ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНО ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
“ВЫСШАЯ ШКОЛА ЭКОНОМИКИ”

Факультет компьютерных наук

Департамент программной инженерии

УДК 004.021

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
“Программная инженерия”,
профессор департамента программной
инженерии, канд. техн. наук
В.В. Шилов

_____
Подпись, Дата

Выпускная квалификационная работа

на тему: “Исследование методов повышения временной эффективности метода ветвей и границ для решения задачи коммивояжёра”

по направлению подготовки 09.03.04 “Программная инженерия”

Научный руководитель профессор департамента программной инженерии, д.т.н. М.В. Ульянов _____ Оценка _____ Подпись, Оценка	Выполнил студент 4 группы БПИ121 4 курса бакалавриата образовательной программы “Программная инженерия” М.И. Фомичёв _____ Подпись, Дата

Москва 2016

Реферат

Объектом исследования в данной работе являются точные алгоритмы решения задачи коммивояжёра, реализующие метод ветвей и границ и его различные эффективные модификации. Предмет исследования – время и выделенная память, затраченная на решение задачи коммивояжёра.

Целью данной работы является анализ различных подходов, которые могут привести к уменьшению времени решения задачи коммивояжёра с помощью метода ветвей и границ.

В процессе работы проводились экспериментальные исследования случайно сгенерированных полных графов.

Результаты исследования можно применять в задачах логистики и прикладного программирования, например, при реализации навигаторов.

Работа содержит 37 страниц, 4 главы, 6 рисунков, 9 таблиц, 30 источников.

Ключевые слова: задача коммивояжёра, метод ветвей и границ, гамильтонов цикл, двоичная куча, упорядоченный список, самобалансирующиеся деревья, муравьинный алгоритм.

Abstract

The object of the study is accurate algorithms solving the Traveling Salesman Problem, which are based on the Branch and Bound Method and its efficient modifications. The subject of the study is time and dedicated memory, required to solve the Traveling Salesman Problem.

The target of the study is an analysis of different approaches, which can reduce running time of solving the Traveling Salesman Problem.

Experimental research of random generated complete graphs was conducted in course of the work.

The result of the study can be applied to tasks of logistics and application programming, for instance, an implementation of navigators.

The paper contains 37 pages, 4 chapters, 6 illustrations, 9 tables, 30 bibliography items.

Keywords: Traveling Salesman Problem, the Branch and Bound Method, hamiltonian cycle, binary heap, ordered list, self-balancing binary search tree, ant algorithm.

Оглавление

Определения

В настоящей ВКР используются следующие термины с соответствующими определениями:

- Временная сложность (или просто сложность) алгоритма – асимптотическая оценка функции трудоёмкости [29].
- Гамильтонов цикл – это простой цикл, который проходит через каждую вершину графа G [22].
- Граф – это конечное множество V, называемое множеством вершин, и множество E (множество ребер) двухэлементных подмножеств множества V [22].
- Задача является NP-полной (NPC или NP-Complete), если она принадлежит классу NP и к ней должны полиномиально сводиться все задачи класса NP [29].
- Задача является NP-трудной (NP-hard), если эта задача в постановке принятия решения (decision problem) принадлежит к классу NP-полных задач [29].
- Класс NP – класс задач с полиномиально-проверяемым решением или класс полиномиально-проверяемых задач. Задача относится к классу NP, если её решение может быть проверено с полиномиальной временной сложностью [29].
- Класс P – класс полиномиально-решаемых задач. Задача относится к классу P, если существуют константа k и алгоритм, решающий эту задачу за время $O(n^k)$, где n - для входа [29].
- Модель вычисления – это набор базовых операций и издержки на их применения в алгоритме [29].
- Полный граф – это граф, у которого любые две вершины соединены ребром [22].
- Простой цикл – это цикл, соединяющий вершину v саму с собой и не содержащий повторяющихся вершин, кроме v [22].
- Трудоёмкость алгоритма – это количество базовых операций в принятой модели вычислений, задаваемых алгоритмом на конкретном входе [29].
- Функция трудоемкости – это отношение, связывающее входные данные алгоритма с количеством элементарных операций [29].
- Цикл (в теории графов) – это путь ненулевой длины, соединяющий вершину саму с собой и не содержащий повторяющихся рёбер [22].

Обозначения

В настоящей ВКР используются следующие обозначения:

- $t_{alg}(input)$ – время работы реализации алгоритма alg для решения задачи на конкретном входе в микросекундах.
- $M_{alg}(matrix)$ – максимальный объём памяти, потребляемый реализацией алгоритма alg на конкретном входе в байтах.
- $\bar{t}_{alg}(n)$ – среднее время работы реализации алгоритма alg на входе длины n для представленной выборки в микросекундах.
- $\overline{M}_{alg}(n)$ – средний максимальный объём памяти, потребляемый реализацией алгоритма alg на входе длины n для представленной выборки в микросекундах.
- ARS – классический алгоритм метода ветвей и границ [16] для решения задачи коммивояжёра.
- $ARS_{matrices\_in\_all\_nodes}$ – ARS с хранением матрицы в каждой вершине дерева решений.
- $ARS_{matrices\_in\_leaves}$ – ARS с хранением матрицы в каждом листе дерева решений из которого возможно дальнейшее ветвление.
- ARS_{heap} – ARS, где для хранения листьев дерева решения используется двоичная куча.
- ARS_{ord_vector} – ARS, где для хранения листьев дерева решения используется упорядоченный список.
- ARS_{RBT} – ARS, где для хранения листьев дерева решения используется красно-черное дерево.
- ARS_{greedy} – ARS с предвычисленным туром, полученным с помощью жадного алгоритма.
- ARS_{ant_x} – ARS с предвычисленным туром, полученным с помощью муравьинного алгоритма, где x– время жизни колонии.

Введение

В современном мире промедление в секунду, или даже долю секунды, может стоить миллионы рублей. Заинтересованному лицу важно получить точный ответ на вопрос в кратчайшие сроки. Но, к сожалению, даже при нынешних вычислительных мощностьях, многие задачи не могут быть решены за приемлемое время.

Задача коммивояжёра является одной из таких задач. Её исключительной особенностью является то, что для весьма небольшого, по современным меркам, объёма данных ответ можно получить не раньше чем через месяц, не говоря уже о том, что на сегодняшний день не существует способа узнать, хотя бы с точностью до 25%, через какое время будет получено решение.

Более того, целый ряд практических постановок в области бизнеса и логистики сводится к классической задаче коммивояжера. Обилие эвристических методов её решения не означает отказа от возможности получения точных методов решений этой задачи. Очевидно, что исследователи хотели бы иметь для точных методов, обладающих экспоненциальной сложностью, оценки размерности задач, решаемых за приемлемое время, равно как и модифицированные точные алгоритмы с лучшей временной эффективностью.

Целью данной работы является анализ различных подходов, которые могут привести к уменьшению времени решения задачи коммивояжёра с помощью метода ветвей и границ. Для достяжения этой цели перед автором поставлены следующие задачи:

1. Сформировать пул матриц стоимостей для изучения задачи коммивояжёра.
2. Оценить влияние выделения дополнительной памяти для хранения усеченных матриц в вершинах поискового дерева решений на время решения.
3. Проанализировать существующие структуры данных, предназначенные для хранения дерева решений и выявить наиболее рациональную.
4. Оценить влияние предвычисленного тура, полученного с помощью приближенного алгоритма, на метод ветвей и границ для решения задачи коммивояжёра.

Объектом исследования в данной работе являются точные алгоритмы решения задачи коммивояжёра, реализующие метод ветвей и границ и их различные эффективные модификации. Предметом исследование – время и выделенная память, затраченная на решение задачи коммивояжёра.

Результаты данного исследования и предыдущии работы автора [22], [25], [26], [27] по задаче коммивояжёра будут способствовать дальнейшему исследованию, направленному на кластеризацию индивидуальных задач коммивояжёра в пространстве характеристик исход-

ных матриц, идентификацию распределения времен в целях классификации и прогназирования временных характеристик индивидуальных задач.

Данная работа структурирована следующем образом: в главе один представлено описание задачи коммивояжёра и современное состояние исследования. В главе два представлен теоретический анализ исследуемых подходов к повышению временной эффективности метода ветвей и границ, а в главе три – эксперементальные результаты. В четвёртой главе проведён анализ и оценка полученных результатов.

Глава 1. Обзор задачи коммивояжёра

1.1. Содержательная постановка задачи коммивояжёра

Торговому агенту (он же *коммивояжёр*) необходимо посетить $n-1$ город и вернуться обратно в стартовый город. Ему известно, что из любого города он может добраться в любой другой город и он знает сколько ему будет стоить поездка (иными словами ему известна для каждой из пар n^2-n пар городов стоимость переезда). Его цель найти такой *тур*, чтобы его издержки были минимальными, а в каждом городе (кроме стартового) он побывал бы ровно один раз.

