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

Кафедра Прикладная математика и информатика
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Программная реализация модифицированного метода динамического программирования для решения задачи коммивояжера»

Обучающийся Т.К. Аминов (Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, Н.А. Сосина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент, С.А. Гудкова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Темой выпускной квалификационной работы является «Программная реализация модифицированного метода динамического программирования для решения задачи коммивояжера», выполненная бакалавром Тольяттинского государственного университета, кафедры «Прикладная математика и информатика», группы ПМИб–2002а, Аминовым Тимуром Константиновичем.

Объект работы: алгоритм решения задачи коммивояжера.

Предмет исследования: анализ эффективности применения модифицированного алгоритма динамического программирования для решения задачи коммивояжера на языке программирования Python.

Цель работы: реализовать программу на языке программирования для модифицированного метода ДП решения задачи коммивояжера, чтобы сравнить эффективность его работы с базовым методом. Чтобы достичь цели работы, необходимо решить следующие задачи:

1) рассмотреть методы решения задачи коммивояжера,

2) реализовать модифицированный алгоритм динамического программирования на языке Python,

3) выявить достоинства и недостатки реализуемого алгоритма, сделать сравнительный анализ, построить диаграммы.

Отчет состоит из введения, трех глав и заключения.

В первой главе представлены теоретические аспекты теории графов, описаны задача коммивояжера и алгоритмы, используемые для её решения. Во второй главе предлагается идея усовершенствования базового алгоритма ДП, представлен листинг программы на Python.

В третьей главе сделан сравнительный анализ для базового и модифицированного алгоритма.

ВКР представлена на 44 страницах, имеющих 22 иллюстрации, список используемых источников информации в количестве 20.

Abstract

The topic of the final qualifying work is "Software implementation of a modified dynamic programming method for solving the traveling salesman problem", performed by Timur Konstantinovich Aminov, Bachelor of Tolyatti State University, Department of Applied Mathematics and Computer Science, PMIb–2002a group.

The object of the work: an algorithm for solving the traveling salesman problem.

Subject of research: analysis of the effectiveness of using a modified dynamic programming algorithm to solve the traveling salesman problem in the Python programming language.

The purpose of the work: to implement a program in a programming language for a modified DP method for solving the traveling salesman problem in order to compare the effectiveness of its work with the basic method. To achieve the goal of the work, it is necessary to solve the following tasks:

1) consider the methods of solving the traveling salesman problem,

2) implement a modified dynamic programming algorithm in Python,

3) identify the advantages and disadvantages of the implemented algorithm, make a comparative analysis, and build diagrams.

The report consists of an introduction, three chapters and a conclusion.

The first chapter presents the theoretical aspects of graph theory, describes the traveling salesman problem and the algorithms used to solve it. In the second chapter, the idea of improving the basic DP algorithm is proposed, and a listing of the Python program is presented.

In the third chapter, a comparative analysis is made for the basic and modified algorithm. The WRC is presented on 44 pages with 22 illustrations, a list of 20 sources of information used.

Оглавление

Введение

Задача коммивояжера применяется в разных областях и сферах деятельности человека. В сфере транспортной логистики и доставки решение задачи коммивояжера способствует оптимизации маршрутов, что в конечном итоге приводит к снижению затрат работы и повышению общей производительности.

Также, задача коммивояжера оказывает влияние на область проектирования сетей связи, где оптимальное распределение ресурсов и минимизация времени передвижения имеют важное значение для обеспечения эффективной связности.

«Одним из основных подходов к решению этой задачи является метод динамического программирования. Его отличают следующие особенности: на основе принятия решения на предыдущих шагах на каждой итерации строится функция Беллмана» [1].

В выпускной квалификационной работе предложена идея модификации базового метода динамического программирования для эффективного решения задачи с большим количеством городов.

Мы сравниваем этот подход с базовыми методами решения задачи, а также исследуем влияние различных способов организации вычислительного процесса на производительность приложения.

Актуальность темы заключается в том, что точные методы для решения задачи коммивояжера не позволяют эффективно решать задачу при количестве городов, превышающем 25. Поэтому ставится вопрос разработки модификации метода ДП для ресурсоёмких задач, в число которых относится задача коммивояжера.

Объект работы: алгоритмы решения задачи коммивояжера.

Предмет исследования: сравнительный анализ реализаций модифицированного и базового метода динамического программирования на языке программирования Python.

Цель работы: реализовать усовершенствованный алгоритм решения задачи коммивояжера методом динамического программирования, сравнить его эффективность с базовым.

Для достижения цели поставленной работы необходимо решить следующие задачи:

1) рассмотреть точные методы решения задачи, в том числе базовый метод динамического программирования,

2) реализовать модифицированный метод ДП на языке программирования Python,

3) протестировать написанную программу на разных входных данных,

4) выявить достоинства и недостатки усовершенствованного метода, провести сравнительный анализ.

Глава 1 Постановка задачи и методы оптимизации

1.1 Общая характеристика задачи коммивояжера

1.1.1 Основные понятия

Чтобы сформулировать задачу коммивояжера, нужно обратиться к некоторым терминам из теории графов.

Простой граф – это математический объект, представляющий собой совокупность вершин и рёбер, где каждое ребро соединяет две вершины и не имеет повторяющихся рёбер или петель (рёбер, соединяющих вершину с самой собой). В простом графе также отсутствует ориентация рёбер, то есть они не имеют направления.

На рисунке 1 продемонстрирован простой граф.

Рисунок 1 – Простой граф

Мультиграф – простой граф, допускающий присутствие кратных рёбер. Иначе говоря, это граф, у которого вершины соединяются больше, чем одним ребром. Пример мультиграфа показан на рисунке 2.

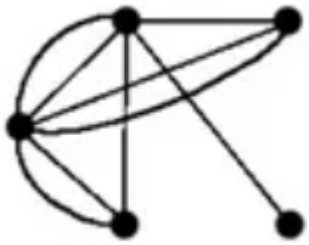

Рисунок 2 – Мультиграф

Псевдограф – это мультиграф, допускающий существование петель. Петля – это ребро, которое соединяет вершину саму с собой. Псевдограф представлен на рисунке 3.

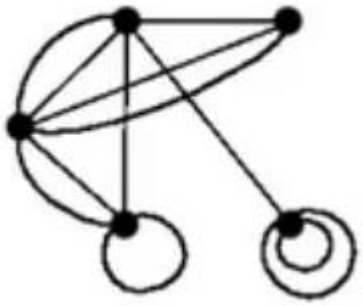

Рисунок 3 – Псевдограф

Ориентированный граф – это граф, рёбра которого имеют направление. Пример ориентированного графа представлен на рисунке 4.

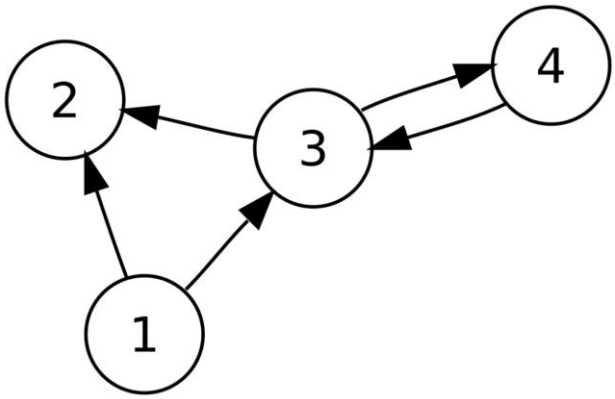

Рисунок 4 – Ориентированный граф

Взвешенный граф – это граф, у которого рёбра имеют веса (стоимость маршрута). Пример такого графа показан на рисунке 5.

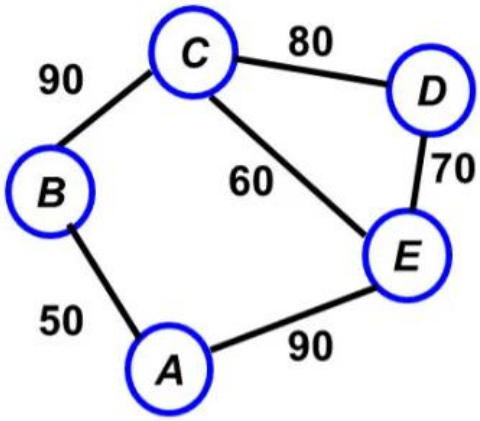

Рисунок 5 – Взвешенный граф

«Цикл Гамильтона – цикл, проходящий через все вершины графа по одному разу. Другими словами – это простой цикл, в который входят все вершины графа» [4].

«Гамильтонов граф – это граф, содержащий гамильтонов цикл» [2]. Пример такого графа представлен на рисунке 6.

