МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

ИССЛЕДОВАНИЕ МЕТОДОВ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Батталовой Нур Алии Илдаровны

Научный руководитель:
Доцент кафедры ИиП _____ Е.В. Кудрина
подпись, дата

Зав. кафедрой:
Доцент, к.ф.-м.н. _____ М. В. Огнева
подпись, дата

Саратов 2019

ВВЕДЕНИЕ

Задача коммивояжера в теории дискретной оптимизации считается классической. Впервые она была сформулирована еще в 1759 году. Суть задачи состоит в том, чтобы найти кратчайший замкнутый путь обхода нескольких городов, заданных своими координатами. Города могут посещаться только единожды.

И в настоящее время задача коммивояжера не потеряла свою **актуальность**, так как применение ее на практике связано с эффективностью и оперативностью планирования перевозок, а от этого напрямую зависит доход предприятия.

Оптимизационная постановка задачи относится к классу NP-трудных задач, впрочем, как и большинство её частных случаев. Задача, в которой ставится вопрос, существует ли маршрут не длиннее, чем заданное значение k, относится к классу NP-полных задач.

В настоящий момент существует несколько типов решения этой задачи: точные (полный перебор, метод ветвей и границ, линейное программирование и т.д), приближенные (жадный алгоритм, метод ближайшего соседа, метод шнурка и т.д.) и эвристические (отжига, генетический, муравьиный и т.д.). Точные алгоритмы находят абсолютно правильный результат, но редко применяются на практике из-за долгого времени выполнения. Приближенные и эвристические алгоритмы выполняются в несколько раз быстрее, они находят достаточно хорошее решение (относительно правильный результат) за приемлемое время. Для определения насколько данное решение близко к абсолютному.

Целью данной работы является исследование методов решения задачи коммивояжера.

Поставленная цель определила следующие задачи:

1. Изучить методы решения задачи коммивояжёра, относящиеся к разным типам: точный метод (метод полного перебора), приближенный метод (жадный алгоритм), эвристический метод (классическая реализация

генетического алгоритма, в котором предполагается, что генотип имеет фиксированную длину).

2. Реализовать изученные методы решения задачи коммивояжера на языке C# в среде Visual Studio.
3. Провести серию экспериментов, по использованию данных методов для решения задачи коммивояжера для различных значений n (количества городов).
4. Провести сравнительный анализ результатов, полученных в ходе экспериментов, с точки зрения затрат времени и памяти при различных значения n.
5. Рассмотреть возможность оптимизации программного кода реализованных методов за счет распараллеливания и выбора структуры данных для представления взвешенного графа, а также оценить эффективность выполненных оптимизаций.

Методологические основы методов решения задачи коммивояжера представлены в работах Applegate D.L. [1], Кобак В.Г.[2], Бураков М.В. [3], Гавриляко В.М.[4], Могилко А.А. [5].

Практическая значимость бакалаврской работы. Все основные экспериментальные данные, программные реализации и выводы, изложенные в бакалаврской работе, получены автором самостоятельно. Полученные результаты позволят оптимизировать и увеличить эффективность решения задач в сфере коммуникации и логистики. Так же их можно использовать как фундамент для реализации более усовершенствованных программных решений.

Структура и объём работы. Бакалаврская работа состоит из введения, 2 разделов, заключения, списка использованных источников и 2 приложений. Общий объем работы – 67 страниц, из них 56 страниц – основное содержание, включая 19 рисунков и 5 таблиц, цифровой носитель в качестве приложения, список использованных источников информации – 33 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Постановка задачи коммивояжера в терминах теории графов» посвящен рассмотрению основных понятий теории графов, постановке задачи коммивояжера и истории ее возникновения, обзору методов решения задачи коммивояжера.

Задача коммивояжёра (англ. Travelling salesman problem, TSP) – задача дискретной оптимизации, заключающаяся в поиске самого выгодного маршрута, проходящего через указанные города ровно по одному разу с последующим возвратом в исходный город. В таком случае выбор осуществляется среди гамильтоновых циклов. В условиях задачи указывается критерий выгодности маршрута (самый дешёвый, кратчайший и т.д.) и соответствующие матрицы стоимости, расстояний и т.д.