1.2. Принятые обозначения

Города будем обозначать арабскими цифрами, например: $1, 2, 3, .., n-1, n$.

Город из которого коммивояжёр выезжает, будем обозначать 1.

Проезд из города i в город j обозначим $i \to j$, а обозначение, что проезд закрыт – $i \nrightarrow j$.

Стоимость проезда из города i в город j обозначим $C(i \to j)$.

Часть тура будем обозначать $i \to k \to j$, что означает, что из города i едем в город k, а из города k едем в город j. Несколько несвязанных частей тура записываются через точку с запятой, например: $i \to k; p \to j$. Стоит заметить, что запись $i \to k \to j$ эквивалентна записи $i \to k; k \to j$.

Запись $C(i \to k \to j)$ обозначает стоимость части тура. Очевидно, что:

$$C(i \to k \to j) = C(i \to k) + C(k \to j) \tag{1.1}$$

Тур будем обозначать заглавными латинскими буквами, например, $T = 1 \to 3 \to 4 \to \to 5 \to 2 \to 1$, обозначает тур из 5 городов (с учётом начального города). Стоимость тура обозначим $C(T) = C(1 \to 3 \to 4 \to 5 \to 2 \to 1)$. Тогда:

$$\begin{aligned} C(T) = C(1 \to 3 \to 4 \to 5 \to 2 \to 1) = C(1 \to 3) + C(3 \to 4) + \\ + C(4 \to 5) + C(5 \to 2) + C(2 \to 1) \end{aligned} \tag{1.2}$$

1.3. Постановка задачи коммивояжёра в терминах теории графов

Если принять, что города - это вершины графа, а проезд между городами обозначим ребрами с весом (вес ребра равен стоимости проезда), то полученный граф будет полным ориентированным асимметричным графом, состоящим из n вершин без собственных петель. Соответственно, задача нахождения тура сводится к задаче нахождения гамильтонова цикла минимальной стоимости.

1.3.1. Представление графа

Полученный граф представим в виде матрицы стоимости C, где $c_{ij} = C(i \to j)$. Отсутствие собственных петель обозначим как $c_{ii} = +\infty \ \forall i = 1..n$. Пример матрицы стоимости приведён в табл. 1.

Таблица 1

Пример матрицы стоимости

	1	2	3	4	5
1	∞	10	8	5	7
2	6	∞	9	5	6
3	8	2	∞	7	4
4	6	4	11	∞	8
5	2	6	12	10	∞

1.4. Современное состояние исследований

Впервые детализация метода ветвей и границ для решения задачи коммивояжёра была предложена в [17], хотя предварительные результаты и общие формулировки идей были представлены в более ранних работах [6], [7], [11], [15], [19]. Хорошо исследован частный случай, так называемая геометрическая (планарная, евклидова) задача коммивояжёра с матрицей расстояний, для которой выполняется неравенство треугольника [2].

Задача коммивояжёра принадлежит к классу NP-трудных задач [28], это означает, что если предположение о неравенстве классов $P \neq NP$ верно, то не существует такого алгоритма, с помощью которого можно найти точное решение задачи коммивояжёра за полиномиальное время. Поэтому, для решения задачи коммивояжёра большой размерности стали применяться эволюционные, в основном генетические, метаэвристические и муравьиные алгоритмы [5], [12], [20].

Собственно, задача коммивояжёра достаточно хорошо исследована, однако, прогнозирование трудоёмкости или временных характеристик для конкретной задачи вызывает существенные трудности в связи со значительным разбросом времен при фиксированном размере матрицы стоимостей.

Проводились исследования по прогнозированию времени, требующегося для решения задачи коммивояжера методом ветвей и границ [4], [13], [16], [17]. Так, в [13] предлагается оценивать число вершин дерева решений на основе случайного выбора. В [4], [16], [17] ис-

следуется вопрос о прогнозе числа вершин дерева решений на основе анализа поддерева решений, полученного на начальном этапе расчётов. В [4] предложена процедура построения такого прогноза, проведены вычислительные эксперименты. Показано, что на основе анализа начальной стадии вычислений можно с удовлетворительной точностью оценивать время работы всего алгоритма.

В известной автору литературе исследуется также вопрос о параллельной реализации метода ветвей и границ [29], а также проблематика, связанная с обобщенной задачей коммивояжёра [8], которые не расматриваются в данной работе.

Автор данной работы участвовал в исследовании влияния способов организации и хранения поискового дерева решений на временные характеристики метода ветвей и границ для задачи коммивояжёра [22], попытка в определении вида распределения времени работы метода ветвей и границ [25] и в разработке обобщенного матричного представления индивидуальных задач [26]. А также в [31] был рассмотрен особый случий, который возникает в ходе работы метода ветвей и границ, который не был расмотрен в оригинальной статье [17].

1.5. Задача исследования

Задачей данной работы является расширение представления о проблематике задачи коммивояжёра посредствам экспериментов в попытках уменьшения времени работы метода ветвей и границ.

1.6. Выводы по главе

В данной главе представлена формальная и содержательная постановка задачи коммивояжёра и перечислены области исследования, в которых ведётся изучение данной задачи.

В следующей главе будут представленны походы, которые могут способствовать уменьшению времени поиска оптимального гамильтового цикла в полном графе.

Глава 2. Теоретический анализ исследуемых подходов

2.1. Хранение усечённых матриц стоимостей

Классический метод ветвей и границ для решения задачи коммивояжёра предполагает, что каждый раз после смены ветви дерева решений необходимо привести матрицу стоимостей. Временная сложность данной операции – $\mathcal{O}(kn^2)$, где k – глубина рассматриваемой ветви дерева, а n – размерность задачи (количество городов). С учётом того, что количество переходов с одной ветви на другую может быть достаточно частым, операция приведения является недопустимо дорогой и требует улучшения. В работе [22] предложен подход к хранению усечённых матриц стоимостей в каждой вершине дерева решений, что приводит к сокращению трудоёмкости и времени выполнения, но к существенному увеличению объёма оперативной памяти.

Таким образом, возникает классическая дилемма между временем расчёта и количеством потребляемой памяти. Для понижения потребляемой памяти можно отказаться от хранения всех матриц стоимостей и хранить только те, которые соответствуют листьям дерева решений. Данный подход должен уменьшить объём выделенной памяти, но повысить трудоёмкость алгоритма. Однако, повышение трудоёмкости не означает, что увеличится время работы алгоритма, потому что количество оперативной памяти, требуемой на хранение дерева решений, уменьшится, и у процессора и у оперативной памяти смогут появиться возможности для кеширования.

2.2. Выбор структуры данных для хранения листьев дерева решений

В ходе работы метода ветвей и границ для решения задачи коммивояжёра, возникает необходимость найти лист дерева решений с минимальной нижней границей, который соответствует множеству допустимых туров с соответствующей нижней границей. Так как операция поиска листа с минимальной нижней границей возникает в каждой итерации цикла, то эта операция должна выполняться так быстро, как возможно. Есть два способа для организации процесса поиска:

1. Обход дерева решений.
2. Хранение листьев в специальной структуре данных.

В подходе, предложенном в [17] и [27], предлагается хранить листья в специальной структуре, хотя тип этой структуры не уточняется.

2.2.1. Обход дерева решений для поиска минимального элемента

Если искать лист с минимальной нижней границей, то эта операция достаточно дорогая, так как необходимо обойти большую часть дерева, а количество вершин дерева растёт экспоненциально с размерностью задачи (количеством городов) [22]. Поэтому экспериментальный анализ будет проведён для того, чтобы показать неэффективность этого подхода.

2.2.2. Структура данных для хранения листьев дерева решений

Выбор структуры данных для хранения листьев дерева решений может иметь существенное влияние на время работы алгоритма. В первую очередь, необходимо определить, какие операции должна поддерживать структура данных и какими свойствами обладать, согласно классической реализации метода ветвей и границ:

1. Удаление наименьшего элемента.
2. Получение (без удаления) наименьшего элемента.
3. Вставка элемента.
4. Возможность хранения нескольких одинаковых элементов (несколько одинаковых ключей).
5. Возможность удаления всех элементов с ключами, большими определённого значения.

Учитывая вышеизложенные особенности стуктуры данных, можно сделать вывод, что рассматриваемая структура данных должна представлять такой абстрактный тип данных, как очередь с приоритетом [28].

Согласно [14], [24], [28], следующие структуры данных следует расмотреть как реализацию очереди с приоритетом:

- упорядоченный список;
- двоичная (бинарная) куча;
- самобалансирующиеся двоичные (бинарные) деревья поиска (например, красно-чёрные деревья и АВЛ-деревья).