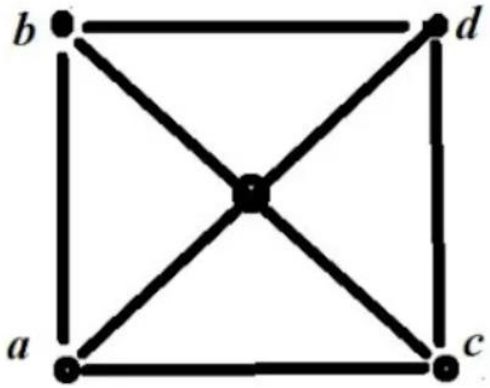

Рисунок 6 – Гамильтонов граф

Два ребра являются смежными, если они имеют одну общую вершину.

Два ребра называются кратными, если они соединяют одну и ту же пару вершин.

Если два графа имеют одинаковое количество вершин, то они называются изоморфными графами. Пример таких графов продемонстрирован на рисунке 7.

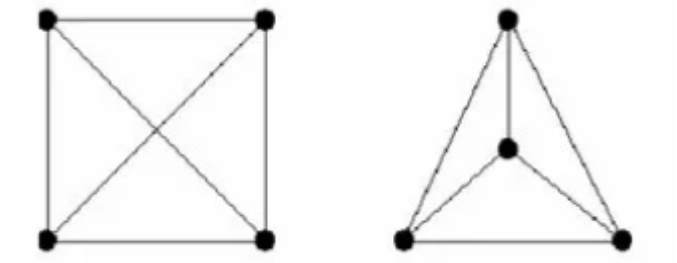

Рисунок 7 – Изоморфные графы

Если степень всех вершин одинакова и равна p, то граф называется регулярным степени p. На рисунке 8 продемонстрирован данный граф.

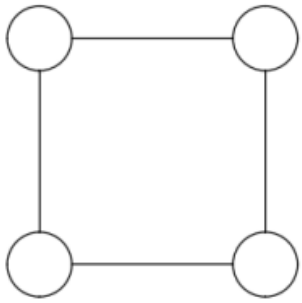

Рисунок 8 – Регулярный граф степени 2

Граф G_0 (V_0, E_0) называется подграфом графа G (V, E) (обозначается $G_0 \subseteq G$), если $V_0 \subseteq V$ и $E_0 \subseteq E$.

1.1.2. Алгоритм Дейкстры

Чтобы решить задачу коммивояжера методом динамического программирования, нужно понимать принцип работы алгоритма Дейкстры.

«Алгоритм Дейкстры – это алгоритм поиска кратчайших путей между узлами в графе. Для заданного исходного узла в графе алгоритм находит кратчайший путь между этим узлом и всеми остальными узлами. Он также может быть использован для поиска кратчайших путей от одного узла к одному узлу назначения путем остановки алгоритма после определения самого быстрого маршрута к узлу назначения» [6].

«Алгоритм Дейкстры основан на принципе релаксации, при котором более точные значения постепенно заменяют приближение к правильному расстоянию, пока не будет достигнуто кратчайшее расстояние» [2]. Примерное расстояние от вершины до вершины всегда завышено по сравнению с исходным расстоянием и его заменяют минимальным из его начального значения длиной снова определённого пути. Используется приоритетная очередь для жадного выбора ближайшей, ещё не обработанной вершины и выполняет этот процесс релаксации для ее исходящих ребер.

Блок-схема алгоритма Дейкстры представлена на рисунке 9.

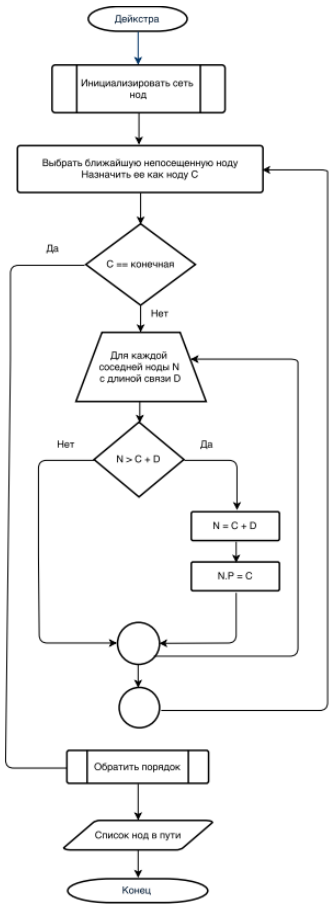

Рисунок 9 – Блок–схема алгоритма Дейкстры

Принцип действия алгоритма на конкретной задаче продемонстрирован на рисунках 10 – 16.

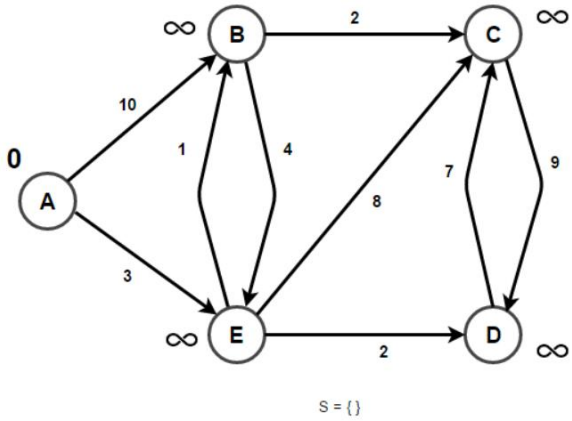

Рисунок 10 – Алгоритм Дейкстры (шаг 0)

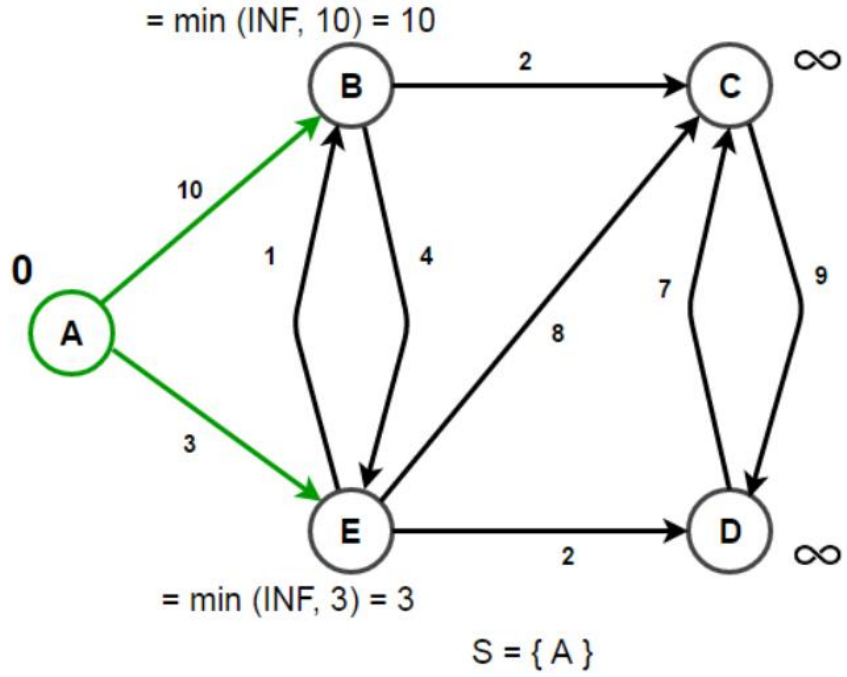

Рисунок 11 – Алгоритм Дейкстры (шаг 1)

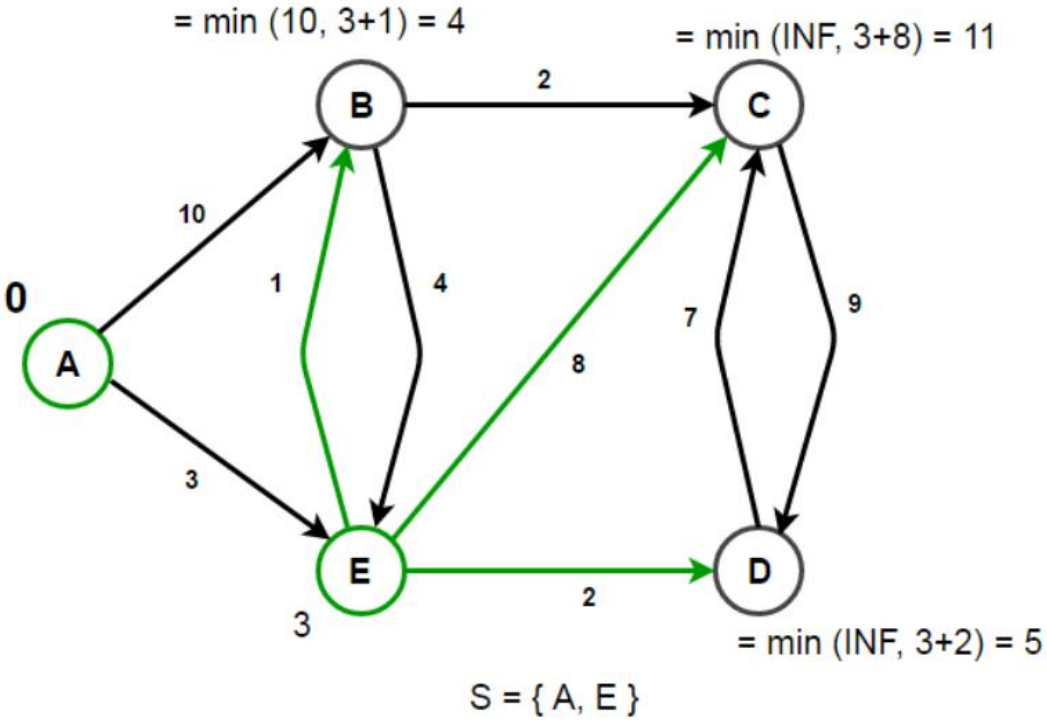

Рисунок 12 – Алгоритм Дейкстры (шаг 2)

