринов С.В., Разработка новых подходов к решению транспортной задачи с использованием геоинформационных технологий // Известия ТРТУ. Тематический выпуск «Интеллектуальные САПР», 2009. №2. С. 141-144.
5. Гладков Л.А., Гладкова Н.В. Особенности использования нечетких генетических алгоритмов для решения задач оптимизации и управления // Известия ЮФУ. Технические науки. 2009. № 4 (93). С. 130-136.
6. Гончарова А.Б., Поборчий И.В. Исследование методов решения задачи коммивояжера при управлении транспортными потоками предприятия // Процессы глобальной экономики: Сборник научных трудов XX Всероссийской научно-практической конференции с международным участием. СПб.: Издательство Политехнического университета, 2015. С. 318-324.
7. Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Р., Штайн К. Алгоритмы. Построение и анализ. 2 изд. М.: Вильямс, 2012. 1296 с.
8. Костюк Ю. Л. Эффективная реализация алгоритма решения задачи коммивояжера методом ветвей и границ // Прикладная дискретная математика. Вычислительные методы в дискретной математике, 2010.

№2 (20). С. 78-90.

9. Курейчик В.В., Курейчик В.М. Генетический алгоритм определения пути коммивояжера // Известия Российской академии наук. Теория и системы управления. 2006. № 1. С. 94-100.
10. Курейчик В.М., Кажаров А.А. Муравьиные алгоритмы для решения транспортных задач. // Известия РАН. Теория и системы управления. – 2010. № 1. С. 32-45.
11. Левитин А. В. Алгоритмы: введение в разработку и анализ. М.: Вильямс, 2006. 576 с.
12. Маций О.Б. Повышение точности симметричной задачи класса коммивояжера большой размерности // Вестник Харьковского национального автомобильно-дорожного университета. 2011. № 55. С. 100-102.
13. Мудров В. И. Задача о коммивояжёре. М.: Знание, 1969. 62 с.
14. Пекарская С.С. Расчёт весовых коэффициентов для нахождения кратчайшего по времени пути // Сборник докладов Всероссийской научно-практической конференции «Современные техника и технологии», Томск, 2012.
15. Пекарская С.С. Построение рационального маршрута при решении задач транспортной логистики с учетом нагрузки в дорожной сети // Сборник «Перспективы развития информационных технологий» Труды Всероссийской молодежной научно-практической конференции. Кузбасский государственный технический университет имени Т.Ф. Горбачева, Международный научно-образовательный центр КузГТУ-Arena Multimedia. 2014. С. 382-383.
16. Седжвик Р. Фундаментальные алгоритмы на C++. Части 1-4. Анализ. Структуры данных. Сортировка. Поиск = Algorithms in C++, СПб: ДиаСофт, 2002. 688 с.
17. Ураков А.Р., Михтанюк А.А. Оценка количества вариантов обхода в задаче коммивояжера с дополнительными условиями //Глобальный

научный потенциал, 2012. № 21. С. 82-86.

18. Хухрянская Е.С., Юдина Н.Ю., Ющенко Е.В. Анализ возможностей применения методов теории расписаний к задачам деревообрабатывающих производств и их формализация // Лесотехнический журнал. 2011. № 3. С. 37-40.
19. Bang-Jensen J., Gutin G., Yeo A. When the greedy algorithm fails // Discrete Optimization. 2004 Vol. 1, No 2. P. 121-127.
20. Berbeglia G., Cordeau J.F., Gribkovskaia I., Laporte G. Static pickup and delivery problems: A classification scheme and survey // TOP. 2007. V. 15. No 1. P. 1-31.
21. M. Dorigo, V. Maniezzo & A. Colorni. Ant System: optimization by a colony of cooperating agents // IEEE transactions on systems, man, and cybernetics-part B. 1996 No. 26 (1). P. 29-41.
22. Gutin G., Jakubowicz H., Ronen, Sh., Zverovitch A. Seismic vessel problem, Communications in DQM, 2005 No. 8. P. 13-20.
23. Hahsler M., Hornik K. TSP – Infrastructure for the traveling salesperson problem // Journal of statistical software, 2007 No. 23 (2). P. 1-21.
24. Kellerer H., Pferschy U., Pisinger D. Knapsack problems. // Springer-Verlag, 2003. 548 c.
25. Land A. H., Doig A. G. An automatic method of solving discrete programming problems // Econometrica. 1960 Vol. 28. P. 497-520.
26. Little J. D. C., Murty K. G., Sweeney D. W., Karel C. An algorithm for the traveling salesman problem // Operations Research. 1963 Vol. 11, No 6. P. 972-989.
27. Paar C., Pelzl J. Understanding cryptography: a textbook for students and practitioners. Berlin: Springer, 2010. 372 c.
28. Parragh S., Doerner K., Hartl R. A survey on pickup and delivery problems. Part II: Transportations between customers and depot // J. Betriebswirtschaft. 2008. V. 58. No 2. P. 81-117.

Приложение

Скрипт для параллельного запуска решения экземпляров задач в фоновом режиме.

Основной скрипт (script.sh):

```
#!/bin/bash
#PBS -N MViG_1
#PBS -q middle
#PBS -l nodes=1:ppn=8

hostname
export LANG=C
cd /tmp

mkdir 1
mkdir 2
mkdir 3
mkdir 4
mkdir 5
mkdir 6
mkdir 7
mkdir 8

cd 1
bash /home/poborchiy/webws/ws/MViG/mvig_prog.sh &
cd ../2
sleep 1
bash /home/poborchiy/webws/ws/MViG/mvig_prog.sh &
cd ../3
sleep 1
bash /home/poborchiy/webws/ws/MViG/mvig_prog.sh &
cd ../4
sleep 1
bash /home/poborchiy/webws/ws/MViG/mvig_prog.sh &
```

```
cd ../5
sleep 1
bash /home/poborchiy/webws/ws/MViG/mvig_prog.sh &
cd ../6
sleep 1
bash /home/poborchiy/webws/ws/MViG/mvig_prog.sh &
cd ../7
sleep 1
bash /home/poborchiy/webws/ws/MViG/mvig_prog.sh &
cd ../8
sleep 1
bash /home/poborchiy/webws/ws/MViG/mvig_prog.sh &

wait;
```

Вызываемый скрипт (mvig_prog.sh):

```
!/bin/bash

time /home/poborchiy/webws/ws/MViG/MViG
cat metod.txt >> /home/poborchiy/webws/ws/MViG/sum_metod_100.txt

rm *
```