Эти структуры данных поддерживают все необходимые операции и обладают нужными свойствами. Кроме того, сложность операции по получению минимального элемента может быть сведена к $\mathcal{O}(1)$, если кешировать минимальный элемент. Вдобавок, эти структуры данных демонстрируют линейное использование памяти $\mathcal{O}(n)$, где n – количество хранимых элементов. Однако, сходство структур данных на этом заканчивается. Сложность операции по удалению минимального элемента для упорядоченного списка – $\mathcal{O}(n)$, в то время, как сложность аналогичной операции для самобалансирующегося двоичного дерева и двоичной кучи – $\mathcal{O}(\log n)$. С другой стороны, сложность операции удаления элементов, чей ключ больше заданого чис-

ла – $\mathcal{O}(n)$, в то время, как для самобалансирующегося двоичного дерева и двоичной кучи – $\mathcal{O}(n \log n)$. Что касается операции вставки, то в худших случаях стоимость операции вставки для упорядоченного списка и двоичной кучи – $\mathcal{O}(n)$, а для самобалансирующегося двоичного дерева – $\mathcal{O}(\log n)$.

Вышеизложенные определения сложности представлены в табл. 2.

Таблица 2

Сложность операций над структурами данных в худшем случае

Операция	Упорядоченный список	Двоичная куча	Самобалансирующееся двоичное дерево
Удаление наименьшего элемента	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
Получение наименьшего элемента	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Вставка элемента	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
Удаление элементов, чьи ключи больше, чем заданное значение	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$

Из вышепредставленного анализа сложности операций над структурами данных в худших случаях сложно сделать какие-либо выводы о эффективности той или иной структуры, поэтому с выбранными структурами также будет проведено экспериментальное исследование.

2.3. Метод ветвей и границ с предвычисленным туром

В классическом методе ветвей и границ для решения задачи коммивояжёра предполагается, что мы не можем начать отсекать поддеревья дерева решений до тех пор, пока не найдём какой-либо тур. В виду особености задачи, дерево решений может серьёзно разрастись, пока первый тур не будет найдён.

Если до начала работы метода ветвей и границ уже будет известен какой-то тур (назовём этот тур предвычисленным), то мы можем сократить размеры дерева решений. Такой подход предоставляет следующие преимущества:

- уменьшение времени выполнения, потому что отсутствует необходимость создавать неперспективные узлы дерева решений и посещать их;

- уменьшение выделенной памяти, потому что отсутствует необходимость хранить неперспективные узлы дерева.

Таким образом, данный подход сокращает и время расчёта и объём выделенной памяти.

Однако, возникает вопрос, как можно найти предвычисленный тур? Это очевидно что чем предвычисленный тур ближе к оптимальному, тем меньше времени потребуется для нахождения оптимального решения. С другой стороны, предвычисленный тур должен быть найден достаточно быстро. Это значит, что время затраченное на работу классического метода ветвей и границ, по крайней мере, должно быть не больше, чем время расчёта метода ветвей и границ с предвычисленным туром. Другими словами, использование предвычисленного тура должно быть оправданным и рациональным.

Предвычисленный тур может быть найден приближёнными алгоритмами, например, методом эластичной сети [10], алгоритмом Лина-Кернигана [16]. Однако, в этой работе будут проанализированы жадный алгоритм (так же известный как алгоритм ближайшего соседа) и, хорошо себя зарекомендовавший, муравьиный алгоритм.

Жадный алгоритм является одним из самых простых в реализации и понимании принципа работы. Он основывается на выборе самого дешевого перехода между городами последовательно для каждого города, начиная с 1-го и заканчивая n-ым, запрещая возможные подциклы. Нет никаких сомнений, что этот алгоритм не гарантирует точное решение задачи коммивояжёра, более того, тур, полученный с помощью жадного алгоритма, может быть очень далёк от оптимального. В тоже время, он работает достаточно быстро ($\Theta(n^2)$ в худшем, среднем и лучшем случае).

Муравьиный алгоритм для задачи коммивояжёра был предложен коллективом авторов во главе с Марко Дориго в 1991 г. [3]. Он заключается в эмуляции поведения муравьиной колонии. Его точность зависит от следующих параметров:

- время жизни колонии;
- количество муравьёв;
- начальное расположение муравьёв;
- начальное количество феромона;
- объём выделенного феремона;
- коэффициент испарения;
- количество элитных муравьёв.

Вероятность перехода k-го муравья из города i в город j в момент времени t определяется следующей формулой:

$$P_{i,j,k}(t) = \begin{cases} \dfrac{(\tau_{i,j}(t))^{\alpha}(\eta_{i,j})^{\beta}}{\sum\limits_{l \subseteq J_{j,k}} (\tau_{i,l}(t))^{\alpha}(\eta_{i,l})^{\beta}} & j \subseteq J_{j,k} \\ 0 & j \nsubseteq J_{j,k} \end{cases} \tag{2.1}$$

где $\tau_{i,j}(t)$ – количество феромона между городом i и городом j в момент времени t;

$\eta_{i,j}$ – видимость города j из города i, которая определяется согласно правилу: $\eta_{i,j} = C(i \to j)$;

$J_{j,k}$ – множество городов (*tabu list*), в которые возможно перейти k-му муравью из города i, так, чтобы в туре не было повторяющихся городов и подциклов;

α – параметр, задающий вес следа феромона;

β – параметр, задающий вес видимости другого города.

Стоит заметить, что матрица видимости (η) не изменяется на протяжение всего алгоритма и одинакова для всех муравьёв, количество феромона между городами (τ) изменяется во время жизни колонии (в конце каждой итерации), но, также как и видимость, одинаково для всех муравьёв в конретный момент времени. С другой стороны, множество $J_{j,k}$ у разных муравьёв разное, поэтому вероятности перехода между одинаковыми городами у разных городов отличаются.

Кроме того, формула (2.1) является вероятностно-пропорциональным правилом, что означает, что вероятность – это "ширина" города j, далее генерируется случайное число (от нуля до еденицы), по которому муравей и выбирает свой дальнейший путь (это правило также известно как "колесо рулетки").

После того, как каждый муравей нашёл тур, необходимо обновить феромон на матрице феромонов по следующему правилу:

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \Delta\tau_{i,j}(t) + \Delta\tau_{e}(t) \tag{2.2}$$

$$\Delta\tau_{i,j}(t) = \sum_{k=1}^{m} \Delta\tau_{i,j,k}(t) \tag{2.3}$$

$$\Delta\tau_{i,j,k}(t) = \begin{cases} \dfrac{Q}{C(T_{k,t})} & i \to j \subseteq T_{k,t} \\ 0 & i \to j \nsubseteq T_{k,t} \end{cases} \tag{2.4}$$

$$\Delta\tau_{e}(t) = \begin{cases} \dfrac{eQ}{C(T_t^*)} & i \to j \subseteq T_t^* \\ 0 & i \to j \nsubseteq T_t^* \end{cases} \tag{2.5}$$

где T_t^* – наилучший тур, найденный к моменту t,

e – количество элитных муравьёв,

Q – регулируемый параметр, влияющий на количество феромона,

$T_{k,t}$ – тур, пройденный k-ым муравьём в момент времени t,

m – количество муравьёв,

ρ – коэфициент испарения феромона за одну итерацию.

На основе [3] и правил, изложенных выше, можно составить псевдокод алгоритма (псевдокод приведён в схеме 1).

```
1  Input: t_max, m, α, β, e, Q, τ_0, ρ, C.
2  Output: T*_{t_max}.
3  Begin
4    init vision (η) and pheromone (τ), where η_{i,j} := 1/C(i→j) and τ_{i,j} := τ_0 ∀i ≠ j;
5    put all ants it own random city without repetition;
6    for t := 1 to t_max:
7      for i := 1 to m:
8        find T_{i,t} according to (2.1);
9        if C(T_{i,t}) < C(T*_{best}) then:
10         T*_{best} := T_{i,t};
11   for each element of τ:
12     update τ_{i,j} according to (2.2), (2.3), (2.4) and (2.5);
13   return T*_{best};
14 End.
```

Схема 1. Псевдокод муравьиного алгоритма

Исходя из схемы 1, можно вычислить сложность муравьиного алгоритма $\mathcal{O}(t_{max}mn^2)$, где t_{max} – время жизни колонии, m – количество муравьёв и n – количество городов. Согласно [1], [3] и [9], рекомендуемые праметры для муравьиного алгоритма представлены в табл. 3.