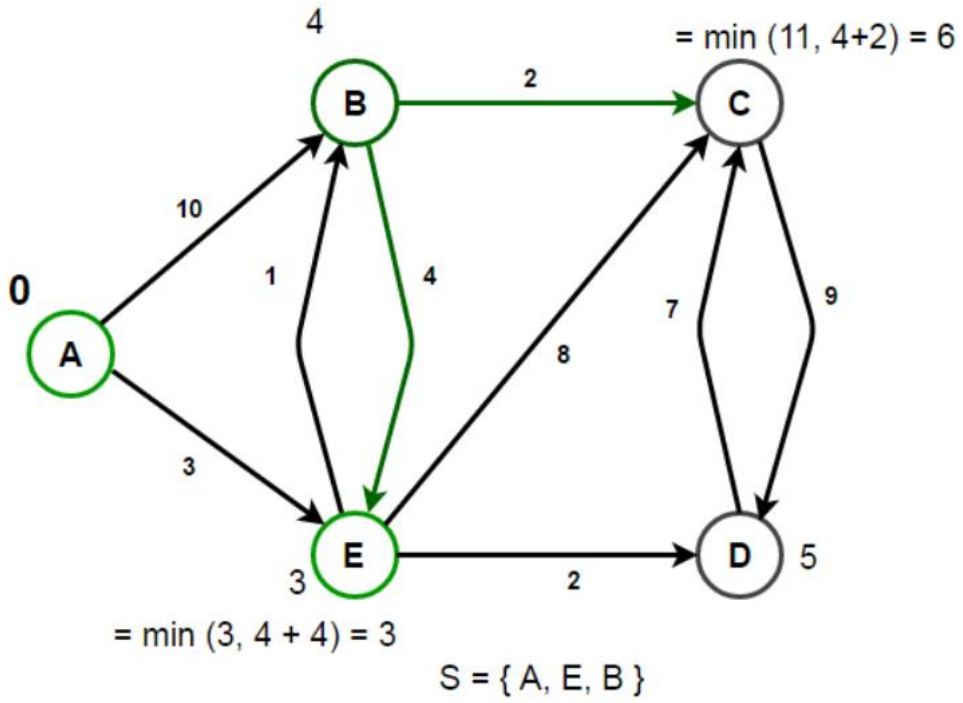

Рисунок 13 – Алгоритм Дейкстры (шаг 3)

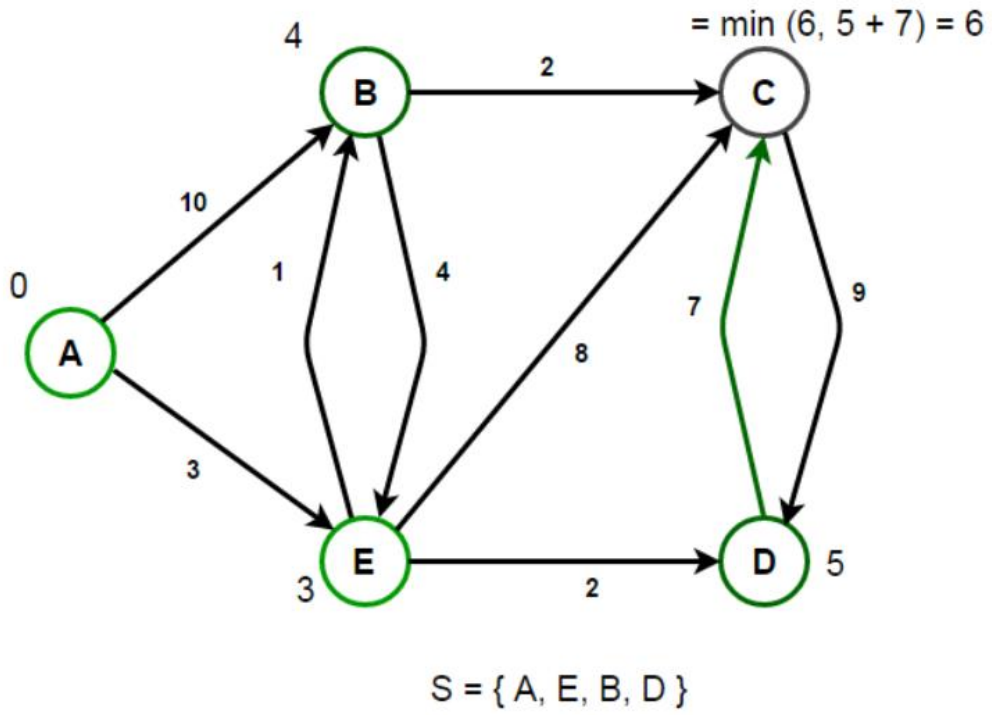

Рисунок 14 – Алгоритм Дейкстры (шаг 4)

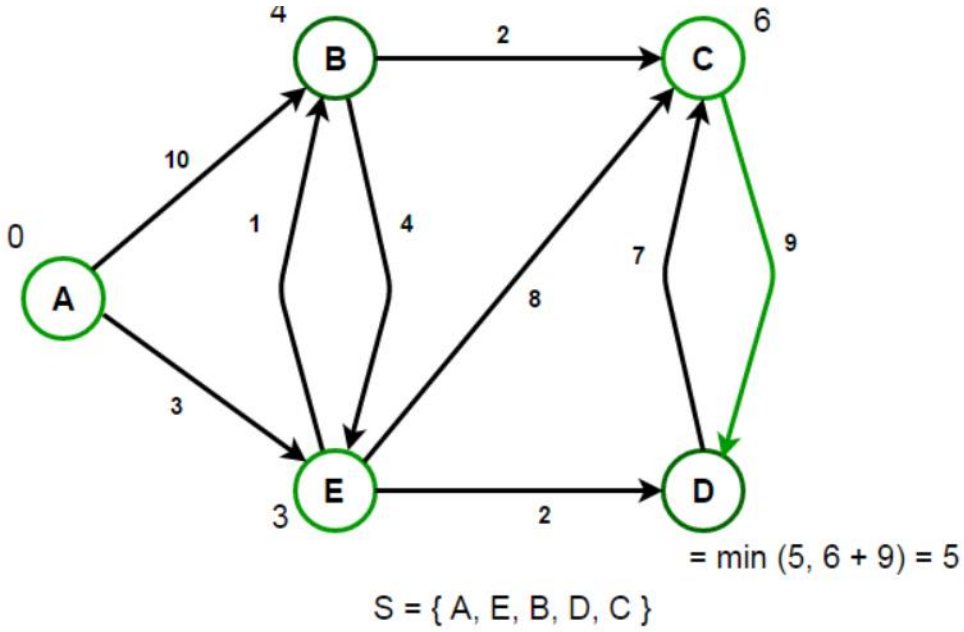

Рисунок 15 – Алгоритм Дейкстры (шаг 5)

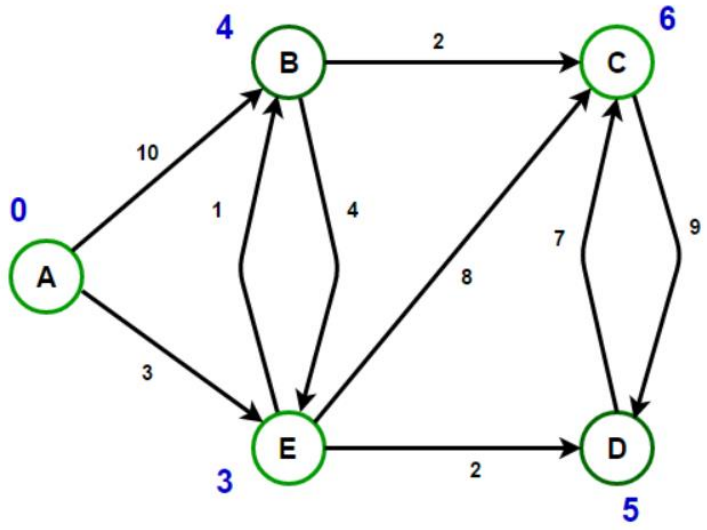

Рисунок 16 – Алгоритм Дейкстры (последний шаг)

Как видно из определения данного алгоритма, метод динамического программирования является частным случаем алгоритма Дейкстры. Требуется добавить лишь ограничения на то, что начальная и конечная точка маршрута должны совпадать и город не может быть посещён более одного раза.

1.1.3 Постановка задачи коммивояжера

«Задача коммивояжера представляет собой ключевой элемент в области комбинаторной оптимизации, обретая актуальность в различных областях современной деятельности» [1].

«Суть задачи состоит в том, что коммивояжер, выезжая из начального города n, посещает все данные города и возвращается в исходный город. Требуется найти такой маршрут, который дает наименьшую стоимость объезда городов. Также каждый город должен быть посещён ровно один раз. Другими словами, требуется найти гамильтонов цикл наименьшей длины» [8].

«Задача коммивояжера является трансвычислительной задачей – это означает, что при наличии более 66 пунктов в маршруте обхода, она методом перебора вариантов может решаться с помощью даже современной вычислительной техники очень большое время (около нескольких миллиардов лет)» [6].

«Задачу коммивояжера так же можно сформулировать, используя теорию графов. Задан ненаправленный граф и цена для каждого ребра этого графа. Необходимо найти Гамильтонов цикл с минимальной общей стоимостью» [9].

«Идеи, относящиеся к задаче коммивояжера, появились уже очень давно. В 1736 году Леонард Эйлер занимался проблемой нахождения кольцевого маршрута через семь мостов Кенигсберга. В 1832 была выпущена книга с названием «Коммивояжёр – как он должен вести себя и что должен делать для того, чтобы доставлять товар и иметь успех в своих делах – советы старого 8 курьера», которая включала в себя примеры маршрутов. В 1850-ых годах, Уильям Гамильтон изучал гамильтоновы циклы на графах. Он также выпустил игру с названием «Икосиан», целью которой было пройти все вершины додекаэдра ровно один раз от вершины к соседней, и вернуться в начало. Впервые, в качестве математической задачи, задача коммивояжера была предложена в 1930–ом году Карлом Менгером. Позже Хасслер Уитни предложил известное на данный момент название задачи странствующего торговца» [2].

Существует два вида задач. Ассиметричная задача предполагает, что расстояния от города i до города j не равно расстоянию от j до i. Другими словами, модель представляет из себя ориентированный взвешенный граф. За расстояние в данной задаче понимается стоимость перевозки из одного города в другой. Пример матрицы смежности для такой задачи продемонстрирован на рисунке 17.

	1	2	3	4	5
1	∞	25	40	31	27
2	5	∞	17	30	25
3	19	15	∞	6	1
4	9	50	24	∞	6
5	22	8	7	10	∞

Рисунок 17 – Матрица смежности асимметричной задачи коммивояжера

Симметричную задачу, в свою очередь, отличает равенство стоимости передвижения из города А в В и обратно. Матрица смежности в таких задачах симметрична, модель представляет из себя неориентированный взвешенный граф.

Впоследствии будет рассматриваться пример решения конкретно данной задачи.

1.1.4 Актуальность задачи коммивояжера

Задача становится особенно значимой в контексте управления логистикой, где эффективное планирование маршрутов приводит к существенному сокращению времени и ресурсов. В сфере транспортной логистики и доставки решение задачи коммивояжера способствует

оптимизации маршрутов, что в конечном итоге приводит к снижению затрат и повышению общей производительности.

Также, задача коммивояжера оказывает влияние на область проектирования сетей связи, где оптимальное распределение ресурсов и минимизация времени передвижения имеют важное значение для обеспечения эффективной связности.

В сфере туризма и гостеприимства решение этой задачи может привести к более эффективному обслуживанию клиентов, оптимизации туристических маршрутов и созданию более комфортного опыта для путешественников.

Таким образом, задача коммивояжера не только представляет теоретический интерес, но и имеет практическое применение, существенно влияя на различные аспекты современного бизнеса и технологического развития.

В современном обществе, где данные играют ключевую роль, задача приобретает особое значение. «Алгоритмы решения этой проблемы в конечном итоге становятся основой для машинного обучения, что открывает новые возможности в области автоматизации подходов к выборке и оптимизации» [7].

Эффективное планирование маршрутов не только сокращает расходы на топливо и уменьшает выбросы вредных веществ, но и способствует созданию более экологичных и эффективных транспортных сетей.

Таким образом, задача коммивояжера оказывает влияние на различные аспекты общества, от бизнеса и технологий до экологии и управления данными.

1.1.5 Математическая модель задачи коммивояжера

Введём обозначения. Пусть $x_{ij} = 1$, если коммивояжер передвигается из города i в город j и $x_{ij} = 0$, если j–й город не посещается. Чтобы избежать ситуаций, когда коммивояжер попадает в замкнутый цикл, введем

дополнительные ограничения, которые связывают переменные x_{ij} и переменные u_i.

$u_1 = 0, u_{n+1} =$ n – переменные, равные номеру посещения города на пути (фиктивные переменные).

Тогда математическая модель задачи коммивояжера будет выглядеть следующим образом:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}, \quad (1)$$

при ограничениях:

$$\sum_{i=1}^{n} x_{ij} = 1, 1 \leq j \leq n, \quad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1, 1 \leq i \leq n, \quad (3)$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1, \qquad S \neq \emptyset, S \subset \{1, \dots, n\}, \quad (4)$$

$$x_{ij} \in \{0,1\}, 1 \leq i, j \leq n. \quad (5)$$

В данном подпункте продемонстрирована математическая модель задачи коммивояжера.

1.2 Методы решения задачи коммивояжера

1.2.1 Метод полного перебора

«Полный перебор (или метод «грубой силы») – метод решения задачи путем перебора всех возможных вариантов» [1].

«Алгоритм полного перебора относится к классу методов поиска решения исчерпыванием всевозможных вариантов» [3]. Сложность данного

алгоритма зависит от количества городов в исходных данных задачи и составляет:

$$O(n!). \quad (6)$$

Данный алгоритм не эффективен, так как при большом количестве городов метод имеет огромные затраты по времени и требует высоких вычислительных возможностей компьютера.

1.2.2 Метод ветвей и границ

«Смысл метода «ветвей и границ» состоит, однако, в том, чтобы не просматривать все допустимые решения, а отсекать большинство ветвей на возможно более раннем этапе» [2].

Сложность данного алгоритма зависит от количества городов в исходных данных задачи и составляет:

$$0(n * \log_2 n). \quad (7)$$

Этот метод ориентирован на оптимизацию. «Цель алгоритма ветвей и границ – это найти нужную конфигурацию, на которой функция стоимости достигает своего максимального или минимального значения» [16].

Метод ветвления и границ так же не стабилен по времени. Если же применять отсечение маловероятных маршрутов, то результат будет очень далёк от точного. Поэтому было принято решение остановиться на методе ДП.

1.2.3 Метод динамического программирования

Рассмотренные выше алгоритмы являются методами для точного решения задачи коммивояжера. Они эффективны лишь при малом количестве городов. Подсчёты на компьютере для задачи даже с 26 городами займут несколько лет. На практике их количество достигает 30 и более городов, и тогда приходится применять приближённые методы, которые уступают по точности,

но выигрывают по быстродействию. Одним из таких подходов является динамическое программирование.

«Как известно, идея динамического программирования, опирающаяся на принцип оптимальности, состоит в том, что процесс оптимизации функционала рассматривается как последовательный n–шаговый процесс принятия решений, на каждом шаге которого принимается оптимальное решение. Для этого вычисляется оптимальное значение функционала» [11].

Особенности математической модели динамического программирования следующие:

1) целевая функция обладает свойством аддитивности, то есть она равна сумме функций на каждом шаге,

2) состояние системы зависит только от решения на предыдущем шаге оптимизации и не влияет на последующие шаги,

3) управление на каждых шагах зависимо от конечного количества управляющих переменных.

Для задач, которые решаются методом ДП, строится функция Беллмана. Для каждой конкретной задачи она уникальна и не может быть использована в других задачах.

Чтобы применить метод динамического программирования, задача должна обладать следующими свойствами:

1) задача определена для любого количества шагов, структура такой задачи не должна зависеть от их числа,

2) оптимальное решение на каждом шаге не влияет на решения, выбранные на предыдущем шаге,

3) задача интерпретируется как многошаговы процесс принятия решений.

Задача коммивояжера обладает всеми вышеперечисленными свойствами и может быть решена методом ДП.

«Представим метод динамического программирования в контексте задачи коммивояжера как последовательный процесс принятия решений на

каждом этапе маршрута. На каждом шаге коммивояжер выбирает оптимальный маршрут для объезда оставшихся городов, учитывая свое текущее местоположение. Допустим, коммивояжер достиг города i и перед возвращением в город n должен посетить k других городов: $j_1, j_2, \ldots, j_k$. Обозначим оптимальный путь от города i до города n через $B(n; j_1, j_2, \ldots, j_k)$. Здесь $B(n; j_1, j_2, \ldots, j_k)$ – функция Беллмана, которая определяет наименьшую стоимость маршрута» [5].