В табл. 3 T_{greedy} – тур, полученный с помощью жадного алгоритма. Так как рекомендуемое количество муравьёв равно количеству городов ($m = n$), то сложность муравьиного алгоритма выраждается в $\Theta(t_{max}n^3)$. Что касается t_{max}, то согласно [3] и [9], чем больше значение, тем приближённый тур ближе к оптимальному, но этот параметр имеет серьёзное влияние на время работы.

Таблица 3

Рекомендуемые параметры для муравьиного алгоритма

Параметр	Значение
m	n
α	1
β	5
e	8
Q	100
τ_0	$\frac{1}{nC(T_{greedy})}$
ρ	0.5

Хотя тур, полученный с помощью муравьиного алгоритма, достаточно близок к оптимальному [3], используя его для нахождения предвычисленного тура, нельзя забывать о рациональном использовании.

2.4. Выводы по главе

В данной главе перечислены и теоретически проанализированы расматриваемые в этой работе подходы для повышения временной эффективности метода ветвей и границ в задаче коммивояжёра, а именно:

- дополнительная память для хранения усечённых матриц;
- структура данных для хранения листьев поискового дерева;
- предвычисленный тур.

В следующей главе будут представлены характеристики исследуемой выборки и экспериментальные результаты.

Глава 3. Экспериментальное исследование

3.1. Описание плана эксперимента

3.1.1. Входные данные

Для исследования сформирован пул несимметричных матриц стоимостей. Размерность матриц от 20 до 45 с шагом один. Количество различных входов для каждой размерности: 100000. Элементами матриц (стоимости перехода между городами) генерировались с помощью стандартного равномерного генератора псевдослучайных чисел в интервале $(0; 1)$ с точностью до шестого знака после запятой.

3.1.2. Аппаратные и программные особенности компьютера

Все эксперименты проводились на персональном компьютере со следующими характеристиками:

- процессор Intel i7 3770K 3800 МГц;
- оперативная память Kingston KHX1600C9D3P1 16 ГБ;
- материнская плата GIGABYTE GA-Z77X-D3H.

В качестве операционной системы использовался дистрибутив Linux CentOS 7.0 с версией ядра 3.10.0-3247.4.4.e17. Для минимизации шумов операционой системы, были отключены ненужные для исследования фоновые процессы (например, фаервол), а также у дистрибутива отсутсвуют компоненты графического пользовательского интерфейса (управление осуществляется посредством командной строки). Более того, отключён своппинг, чтобы скорость работы HDD не сказывалась на времени работы реализации.

3.1.3. Особенности реазлиции

Реализация всех алгоритмом осуществлялась на языке C++ с использованием библиотеки стандартных шаблонов (STL). Версия компилятора: GNU GCC 4.8.5 20150623.

Замер времени работы алгоритмов осуществлялся посредством chrono. А для замера используемой памяти использовались счётчики.

Наличие погрешности у чисел с плавующей точкой (согласно стандарту IEEE 754-2008) может привести к изменённому процессу ветвления. Как показывает практика, хотя процесс ветления и изменяется, и ошибка может накапливаться, это не влияет на поиск оптимального тура.

3.2. Эксперементы по хранению усечённых матриц

В табл. 4, табл. 5 и на рис. 1 и рис. 2 показаны результаты экспериментов для классического метода ветвей и границ (ARS), для метода ветвей и границ с хранением усеченных

матриц стоимостей в каждой вершине дерева решений ($ARS_{matrices\_in\_all\_nodes}$) и для метода ветвей и границ с хранением усеченных матриц стоимостей в каждом листе дерева решений ($ARS_{matrices\_in\_leaves}$).

Таблица 4

Среднее затраченное время (в мкс) для ARS, $ARS_{matrices\_in\_all\_nodes}$ и $ARS_{matrices\_in\_leaves}$

n	$\bar{t}_{ARS}(n)$	$\bar{t}_{ARS_{matrices\_in\_all\_nodes}}(n)$	$\bar{t}_{ARS_{matrices\_in\_leaves}}(n)$
20	2228	1026	904
21	2993	1322	1174
22	4134	1700	1504
23	5533	2243	1958
24	7243	2886	2549
25	9534	3652	3240
26	12258	4619	4053
27	16266	5830	5159
28	20917	7289	6432
29	27167	9269	8060
30	35197	11750	10163
31	45411	14784	12731
32	58333	18499	15818
33	74381	23029	19559
34	96740	29386	24805
35	123770	36803	30974
36	159891	46562	39161
37	204587	58352	48997
38	261227	72985	61323
39	333328	91404	76778
40	424082	114253	96085
41	537207	141658	119288
42	680455	176311	148712
43	872727	221532	187109
44	1093284	273760	231118
45	1383283	339270	286855

Таблица 5

Средняя затраченная память (в байтах) для ARS, $ARS_{matrices\_in\_all\_nodes}$ и $ARS_{matrices\_in\_leaves}$

n	$\overline{M}_{ARS}(n)$	$\overline{M}_{ARS_{matrices\_in\_all\_nodes}}(n)$	$\overline{M}_{ARS_{matrices\_in\_leaves}}(n)$
20	7056	276052	140393
21	7744	365946	185348
22	8464	481596	243056
23	9216	632730	318469
24	10000	823062	413373
25	10816	1075206	538891
26	11664	1386038	693484
27	12544	1803587	901551
28	13456	2320088	1158351
29	14400	2975028	1484120
30	15376	3831116	1909913
31	16384	4881904	2432272
32	17424	6220723	3098717
33	18496	7853374	3910350
34	19600	10099647	5026411
35	20736	12777555	6359054
36	21904	16373952	8148964
37	23104	20707256	10300458
38	24336	26173095	13022067
39	25600	33067161	16449163
40	26896	41659335	20725322
41	28224	52254346	25996348
42	29584	65522775	32597173
43	30976	83039754	41309813
44	32400	103625078	51547868
45	33856	129380269	64347978

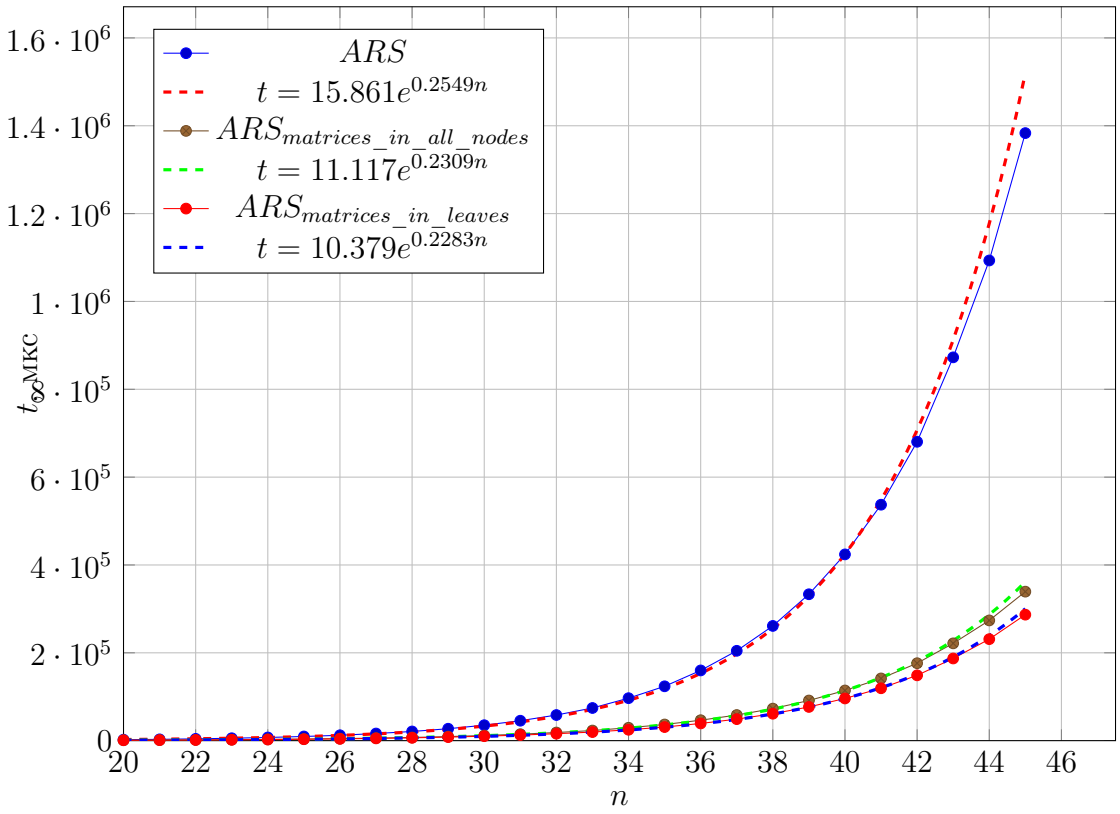

Рисунок 1. Среднее время работы алгоритмов с хранением и без хранения усечённых матриц