1.2.4 Пример решения задачи коммивояжера методом динамического программирования

Работу алгоритма представим на примере задачи с пятью городами, для которого матрица стоимости переездов из города в город имеет вид:

$$C = \begin{bmatrix} \infty & 25 & 40 & 31 & 27 \\ 5 & \infty & 17 & 30 & 25 \\ 19 & 15 & \infty & 6 & 1 \\ 9 & 50 & 24 & \infty & 6 \\ 22 & 8 & 7 & 10 & \infty \end{bmatrix}$$

«Согласно концепции Р. Беллмана, минимальная стоимость объезда городов будет найдена, если на последнем (пятом) шаге будет вычислено значение $B(5; j_1, j_2, j_3, j_4)$, т. е. определено значение наикратчайшего пути из города 5. Из этого города коммивояжер может выбрать путь в города 1, 2, 3, 4, а затем из каждого вернуться в исходный город. Для четвертого шага принятия решений, рассматривая процедуру в обратном порядке (от пятого к первому шагу), необходимо определить функции Беллмана для городов (состояний) 1, 2, 3, 4. Исходя из того, что коммивояжер из города 1 должен посетить города j_2, j_3, j_4 , из города 2 – города j_1, j_3, j_4 , из города 3 – города j_1, j_2, j_4 и из города 4 – города j_1, j_2, j_3, функции Беллмана для этих состояний будут выглядеть как $B(1; j_2, j_3, j_4)$, $B(2; j_1, j_3, j_4)$, $B(3; j_1, j_2, j_4)$, $B(4; j_1, j_2, j_3)$» [15]. Так как $B(5; j_1, j_2, j_3, j_4)$ представляет наименьшую стоимость, то выражение для нахождения значения функции Беллмана на шаге 5 представимо формулой:

$$B(5; j_1, j_2, j_3, j_4) = \{[c_{51} + B(1; j_2, j_3, j_4)], [c_{52} + B(2; j_1, j_3, j_4)], [c_{53} + B(3; j_1, j_2, j_4)], [c_{54} + B(4; j_1, j_2, j_3)]\}. \quad (8)$$

В формуле 8 c_{51}, c_{52}, c_{53}, c_{54} эквивалентны стоимости перемещений из города 5 в города 1, 2, 3, 4, которые берутся из исходной матрицы.

«Значения функций Беллмана $B(1; j_2, j_3, j_4)$, $B(2; j_1, j_3, j_4)$, $B(3; j_1, j_2, j_4)$, $B(4; j_1, j_2, j_3)$, которые должны вычисляться на четвёртом шаге принятия решений, учитывают, что коммивояжер, начиная свой маршрут из городов 1, 2, 3, 4, должен посетить три оставшихся города» [17]. Значения функции Беллмана на 4 шаге определяются формулами (2), с учетом стоимостей перемещений:

$$B(1; j_2, j_3, j_4) = \min = \{[c_{12} + B(2; j_3, j_4)], [c_{13} + B(3; j_2, j_4)], [c_{14} + B(4; j_2, j_3)]\}, (9)$$

$$B(2; j_1, j_3, j_4) = \min = \{[c_{21} + B(1; j_3, j_4)], [c_{23} + B(3; j_1, j_4)], [c_{24} + B(4; j_1, j_3)]\}, (10)$$

$$B(3; j_1, j_2, j_4) = \min = \{[c_{31} + B(1; j_2, j_4)], [c_{32} + B(2; j_1, j_4)], [c_{34} + B(4; j_1, j_2)]\}, (11)$$

$$B(4; j_1, j_2, j_3) = \min\{[c_{41} + B(1; j_2, j_3)], [c_{42} + B(2; j_1, j_3)], [c_{43} + B(3; j_1, j_2)]\}. \quad (12)$$

В формулах 9 – 12 учитывается стоимость перемещения из города 5 в каждый из городов 1, 2, 3, 4, добавленная к соответствующим значениям функций Беллмана, вычисленным на пятом шаге процесса принятия решений.

Очевидно, что вычисление по формулам 9 – 12 возможно в предположении, что на третьем шаге принятия решений мы уже определили значения функций Беллмана. В данном контексте, записи функций указывают,

что, «например, для города 1 они должны быть рассчитаны с учетом различных вариантов посещения городов коммивояжером в последовательности 2, 3 или 3, 2; городов 2, 4 или 4, 2; а также городов 3, 4 или 4, 3» [18]. Оптимальный маршрут для каждого варианта может быть представлено соответствующим выражением:

$$B(1; j_2, j_3) = \min\{[c_{12} + B(2; j_3)], [c_{13} + B(3; j_2)], \quad (13)$$

$$B(1; j_2, j_4) = \min\{[c_{12} + B(2; j_4)], [c_{14} + B(4; j_2)], \quad (14)$$

$$B(1; j_3, j_4) = \min\{[c_{13} + B(3; j_4)], [c_{14} + B(4; j_3)]. \quad (15)$$

В формулах 13 – 15 используется функция минимума, чтобы выбрать оптимальный маршрут для каждой из возможных последовательностей посещения городов.

Аналогично получим формулы 16 – 24, которые определяют значения функций Беллмана для состояний с номерами 2, 3 и 4. Таким образом получаем следующие формулы для состояний системы:

$$B(2; j_1, j_3) = \min\{[c_{21} + B(1, j_3)], [c_{23} + B(3, j_1)]\}, \quad (16)$$

$$B(2; j_1, j_4) = \min\{[c_{21} + B(1, j_4)], [c_{24} + B(4, j_1)]\}, \quad (17)$$

$$B(2; j_3, j_4) = \min\{[c_{23} + B(3, j_4)], [c_{24} + B(4, j_3)]\}, \quad (18)$$

$$B(3; j_1, j_2) = \min\{[c_{31} + B(1, j_2)], [c_{32} + B(2, j_1)]\}, \quad (19)$$

$$B(3; j_1, j_4) = \min\{[c_{31} + B(1, j_4)], [c_{34} + B(4, j_1)]\}, \quad (20)$$

$$B(3; j_2, j_4) = \min\{[c_{32} + B(2, j_4)], [c_{34} + B(4, j_2)]\}, \quad (21)$$

$$B(4;\ j_1, j_2\ ) = \min\{[c_{41} + B(1, j_2)], [c_{42} + B(2, j_1)]\}, \quad (22)$$

$$B(4;\ j_1, j_3\ ) = \min\{[c_{41} + B(1, j_3)], [c_{43} + B(3, j_1)]\}, \quad (23)$$

$$B(4;\ j_2, j_3\ ) = \min\{[c_{42} + B(2, j_3)], [c_{43} + B(3, j_2)]\}. \quad (24)$$

Выше представленные функции можно найти, если известны $B(1;\ j_2\ )$, $B(1;\ j_3\ )$, $B(1;\ j_4\ )$, $B(2;\ j_1\ )$, $B(2;\ j_3\ )$, $B(2;\ j_4\ )$, $B(3;\ j_1\ )$, $B(3;\ j_2\ )$, $B(3;\ j_4\ )$, $B(4;\ j_1\ )$, $B(4;\ j_2\ )$, $B(4;\ j_3\ )$ состояния. Поэтому составим для них функции Беллмана:

$$B(1;\ j_2\ ) = c_{12} + B(2;\ j_5\ ), \quad (25)$$

$$B(1;\ j_3\ ) = c_{13} + B(3;\ j_5\ ), \quad (26)$$

$$B(1;\ j_4\ ) = c_{14} + B(4;\ j_5\ ), \quad (27)$$

$$B(2;\ j_1\ ) = c_{21} + B(1;\ j_5\ ), \quad (28)$$

$$B(2;\ j_3\ ) = c_{12} + B(2;\ j_5\ ), \quad (29)$$

$$B(2;\ j_4\ ) = c_{24} + B(4;\ j_5\ ), \quad (30)$$

$$B(3;\ j_1\ ) = c_{31} + B(1;\ j_5\ ), \quad (31)$$

$$B(3;\ j_2\ ) = c_{32} + B(2;\ j_5\ ), \quad (32)$$

$$B(3;\ j_4\ ) = c_{34} + B(4;\ j_5\ ), \quad (33)$$

$$B(4;\ j_1\ ) = c_{41} + B(1;\ j_5\ ), \tag{34}$$

$$B(4;\ j_2\ ) = c_{42} + B(2;\ j_5\ ), \tag{35}$$

$$B(4;\ j_3\ ) = c_{43} + B(3;\ j_5\ ). \tag{36}$$

«В связи с тем, что функции $B(1;\ j_5\ )$, $B(2;\ j_5\ ), B(3;\ j_5\ ), B(4;\ j_5\ )$ представляют стоимость перемещений c_{15}, c_{25}, c_{35}, c_{45}, извлеченных из матрицы С, обратный процесс принятия решений может быть инициирован с первого шага» [19].

«Исходя из данных первой строки и уравнений 8 и 9, получаем значения функций Беллмана для второй строки. Затем, используя уравнения 10 и 11, вычисляем значения функций для третьей строки и так далее. Процесс вычислений и заполнения строк таблицы завершается получением значения функции Беллмана для пятого шага» [12]. Итак, наименьшая стоимость объезда всех исходных городов равна 62.

«Чтобы определить маршрут, соответствующий найденной стоимости, следуем следующим образом. Из $B(5; j_1\ , j_2\ , j_3\ , j_4)$ находим элемент, минимизирующий это выражение» [20]. Таким элементом значится c_{54}+$B(4; j_1\ , j_2\ , j_3\ )$ = 10 + 52 = 62, что указывает на то, что коммивояжер должен посетить город 4, покидая город 5 (индексы компоненты c_{54}). Чтобы определить следующий город, к которому он направится из города 4, находим элемент, минимизирующий функцию $B(4; j_1\ , j_2\ , j_3\ )$. Этим элементом является c_{41}+$B(1;\ j_2\ , j_3\ )$ = 9 + 43 = 52 , что указывает на то, что, покидая город 4, коммивояжер должен посетить город 1. Минимум для $B(1;\ j_2\ , j_3\ )$ достигается элементом c_{12}+$B(2;\ j_3\ ) = 25 + 18 = 43$, что говорит о том, что из города 1 коммивояжер направляется в город под номером 2. Далее, минимум для $B(2;\ j_3\ )$ равен c_{23}+$B(3;\ j_5\ )$ = 17+1 = 18. Таким образом, покидая город 2, коммивояжер движется в город 3, а затем возвращается в город 5.

Итоговый оптимальный маршрут объезда городов: 5, 4, 1, 2, 3, 5, представлен последовательностью дуг (5, 4), (4, 1), (1, 2), (2, 3), (3, 5).

Суммируя вышеизложенное, следует отметить, что «алгоритм решения задачи коммивояжера методом динамического программирования, аналогично многим случаям использования данного метода, включает в себя два этапа: вычисление оптимального значения функционала и на его основе определение последовательности оптимальных решений» [10].

Сформулируем выводы к первой главе.

В первой главе выпускной квалификационной работы представлена постановка задачи коммивояжера. Отмечена её актуальность в настоящее время, построена математическая модель.

Рассмотрены основные моменты из теории графов для понимания подходов к решению задачи. Продемонстрирован алгоритм Дейкстры по нахождению наименьшего пути. Выявлено, что подход ДП – частный случай алгоритма Дейкстры. Обозначены уже существующие методы решения задачи и выбран приближенный метод, называемый динамическим программированием, как самый эффективный по времени работы алгоритм, позволяющий за короткие сроки решать задачу с имеющимися 30 городами в условии.

Приведён пример решения задачи с матрицей смежности размерностью 5x5. Данная задача будет тестовой для проверки работоспособности усовершенствованного метода ДП.

Глава 2 Реализация модифицированного метода динамического программирования для решения задачи коммивояжера

2.1 Идея модифицированного метода динамического программирования

Под словом «модифицированный» понимается постановка ограничений на количество итераций, на которое разрешается возвращаться при программной реализации метода. Базовый метод динамического программирования предполагает, что если на k–й итерации невозможно построить маршрут на k–1 итерации, то метод рассматривает k–2, k–3, k–4... Такие действия занимают много времени и требуют дополнительной памяти на компьютере.

«Предположим, что на итерации k использование маршрутов итерации k–2 возможно. Тогда построение начинается с анализа маршрутов, найденных на итерации k–1. Для каждого пункта n рассматривается предыдущий маршрут. На данном этапе может возникнуть необходимость построения альтернативных маршрутов на основании (k–2) –й итерации» [14].

Предположим, что «для пункта I данная необходимость возникла. Общее количество этих пунктов равно n–1. Для каждого такого пункта j (из числа указанных) целесообразно просмотреть n – 2 маршрута, так как остальные два из общего количества обязательно начинаются с пунктов i или j» [14].

Таким образом, «получаем маршруты (k – 2)–й итерации в количестве (n–2)*(n–1) . Всего таких маршрутов может оказаться (n–1). Как итог, для пункта i может быть просмотрено маршрутов в количестве $(n-2)*(n-1)^2$» [14].

«Количество пунктов, для которых требуется построить альтернативные маршруты, может быть различным для каждой i–й итерации. На начальных итерациях скорее всего не будет ни одного такого пункта, а на последних их

количество возрастёт» [14]. Для матрицы смежности тестовой задачи, которая представлена в главе 2, таких городов не имелось.

Пусть на каждой итерации требуется построение альтернативных маршрутов лишь для одного пункта. В таком случае верхняя граница для числа просмотренных на k–й итерации маршрутов покажется следующей формулой:

$$Q_{m(k)} \approx (n-2)(n-1)^2 + n(n-1). \quad (37)$$

А значит просмотр осуществится не более числа маршрутов, представляемого следующей формулой:

$$Q_m = (n-2)(n-1)^3 + n(n-1)^2 + n. \quad (37)$$

«Ограничив число итераций, мы повысим производительность программы, но снизим точность вычисления. Для повышения точности было решено увеличить количество просмотров вариантов на небольшом количестве итераций» [14]. Данные модификации сделаны конкретно для задачи коммивояжера. Таким образом, модифицированный метод приведёт к более качественным решениям, чем стандартный.

2.2 Постановка требований

Поставлена задача программной реализации усовершенствованного метода ДП для решения задачи коммивояжера. Программа должна соответствовать поставленным требованиям:

1) программа должна запускаться и работать правильно на всех операционных системах,

2) должна иметься возможность удобного задания любого количества городов и расстояний между ними,

3) применяя заданные пользователем исходные данные для задачи, программа обязана корректно искать минимальный путь для обхода пунктов,

4) в качестве результата работы программы должен быть выведенный на экран оптимальный маршрут и последовательность городов, которые были посещены коммивояжером.

2.3 Выбор технических средств

«Язык программирования – формальный язык, указывающий на набор инструкций, которые используются для производства разных видов продукции. Языки программирования обычно состоят из инструкций для компьютера» [18].

«Выбранный язык программирования может в значительной степени повлиять как на сам процесс работы программы, так и на её результат. Мною был выбран язык программирования Python. Он имеет ряд встроенных библиотек, которые будут использованы в алгоритме решения задачи коммивояжера» [17].

«Python – это интерпретируемый высокоуровневый язык программирования общего назначения. Язык был создан Гвидо ван Россумом и впервые выпущен в 1991 году. Философия Python подчеркивает удобочитаемость кода с его заметным использованием отступов и пробелов. Его языковые конструкции и объектно-ориентированный подход нацелены на то, чтобы помочь программистам написать понятный, логичный код для малых и крупных проектов» [13].

«Python обладает динамической типизацией и сборщиком мусора. Он поддерживает несколько парадигм программирования, включая процедурное, объектно-ориентированное и функциональное программирование. Python часто описывается как язык «с батарейками» из–за обширного количества стандартных библиотек» [17].

Среда разработки, в которой будет написана программа, PyCharm, так как он работает на операционных системах Windows, Mac OS X и Linux и имеет удобный интерфейс.

«PyCharm – интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA» [20].

«Основными возможностями PyCharm являются статический анализ кода, подсветка синтаксиса и ошибок, навигация по проекту и исходному коду при помощи отображения файловой структуры проекта и быстрому переходу между файлами, классами, методами и использованиями методов» [20].

2.4 Реализация усовершенствованного метода

Для реализации модифицированного метода сначала следует написать программу, демонстрирующую базовый метод динамического программирования для решения задачи коммивояжера. Также это нужно для того, чтобы впоследствии сравнить эффективность работы двух программ и наглядно увидеть, как модификация в виде добавления ограничений на количество итераций и мемоизация влияют на точность вычислений и скорость работы программы.

Реализация базового метода динамического программирования продемонстрирована в приложении А.

Теперь рассмотрим код модифицированного метода.

На рисунке 18 представлен код, отвечающий за импорт использующихся в программе библиотек. Numpy предназначен для удобной работы с массивом данных. В первую очередь это начальные данные о количестве городов и расстояний между ними.

Переменная n отвечает за количество городов. Двумерный массив dist представляет собой матрицу смежности. При необходимости пользователю будет удобно менять исходные данные, так как они расположены в самом начале программы.

```
1. import numpy as np
2. import matplotlib as mp
3. n = 5
4. dist = [[0, 0, 0, 0, 0], [0, 0, 10, 15, 20], [
5.       0, 10, 0, 25, 25], [0, 15, 25, 0, 30], [0, 20, 25, 30, 0]]
```

Рисунок 18 – Исходные данные

На рисунке 19 представлена функция fun(), в которой реализована основная логика работы программы.