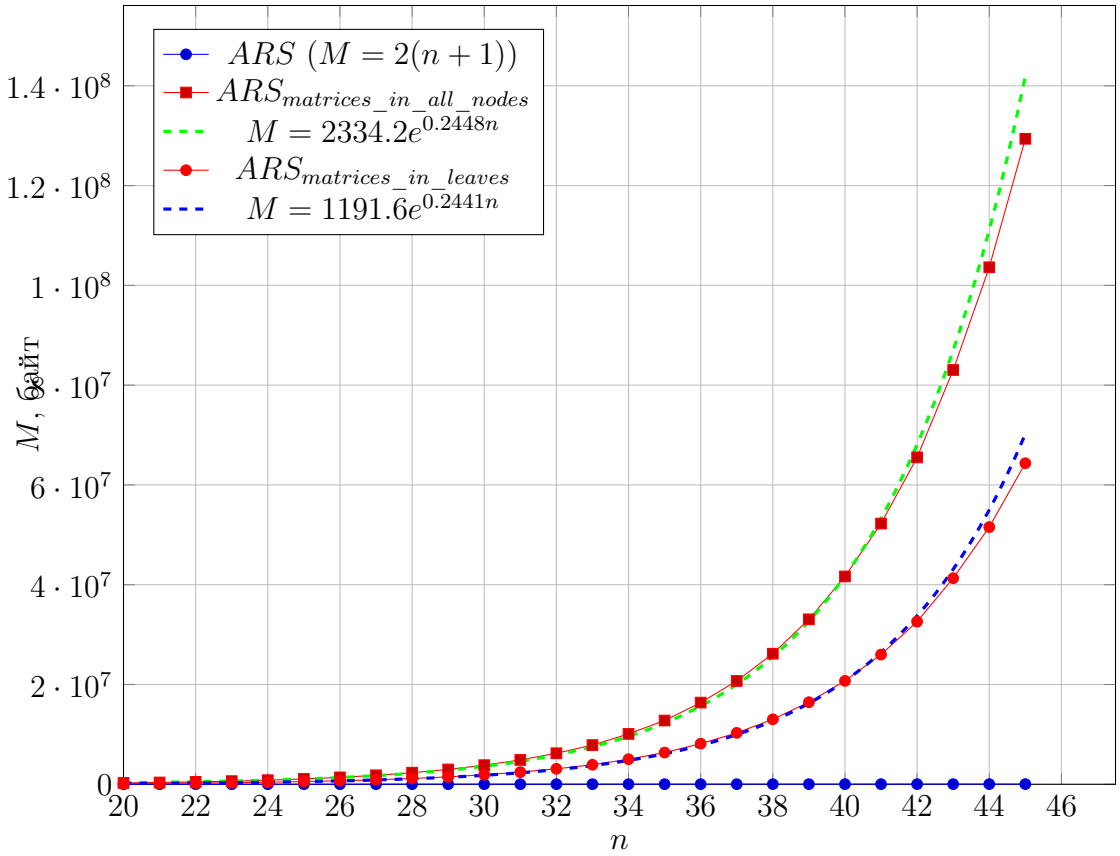

Рисунок 2. Средняя затраченная память алгоритмов с хранением усечённых матриц

3.3. Эксперементы по использованию разных структур данных

В табл. 6 и на рис. 3, рис. 4 показаны результаты экспериментов для метода ветвей и границ, для метода ветвей и границ с использованием двоичной кучи (ARS_{heap}), упорядоченного списка (ARS_{ord_vector}) и самобалансирующегося двоичного дерева (в качестве представителя данного класса структур данных используется красно-чёрное дерево [28]) (ARS_{RBT}), кроме того для сравнения представлены результаты работы реализации метода ветвей и границ без хранения списка вершин, а поиск вершины дерева решений с минимальной верхней границей осуществляется по средствам обхода дерева решений ($ARS_{traverse\_tree}$).

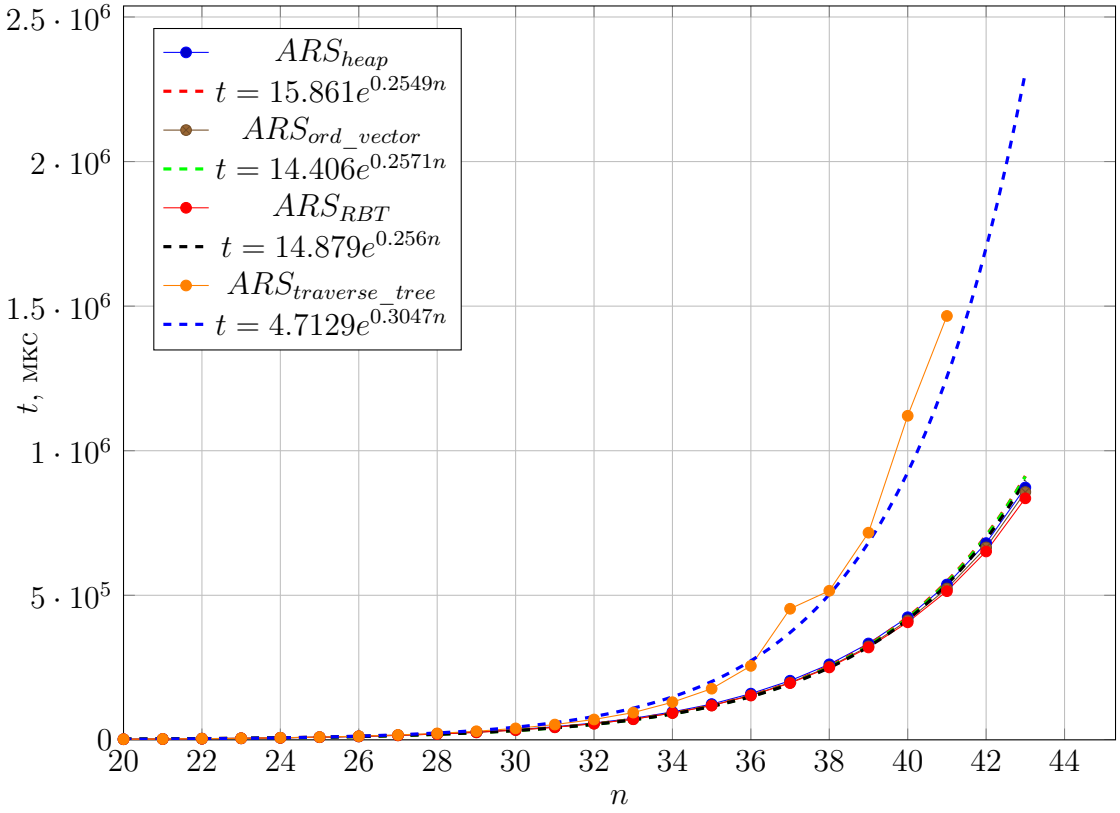

Рисунок 3. Средняя время работы алгоритмов с разными структурами данных

Таблица 6

Средняя затраченное время (в мкс) для ARS, ARS_{heap}, ARS_{ord_vector}, ARS_{RBT} и $ARS_{traverse\_tree}$

n	$\bar{t}_{ARS_{heap}}(n)$	$\bar{t}_{ARS_{ord\_vector}}(n)$	$\bar{t}_{ARS_{RBT}}(n)$	$\bar{t}_{ARS_{traverse\_tree}}(n)$
20	2228	2174	2187	2297
21	2993	2915	2928	3076
22	4134	4021	4004	4255
23	5533	5353	5379	5663
24	7243	6976	7000	7440
25	9534	9138	9220	9799
26	12258	11893	11957	12827
27	16266	15610	15676	17146
28	20917	20121	20155	22440
29	27167	26121	26128	29815
30	35197	33841	33828	39710
31	45411	43606	43606	52947
32	58333	56012	56005	70502
33	74381	71445	71378	94337
34	96740	93028	92799	130210
35	123770	119037	118646	176990
36	159891	154136	153307	255872
37	204587	198026	196167	453380
38	261227	252851	250524	515642
39	333328	323256	319616	716533
40	424082	413123	406551	1120809
41	537207	523332	514260	1466065
42	680455	665432	651617	–
43	872727	857094	835382	–

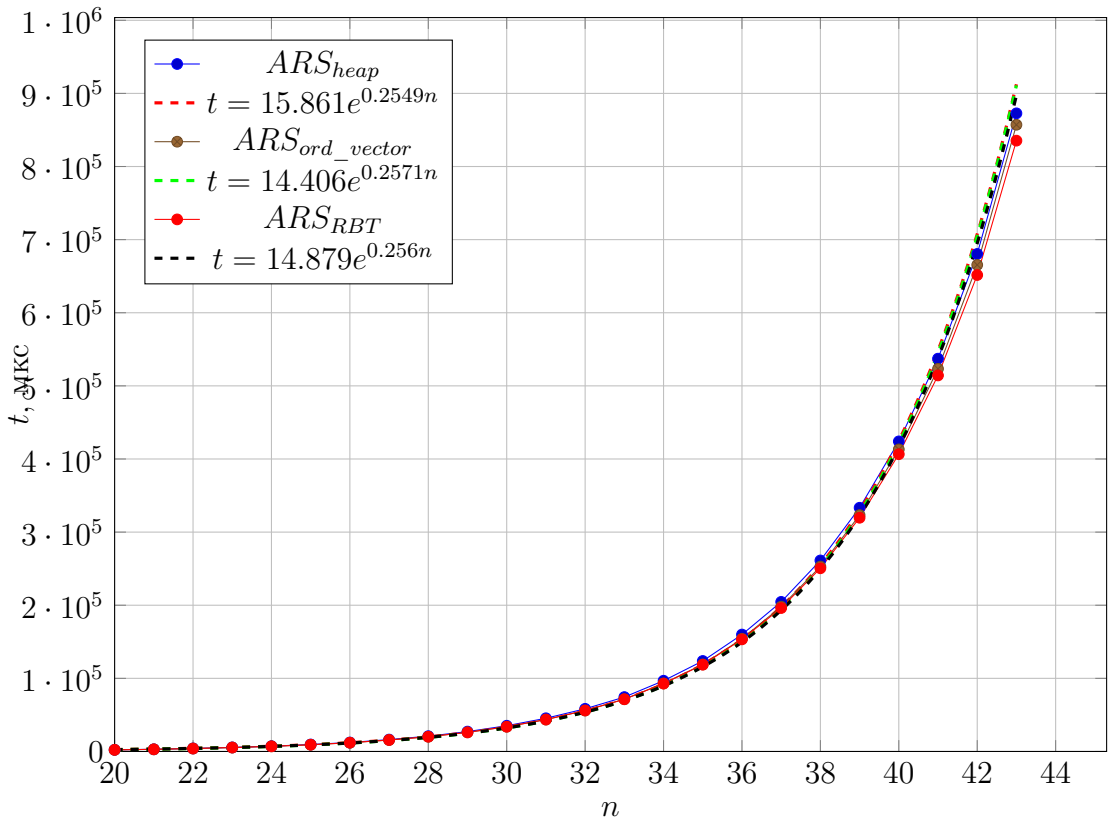

Рисунок 4. Средняя время работы алгоритмов с разными структурами данных

3.4. Эксперементы по использованию предвычисленного тура

В табл. 7 и на рис. 5 представлены результаты расчёта классического метода ветвей и границ (ARS), метод ветвей и границ с предвычисленным туром полученным с помощью жадного алгоритма (ARS_{greedy}), с помощью муравьинного алгоритма, где $t_{max} = 2n$ (ARS_{ant_2n}), $t_{max} = n$ (ARS_{ant_n}), $t_{max} = n/2$ ($ARS_{ant\_n/2}$).

Таблица 7

Среднее затраченное время (в мкс) для ARS, ARS_{greedy}, ARS_{ant_2n}, ARS_{ant_n} и $ARS_{ant\_n/2}$

n	$\bar{t}_{ARS}(n)$	$\bar{t}_{ARS_{greedy}}(n)$	$\bar{t}_{ARS_{ant\_2n}}(n)$	$\bar{t}_{ARS_{ant\_n}}(n)$	$\bar{t}_{ARS_{ant\_n/2}}(n)$
30	36064	35991	103126	69154	52666
31	46189	45997	122431	83832	64393
32	58595	58421	145123	101319	79980
33	75763	75637	173840	124189	99335
34	95272	95074	205753	150077	122596
35	122840	122526	246564	184048	152449
36	160253	159982	298927	228747	194414
37	200241	199928	356744	276314	237227
38	268470	268486	448657	353119	310908
39	331927	331922	532891	425789	377933
40	426732	426760	650774	530512	478796
41	549072	548974	797805	663296	604769
42	682981	682063	958466	807605	745104
43	862458	861599	1144220	995092	928952
44	1050941	1049858	1356959	1200666	1124756
45	1398876	1398777	1733965	1560646	1478504

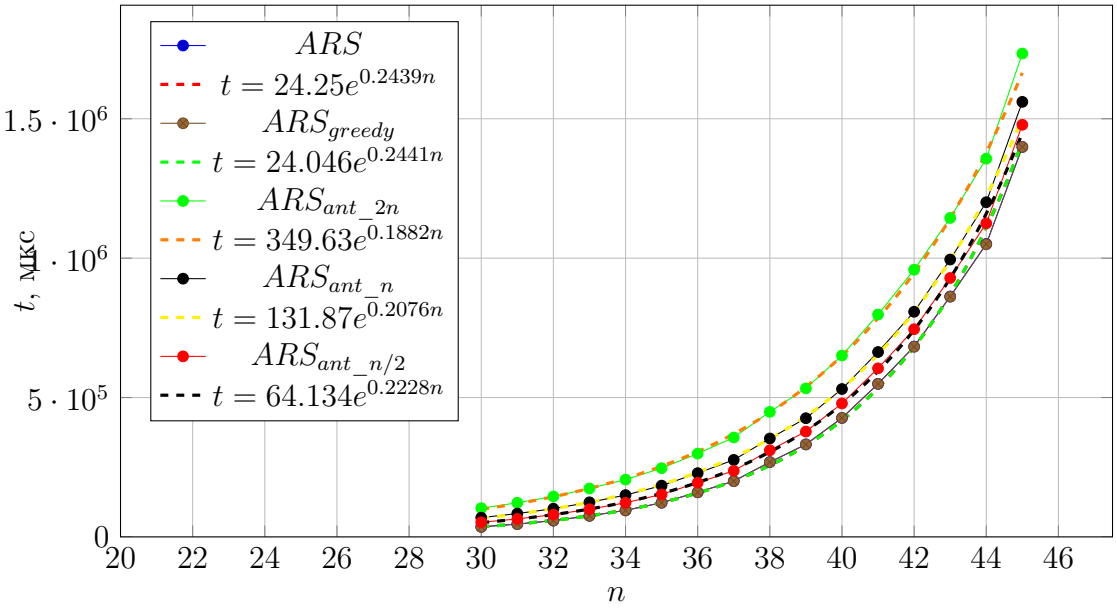

Рисунок 5. Средняя время работы алгоритмов с разными структурами данных

3.5. Эксперементы на втором персональном компьютере

Для проверки того, насколько зависит время работы метода ветвей и границ для решения задачи коммивояжёра от характеристик персонального компьютера, реализация ARS запускается на втором персональном компьютере со следующеми характеристиками:

- процессор Intel i3 2100 3100 МГц;
- оперативная память Kingston KVR16N11S6/2 4 ГБ;
- материнская плата GIGABYTE GA-H61M-DS2 (rev. 3.0).

В качестве операционной систестемы использовался дистрибутив Linux Fedora 23 с версией ядра 4.4.8-300. Для минимизации шумов операционой системы были отключены ненужные для исследования фоновые процессы (например, фаервол), а также у дистрибутива отсутсвуют компоненты графического пользовательльского интерфейса (управление осуществляется посредством командой строки). Более того, отключён своппинг, чтобы скорость работы HDD не сказывалась на времени работы реализации.

Результаты эксперементов представленны в табл. 8. и на рис. 6, где

- ARS_I и $ARS_{I\_matrices\_in\_all\_nodes}$ – ARS и $ARS_{matrices\_in\_all\_nodes}$ работающие на компьютере, на котором проводились все эксперементы;
- ARS_{II} и $ARS_{II\_matrices\_in\_all\_nodes}$ – ARS и $ARS_{matrices\_in\_all\_nodes}$ работающие на компьютере, который описан выше.