В функции единицей помечается город, уже посещённый коммивояжером. Изначально все города помечаются меткой 0. Присутствует мемоизация. «Мемоизация – это сохранение результатов выполнения функций для предотвращения повторных вычислений» [11].

Второй модификацией является добавление ограничения на количество итераций, к которым может обращаться программа при поиске оптимального маршрута.

Переменная res отвечает за результат, выбранный на основе нахождения минимального расстояния между городами. В конце присутствует вывод оптимального пути и последовательности посещения городов коммивояжером.

```
6. def fun(i, mask):
7.     if mask == ((1 << i) | 3):
8.             return dist[1][i]
9.     if memo[i][mask] != -1:
10.            return memo[i][mask]
11.    res = 10**8
12.    for j in range(1, n+1):
13.            if (mask & (1 << j)) != 0 and j != i and j != 1:
14.                    res = min(res, fun(j, mask & (~(1 << i))) + dist[j][i])
15.    memo[i][mask] = res # storing the minimum value
16.    return res
17. ans = 10**8
18. for i in range(1, n+1):
19.    ans = min(ans, fun(i, (1 << (n+1))-1) + dist[i][1])
20. print("The cost of most efficient tour = " + str(ans))
```

Рисунок 19 – Функция по нахождению оптимального маршрута

«Задача коммивояжера решается только при условии, если в транспортной сети имеется возможность построить маршрут между любыми двумя пунктами. Граф, построенный на основании такой транспортной сети, должен быть связным» [20].

«В качестве исходных данных вводятся длины путей между парами пунктов транспортной сети, для которых существует непосредственная транспортная связь. В случае, если транспортная сеть такова, что непосредственная транспортная связь существует не для всех пар ее пунктов, программа позволяет построить маршрут минимальной длины для каждой пары пунктов, не связанных между собой непосредственно. Расстояния между пунктами транспортной сети должны быть заданы целыми числами» [20].

Сформулируем выводы ко второй главе.

Предложена идея усовершенствования базового метода динамического программирования, чтобы была возможность решать задачу коммивояжера, путём добавления ограничения на итерации и добавления мемоизации. Заданы требования для реализуемой программы.

Был выбран язык программирования Python, как самый подходящий для данной задачи, на котором будет написана программа, выбрана среда разработки PyCharm.

Построен алгоритм работы программы на Python и реализована программа.

Продемонстрирован листинг программы на выбранном языке модифицированного метода ДП.

Глава 3 Анализ результатов работы программы

3.1 Результат работы программы

Для проверки работоспособности программы протестируем её на исходных данных, приведённых в главе 2.

Результат работы программы продемонстрирован на рисунке 20.

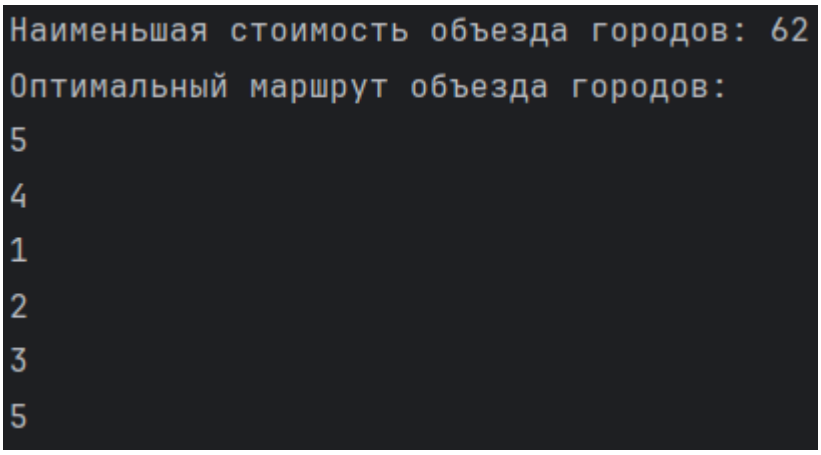

Рисунок 20 – Результат работы программы на тестовых данных

Глядя на результаты решения задачи, можно заявить, что программа корректно считает результаты и для проверки её эффективности введём данные, содержащие 15, 20, 25 и 30 городов.

3.2 Анализ эффективности

Для наглядности все результаты занесём в таблицу. Результаты решения задачи, используя метод динамического программирования без модификации представлены в таблице 1.

В качестве показателя эффективности скорости работы программы будем использовать время работы программы в секундах. Время будем

фиксировать при помощи встроенных библиотек языка программирования Python.

Таблица 1 – Метод динамического программирования

Количество городов n	Время работы программы, с	Минимальная стоимость маршрута
5	0.001	62
10	1.02	81
15	11	105
20	150	180
25	640	277
26	∞	не определена
27	∞	не определена
28	∞	не определена
29	∞	не определена
30	∞	не определена

Опираясь на данные, приведённые в таблице 1, можно сделать вывод, что стандартный подход метода динамического программирования для задачи коммивояжера хорошо показывает себя при количестве городов менее 25.

Если количество городов более 25, определить точное решение задачи методом ДП представляется невозможным за адекватное количество времени.

Теперь протестируем программу, реализующую модифицированный метод ДП. Для этого внесём те же самые данные и результаты запишем в таблицу 2.

Таблица 2 – Модифицированный метод динамического программирования

Количество городов n	Время работы программы, с	Минимальная стоимость маршрута
5	0.001	62
10	1.02	81
15	10	105
20	115	192
25	150	291
26	168	300
27	202	302
28	246	307
29	280	311
30	312	315

Опираясь на данные, представленные в таблице 2, можно сделать вывод, что модификация алгоритма путём введения ограничения на количество итераций, к которым может обращаться программа, стало возможным решить задачу, если количество городов в исходных данных превышает 25.

Можно заметить, что сократилось время работы программы при количестве городов, большем 10, однако снизилась точность вычислений. При количестве городов 10 и менее выигрыш во времени работы программы не получен, однако и точность вычислений не ухудшилась. Вычислим среднеквадратическую ошибку полученных вычислений от точного решения по формуле 39:

$$\boldsymbol{\sigma} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}, \tag{39}$$

Расчёты представлены в таблице 3.

Таблица 3 – Расчёт среднеквадратической ошибки метода

Количество городов, n	Метод ДП (точное решение), y_i	Метод ДП (приближённое решение), $\hat{y}_i$	$(\hat{y}_i - y_i)^2$
5	62	62	0
10	81	81	0
15	105	105	0
20	180	192	144
25	277	291	196
30	-	315	-
∑			340

Пользуясь формулой 37, получаем σ = 8,246, что означает, что в среднем ошибка не превышает 8 единиц, что является хорошим результатом.

Точность получаемого результата зависит также от задаваемого максимального количества шагов возврата, иными словами, от ограничения на количество итераций, к которым может обращаться программа. Данная зависимость занесена в таблицу 4.

Таблица 4 – Зависимость среднего относительного отклонения от количества шагов возврата

Максимальное количество шагов возврата, i	Среднее отклонение от оптимального результата, %
0	6,2
1	4,5
2	3,15
3	2,6
4	1,1
5	1,4
6	1,72

Опираясь на таблицу, делаем вывод, что задавать ограничение на количество шагов более 4 не имеет смысла. Во–первых, эффект повышения точности становится незначительным. Во–вторых, количество городов в исходных данных уменьшается.

Для наглядности на рисунке 21 продемонстрирована диаграмма, показывающая зависимость точности вычислений от числа городов.

На рисунке 22 показана диаграмма, демонстрирующая зависимость времени работы программы от числа городов.