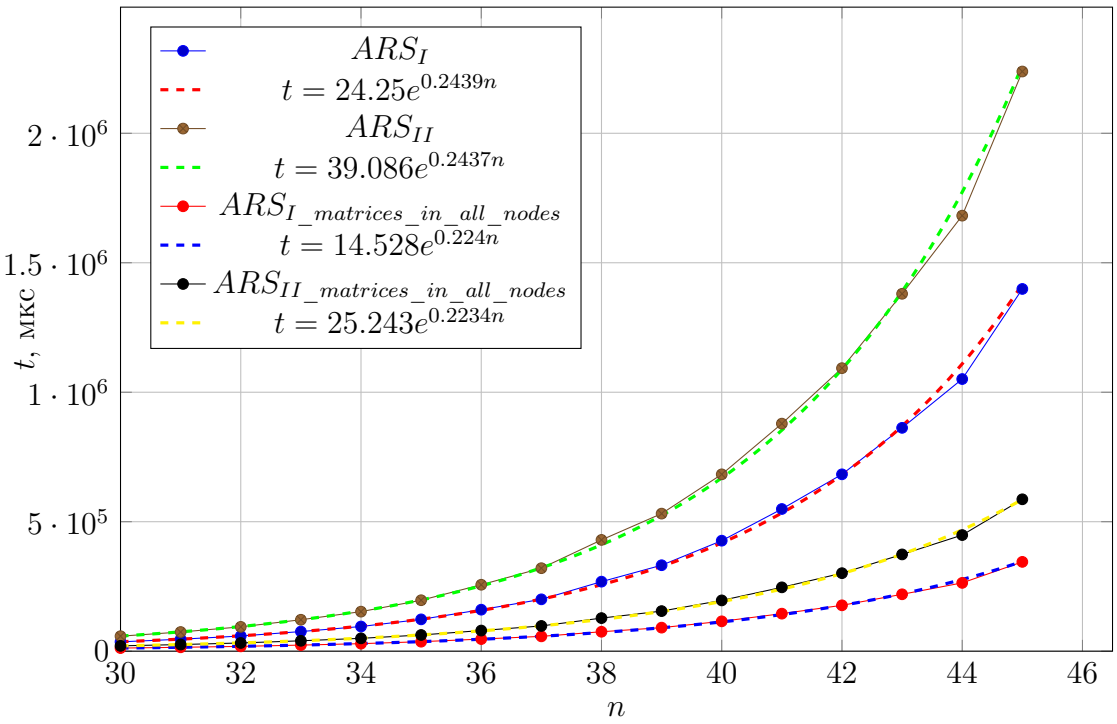

Рисунок 6. Средняя время работы алгоритмов с хранением усечённых матриц на разных компьютерах

Таблица 8

Среднее затраченное время (в мкс) для ARS и $ARS_{matrices\_in\_all\_nodes}$ на разных компьютерах

n	$\bar{t}_{ARS_I}(n)$	$\bar{t}_{ARS_{II}}(n)$	$\bar{t}_{ARS_{I\_matrices\_in\_all\_nodes}}(n)$	$\bar{t}_{ARS_{II\_matrices\_in\_all\_nodes}}(n)$
30	36064	57908	11984	20573
31	46189	74050	14984	25624
32	58595	93931	18556	31716
33	75763	121369	23496	40092
34	95272	152678	29032	49603
35	122840	196751	36649	62564
36	160253	256561	46772	79652
37	200241	320482	57250	97444
38	268470	429713	75028	127747
39	331927	531221	90915	154785
40	426732	682911	115263	196088
41	549072	878734	144923	246530
42	682981	1092971	177121	301236
43	862458	1380129	219702	373664
44	1050941	1681915	263758	448419
45	1398876	2238547	344894	586570

Исходя из полученных результатов, можно предположить, что время работы реализаций существенно зависит от аппаратных и программных особенностей компьютера, на котором проводились экспериментальные расчёты, однако, закономерности во времени остаются, по крайней мере, на домашних персональных компьютерах.

3.6. Выводы по главе

В данной главе представленны эксперементальные результаты для рассматриваемых в этой работе подходов повышения временной эффективности метода ветвей и границ для задаче коммивояжёра, а именно:

- дополнительная память для хранения усечённых матриц стоимостей;
- структура данных для хранения листьев поискового дерева;
- предвычисленный тур, полученный с помощью жадного алгоритма и муравьиного.

В следующей главе будет представлен анализ и интерпретация полученных результатов.

Глава 4. Интерпретация и анализ результатов экспериментов

4.1. Анализ влияния хранения усечённых матриц стоимостей

Экспериментальные данные показали, что хранение усеченных матриц стоимостей в узлах поиского дерева решений существенно уменьшает время расчёта. Уже, начиная с 45 городов, прирост в производительности по сравнению с классической реализацией составляет более, чем в 4 раза. С другой стороны, объём потребляемой памяти растёт экспоненциально с ростом количества городов. Если рассматривать алгоритм с хранением усеченных матриц только в листьях дерева решений, то объём потребляемой памяти уменьшается приблизительно в два раза, но всё также растёт экспоненциально. Особенностью данного алгоритма, является то, что он работает быстрее по сравнению с алгоритмом, где хранятся усеченные матрицы в каждой вершине дерева, не смотря на увеличение функции трудоёмкости алгоритма. Это связано с особенностью работы персональных компьютеров (например, кэширование). С помощью метода наименьших квадратов произведена апроксимация точечных значений, и с помощью полученных закономерностей в табл. 9 представлен прогноз для классического метода ветвей и границ (ARS) и для метода ветвей и границ с хранением усечённых матриц стоимостей в листьях дерева решений ($ARS_{matrices\_in\_leaves}$).

Таблица 9

Прогноз ресурсных характеристик для ARS и $ARS_{matrices\_in\_leaves}$

n	$\bar{t}_{ARS}(n)$	$\bar{t}_{ARS_{matrices\_in\_leaves}}(n)$	Прогноз отношения времен	Прогноз средних затрат дополнительной памяти
45	1.38 с	0.34 с	4.05	61.33 МБ
50	5.44 с	1.18 с	4.61	226.92 МБ
69	11.5 мин	1.2 мин	9.58	22.9 ГБ
79	2.45 ч	11.77 мин	12.5	263.1 ГБ
87	18.84 ч	1.22 ч	15.44	1.81 ТБ
100	21.57 сут	1 сут	21.57	43.25 ТБ

Также, стоит отметить, что начиная с 68 городов прогнозируемые затраты дополнительной памяти уже привышают объём оперативной памяти персонального компьютера в 16 ГБ.

4.2. Анализ влияния выбора структуры данных для хранения листьев дерева решений

Экперементальные результаты показали, что выбор структуры данных не оказывает существенного влияния на время работы алгоритма. Использование двоичной кучи даёт худший результат для всей рассматриваемой выборки. Упорядоченный список работает лучше самобалансирующегося двоичного дерева до $n = 29$, а начиная с $n = 30$ – самобалансирующееся дерево показывает лучший результат. Таким образом, можно сделать рекомендацию, что до $n = 29$ стоит использовать двоичную кучу, а начиная с $n = 30$ – самобалансирующиееся деревья. Это свидетельствует о том, что потребность в операции вставки элемента в очередь с приоритетом растёт с количетсовом городов, а сложность этой операции у красно-черного дерева, в худшем случае, минимальная по сравнению с другими рассматриваемыми структурами данных ($\mathcal{O}(\log n)$).

4.3. Анализ влияния предвычисленного тура

4.3.1. Анализ влияния предвычисленного тура, полученного с помощью жадного алгоритма

Из табл. 7 видно, что предвычисленный тур, полученный с помощью жадного алгоритма, оказывает положительный эффект на метод ветвей и границ. Так как сложность жадного алгоритма достаточно низкая ($\Theta(n^2)$), и время работы метода ветвей и границ с предвычисленным туром меньше чем без него, то использование жадного алгоритма для предвычисленного тура является рациональным. Однако, с увеличением размера задачи эффект от жадного алгоритма уменьшается.

4.3.2. Анализ влияния предвычисленного тура, полученного с помощью муравьинного алгоритма

Из табл. 7 видно, что предвычисленный тур, полученный с помощью муравьинного алгоритма, оказывает негативное влияние на время решения задачи коммивояжёра. Использование муравьинного алгоритма для нахождения предвычисленного тура является нерациональным, то есть эффект от предвычисленного тура меньше, чем временные затраты на поиск самого тура. Такая закономерность наблюдается, по крайеней мере, до $n = 45$. Если с помощью метода наимньших квадратов произвести апроксимацию точечных значений, то получим следующие зависимости времени от размера задачи:

$$ARS : t = 15.861e^{0.2549n} \tag{4.1}$$

$$ARS_{ant\_n/2} : t = 64.134e^{0.2228n} \tag{4.2}$$

$$ARS_{ant\_n} : t = 131.87e^{0.2076n} \tag{4.3}$$

$$ARS_{ant\_2n} : t = 349.63e^{0.1882n} \tag{4.4}$$

Если приравнять уравнения (4.1), (4.2), (4.3), (4.4) и (4.5) (попарно), то можно расчитать, начиная с какого n использование муравьинного алгоритма будет рационально. Таким образом, можно выявить следующую рекомендацию:

- до $n = 43$ – не использовать муравьинный алгоритм;
- с $n = 44$ до $n = 48$ – использовать муравьинный алгоритм с $t_{max} = n/2$;
- с $n = 49$ до $n = 50$ – использовать муравьинный алгоритм с $t_{max} = n$;
- с $n = 50$ – использовать муравьинный алгоритм с $t_{max} = 2n$.