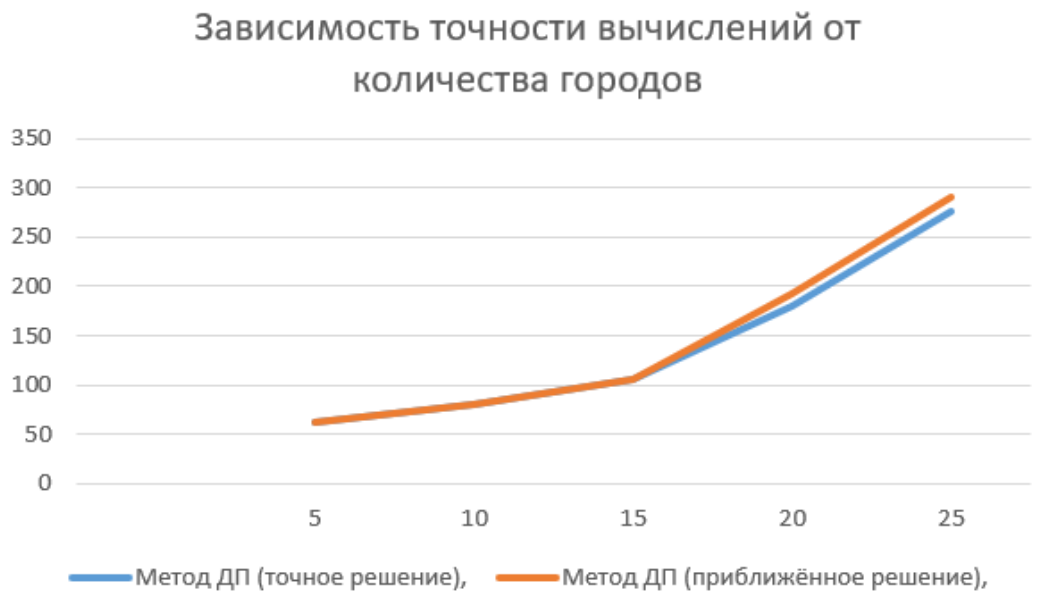

Рисунок 21 – Диаграмма зависимости точности вычислений от количества городов

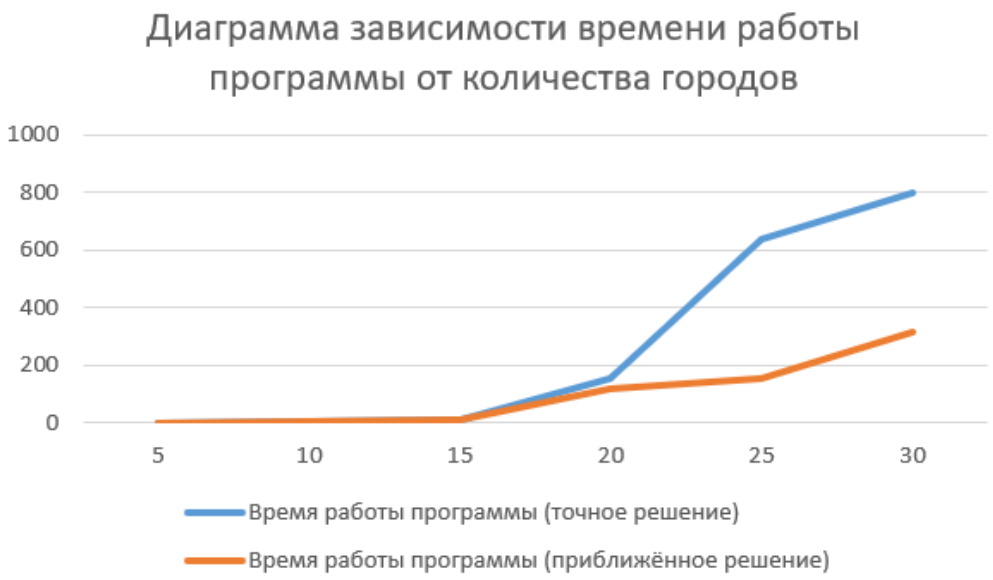

Рисунок 22 – Диаграмма зависимости времени работы программы от количества городов

Сформулируем выводы к третьей главе.

Была проверена работоспособность программы на тестовых данных, взятых из главы 1. Результаты, полученные теоретически и практически, совпали.

Проведён анализ эффективности, сравнены результаты, полученные модифицированным и базовым методом динамического программирования при различных начальных данных. Произведена наглядная демонстрация роста скорости работы программы на диаграмме. Программа показала незначительные отклонения по точности с ростом количества городов. Был достигнут большой выигрыш по времени работы.

Выявлено, что модификация по ограничению числа итераций для возврата наиболее эффективна при ограничении, равном четырём итерациям. В данном случае достигается наибольшее достижение быстродействия при незначительных отклонениях по точности.

Заключение

При выполнении выпускной квалификационной работы изучены теоретические аспекты теории графов, продемонстрирован алгоритм динамического программирования для решения задачи, как частный случай алгоритма Дейкстры. Описаны точные алгоритмы решения задачи коммивояжера, изучены достоинства и их недостатки. Актуальность разработки алгоритма динамического программирования для решения поставленной задачи была выявлена в связи с невозможностью эффективного решения задачи точными методами.

Описана общая постановка задачи коммивояжера. Отмечена её актуальность в настоящее время, построена математическая модель. Обозначены уже существующие методы решения задачи и выбран приближенный метод, называемый динамическим программированием, как самый эффективный по времени работы алгоритм, позволяющий за короткие сроки решать задачу с имеющимися 30 городами в условии.

Приведён пример решения задачи с матрицей смежности размерностью 5x5.

Представлена идея модификации базового метода динамического программирования для решения задачи коммивояжера путём добавления ограничения на итерации и добавления мемоизации. Заданы требования для реализуемой программы.

Был выбран язык программирования Python, на котором была написана программа. Разработка велась в среде PyCharm.

Продемонстрирован листинг программы на выбранном языке модифицированного метода ДП.

Произведена отладка программы на разных входных данных. Тестирование проходило на количестве городов от 5 до 30.

Базовый подход динамического программирования для решения задачи показал свою эффективность лишь на данных, число городов в которых было

25 или меньше. Модификация метода позволила рассчитать оптимальную стоимость маршрута при количестве городов, равном 30.

Проведён анализ эффективности, сравнены результаты, полученные модифицированным и базовым методом динамического программирования при различных начальных данных. Произведена наглядная демонстрация роста скорости работы программы на диаграмме. Программа показала незначительные отклонения по точности с ростом количества городов. Был достигнут большой выигрыш по времени работы.

Было выявлено, что модификация путём количества числа итераций, к которым может обращаться программа, эффективна при ограничении в 4 итерации. В таком случае достигается лучшая точность и самое высокое быстродействие.

Список используемой литературы и используемых источников

1. А. Н. Чаплыгин. Учимся программировать вместе с Питоном. шагу Учебник. — ревизия 226. — 135 с.

2. Беллман Р. Динамическое программирование. — М.: Репринт оригинального издания иностранной литературы, 1960 год, 2012.

3. Гладких Б.А. Методы оптимизации и исследования операций. Часть 1. Введение в исследование операций. Линейное программирование. — Томск: НТЛ, 2009. — 200 с.

4. Доусон М. Программируем на Python. — СПб.: Питер, 2012. — 432 с.

5. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Глава 15. Динамическое программирование // Алгоритмы: построение и анализ IntroductiontoAlgorithms / Под ред. И. В. Красикова. — 2-е изд. — М.: Вильямс, 2005. —1296 с.

6. Лежнев, А.В. Динамическое программирование в экономических задачах [Текст]: учеб. пособие / Лежнев А.В. - Москва: Бином, 2010. - 176 с.

7. Маккинли У. Python и анализ данных. — Перевод с английского. — М.: ДМК Пресс, 2015. — 482 с. —SanjoyDasgupta, ChristosH. Papadimitriou, UmeshVazirani. Algorithms = Algorithms. — 1-е изд. —McGraw-Hill Science/Engineering/Math, 2006. — С. 336.

8. Марк Лутц. Программирование на Python / Пер. с англ. — 4-е изд. — СПб.: Символ-Плюс, 2011. — Т. I. — 992 с.

9. Марк Саммерфилд. Python на практике. — Перевод с английского. — М.: ДМК Пресс, 2014. — 338 с.

10. Некрасова М.Г. Методы оптимизации и теория управления: учебное пособие / М.Г. Некрасова. Комсомольск-на-Амуре: КНАГТУ, 2007. − 132 с.

11. Окулов С.М. Динамическое программирование / С.М. Окулов, О.А. Пестов. Москва: БИНОМ. Лаборатория знаний, 2012. − 260 с.

12. Фёдоров Д. Ю. Основы программирования на примере языка Python / Учебное пособие. —СПб.:Юрайт, 2018. — 167 с.

13. Ширяв В.И. Исследование операций и численные методы оптимизации: учебное пособие / В.И. Ширяв. Москва: Ленанд, 2017. − 224 с.

14. Bertsekas, D. P. (2017), Dynamic Programming and Optimal Control (4th ed.), Athena Scientific.

15. Giegerich, R.; Meyer, C.; Steffen, P. (2004), "A Discipline of Dynamic Programming over Sequence Data" (PDF), Science of Computer Programming, 51 (3): 215–263, doi:10.1016/j.scico.2003.12.005

16. Hamilton, Naomi (5 August 2008). "TheA-ZofProgrammingLanguages: Python". Computerworld. Archivedfromtheoriginalon 29 December 2008. Retrieved 31 March 2010.

17. Lutz, Mark (2013). LearningPython (5thed.). O'ReillyMedia.

18. Mathsemestr [Электронный ресурс] Задачи ДП - Режим доступа: минальой https://math.semestr.ru/dinam/dinam_manual.php.

19. Meyn, Sean (2007), Control Techniques for Complex Networks, Cambridge University Press, archived from the original on 2010-06-19

20. Zhang, Y. Quality of Service Control in High-Speed Networks. / Chen, G., Morgan Kaufmann, 2016. 408 p.