К сожалению, данные представленные в табл. 7, опровергают данную рекомендацию для $n = 44$ и $n = 45$. Для более точных рекомендаций необходимо расширение рассматриваемого интервала для n, что может стать дальнейшей темой для исследования.

Заключение

Задача коммивояжёра является одной из ключевых задач по комбинаторике и теории графов. Не смотря на то, что задача принадлежит к классу NP-трудных задач, это не означает, что стоит отказаться от нахождения оптимального решения и использовать только приближённые алгоритмы для решения задачи коммивояжёра.

В рамках выполнения выпускной квалификационной работы, был проведён анализ трех подходов к уменьшению времени работы метода ветвей и границ для решения задачи коммивояжёра, а именно:

- дополнительная память для хранения усечённых матриц;
- выбор структуры данных для хранения листьев поискового дерева;
- предвычисленный тур.

В процессе выполнения работы были решены следующие задачи:

1. Сформировать пул матриц стоимостей для изучения задачи коммивояжёра.
2. Оценить влияние выделения дополнительной памяти для хранения усеченных матриц в вершинах поискового дерева решений на время решения.
3. Проанализировать существующие структуры данных, предназначенные для хранения дерева решений и выявить наиболее рациональную.
4. Оценить влияние предвычисленного тура, полученного с помощью приближенного алгоритма, на метод ветвей и границ для решения задачи коммивояжёра.

Исходя из полученных результатов можно сделать следующие выводы:

- дополнительная память существенно уменьшает время работы метода ветвей и границ, однако, по прогнозам, начиная уже с $n = 68$, необходимый объём оперативной памяти превысит популярный размер оперативной памяти персонального компьютера в 16 ГБ;
- из рассматриваемых структур данных для хранения листьев дерева решений наиболее подходящей является упорядоченный лист до $n = 29$ и красно-чёрное дерево, начиная с $n = 30$;
- предвычисленный тур, полученный с помощью жадного алгоритма оказывает практически незначительное влияние на время работы метода ветвей и границ;
- использование предвычисленного тура, полученного с помощью муравьинного алгоритма, является нерациональным, по крайней мере, до $n = 45$.

Дальнейшая работа по решению задачи коммивояжёра автору представляется в виде попыток применения метода классов для выделения множества входов с одинаковой трудоёмкостью. Это позволит анализировать не каждую конкретную задачу, а множество задач,

обладающих одинаковыми особенностями.

Список использованных источников

1. Bonavear E., Dorigo M., Swarm Intelligence: from Natural to Artificial Systems, Oxford, the UK: OUP, 1999.
2. Charikar M., Goemans, M. X.; Karloff, H. On the Integrality Ratio for the Asymmetric Traveling Salesman Problem. Mathematics of Operations Research;May2006, Vol. 31 Issue 2, p. 245.
3. Colorni A., Dorigo M., Maniezzo V., "Distributed Optimization by Ant Colonies,"Proceedings of the First European Conference on Artificial Life, Paris, France, F.Varela and P.Bourgine (Eds.), Elsevier Publishing, 134–142, 1991.
4. Cornuéjols, G.; Karamanov, M.; and Li, Y. 2006. Early estimates of the size of branch-and-bound trees. INFORMS Journal on Computing. 2006, v. 18, No. 1, pp. 86—96.
5. Cotta C., Aldana J., Nebro A., Troya J., Hybridizing genetic algorithms with branch and bound techniques for the resolution of the TSP, in: D. Pearson, N. Steele, R. Albrecht (Eds.), Artificial Neural Nets and Genetic Algorithms 2, Springer-Verlag, Wien New York, 1995, pp. 277-280.
6. Dantzig G., Fulkerson R., Johnson S. M.. 1954. Solution of a large scale traveling salesman problem. Technical Report P-510. RAND Corporation, Santa Monica, California, USA.
7. Dantzig G. B., Fulkerson D. R. , Johnson S. M. 1959. On a linearprogramming, combinatorial approach to the traveling-salesman problem. Operations Research 7, 58–66.
8. Dimitrijevic V., Milosavljevic M., Markovic M. Branch and bound algorithm for solving a generalized traveling salesman problem // Univ. Beograd. Publ. Elektrotehn. Fak. Ser. – Mat. 7 (1996). – P. 31-35.
9. Dorigo M., Gambardella L. M., Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 53-66, Apr. 1997.
10. Durbin R. and Willshaw D., An analogue approach to the travelling salesman problem using an elastic net method, Nature, no. 326., pp. 689-691, Apr. 1987.
11. Eastman, W. L. Linear Programming with Pattern Constraints. Ph.D. Thesis. Department of Economics, Harvard University, Cambridge, Massachusetts, USA, 1958.
12. Goldberg D., Lingle R. J., Alleles, loci, and the travelling salesman problem, in: J. Grefenstette (Ed.), Proceedings of an International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale NJ, 1985.
13. Knuth, D. E., Estimating the efficiency of backtracking programs. Mathematics of Computing, v. 29, pp. 121–136, 1975.

14. Knuth, D. E. The Art of Computer Programming: Sorting and Searching, 2nd ed. Cambridge, MA: Addison-Wesley, 1998.
15. Land A.H., Doig A.G. An automatic method of solving discrete programming problems // Econometrica. – 1960. – V. 28 №3. – P. 497-520.
16. Lin S., Kernighan B. , An effective heuristic algorithm for the traveling salesman problem, Oper. Res., vol. 21, no. 2, pp. 498-516, Apr. 1963.
17. Little J.D.C., Murty K.G., Sweeney D.W., Karel C. An algorithm for the traveling salesman problem // Operat. Res., 11 (1963), pp.972-989.
18. Lobjois, L., M. Lemaitre. Branch-and-bound algorithm selection by performance prediction. American Association for Artificial Intelligence, Menlo Park, CA. 1998.
19. Manne, A. S., H. M. Markowitz. On the solution of discrete programming problems. Technical Report P-711. RAND Corporation, Santa Monica, California, USA, 1956.
20. Oliver I., Smith D., Holland J., A study of permutation crossover operators on the traveling salesman problem. J. Grefenstette (Ed.), Proceedings of the 2nd International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale NJ, 1987, pp. 224-230.
21. Purdom, P. W. Tree size by partial backtracking. SIAM J.Comput. 1978, v.7, No 4, pp. 481–491.
22. Ulyanov M. V., Fomichev M. I., Resourse characteristics of ways to organize a decision tree in the branch-and-bound method for the traveling salesman problem, Business Informatics, no. 34, pp. 38-46, Dec. 2015.
23. Андерсон Д.А. Дискретная математика и комбинаторика – М. : Издательский дом "Вильямс", 2004 – 960 с.: ил. – Парал. тит. англ. ISBN 5-8459-0498-6 (рус.).
24. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона / Пер. с англ. Ткачев Ф.В. – М.: ДМК Пресс, 2010. – 272 с.
25. Головешкин В.А., Жукова Г.Н., Ульянов М.В., Фомичёв М.И.: Сравнение ресурсных характеристик трвдиционного и модифицированного метода ветвей и границ для TSP / Современные информационные технологии и ИТ-образование. Т. 2 (№ 11), стр. 150, 2015. – 614 с. (ISSN 2411-1473).

    Головешкин В.А., Жукова Г.Н., Ульянов М.В., Фомичев М.И. Об одном обобщенном представлении классов индивидуальных задач коммивояжера // Автоматизация. Современные технологии. 2016. (Статья принята к публикации).
26. Гудман С., Хидетниеми С. Ведение в разработку и анализ алгоритмов. / – М.: Мир, 1981. – 368 с.
27. Кормен Т. [и др.] Алгоритмы: построение и анализ. /; 3-е издание: Пер. с англ. — М.: Издательский дом "Вильямс", 2014. — 1328 с. :ил. – Парал. тит. англ. ISBN 978-5-8459-

1794-2 (рус.).

28. Сигал И.Х., Бабинская Я.Л., Посыпкин М.А. Параллельная реализация метода ветвей и границ в задаче коммивояжера на базе библиотеки BNB-Solver // Труды ИСА РАН. – 2006. – Т. 25. – С. 26-36.
29. Ульянов М.В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ. / М. : ФИЗМАТЛИТ, 2008 – 304 с. ISBN 978-5-9221-0950-5.
30. Фомичёв М.И.: Особенности классического метода ветвей и границ для задачи коммивояжёра / Международная научно-техническая конференция «Информационные системы и технологии» ИСТ-2016, стр. 348, 2016. – 467 с.