В работе подробно рассмотрены методы полного перебора, жадный и генетический алгоритмы. Эти методы были выбраны, так как они являются типичным представителями свой группы. Полный перебор относится к точным алгоритмам, жадный алгоритм – к приближенным алгоритмам, а генетический – к эвристическим алгоритмам.

Метод полного перебора заключается в том, что выполняется перебор всех возможных комбинаций вершин (пунктов назначения). Как известно из математики, число таких перестановок равно n!, где n – количество точек. Так как в задаче коммивояжера исходный пункт обычно принимается одним и тем же (первая точка), то нам достаточно перебрать оставшиеся, т.е. количество перестановок будет равно (n–1)!. Этот алгоритм всегда дает точное решение задачи коммивояжера, однако продолжительность таких вычислений может занять много времени

Жадный алгоритм заключается в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным. Другими словами, на каждом шаге алгоритма принимается решение, являющееся наиболее выгодным для этого шага, без учета того, что происходит на последующих шагах поиска. Тем самым допуская возможность,

выбрать путь, который окажется дорогостоящим далее. Однако, такой подход сильно сокращает время поиска приближенного решения, так как алгоритм рассматривает только одну смежную вершину

Генетический алгоритм – это алгоритм поиска, используемый для задач оптимизации и моделирования, основанный на механизмах естественного отбора и наследования биологической эволюции. ГА возникли в результате попыток копирования естественных процессов живой природы. Идеи эволюционных вычислений стали появляться в 50-ых годах XX века с работ Нильса Баричелли, а в 1975 г. вышла основополагающая книга Дж. Холланда — «Адаптация в естественных и искусственных системах», в которой был предложен генетический алгоритм для задач оптимизации.

Достоинства ГА: унифицированный подход к решению различных проблем; отличные результаты при решении сложных NP-полных задач; могут оптимизировать решение другого метода.

Основные недостатки ГА: преждевременная сходимость; нахождение субоптимальных решений вместо оптимальных; чувствителен к настраиваемым параметрам и сложен в реализации.

Наиболее приспособленные особи получают возможность "воспроизводить" потомство с другими особями. Новое поколение будет содержать более высокое соотношение хороших характеристик. Эти характеристики распространятся по всей популяции, и популяция будет сходиться к оптимальному решению задачи. Менее приспособленные особи с меньшей вероятностью смогут воспроизвести потомков, их свойства будут постепенно исчезать в процессе эволюции. Упрощенная схема работы ГА:

1. Генерация начальной популяции.
2. Оценка популяции и еѐ селекция (отбор в популяцию родителей).
3. Применение генетических операторов (кроссинговер, мутация, инверсия) к родительской популяции и получение потомков.
4. Генерация новой популяции (для следующей итерации).
5. Проверка условия остановки.

6. Вывод наилучшей особи.

Второй раздел «Программная реализация методов решения задачи коммивояжера» посвящен обзору инструментальных средств и средства профилирования VS 2017, реализации последовательных версий полного перебора, жадного и генетического алгоритмов, реализации параллельных версий полного перебора и генетического алгоритма, проведению сравнительного анализа оптимизированных версий методов решения задачи коммивояжера.

Был реализован метод полного перебора, распараллелен, а так же была проведена оптимизация параллельной версии. Сравнительный анализ трех реализаций полного перебора можно посмотреть в таблице 1.

Таблица 1 – время выполнения последовательного, параллельного и оптимизированного параллельного полного перебора.

Количество вершин	Время выполнения последовательной реализации	Время выполнения параллельной реализации	Время выполнения оптимизированной параллельной реализации
10	791 мс	400 мс	309 мс
11	16 сек 566 мс	8 сек 248 мс	3 сек 111 мс
12	2 мин 56 сек 823 мс	1 мин 23 сек 246 мс	37 сек 578 мс
13	18 мин 14 сек 551 мс	11 мин 40 сек 440 мс	9 мин 26 сек 351 мс

Распараллеливание этого алгоритма дало ускорение в 2 раза, так как вычисления производятся на 2 ядрах одновременно. Оптимизация параллельной версии, привела к ускорению почти в 3 раза, так как в алгоритме производится большое количество обращений к элементам коллекции граф. Доступ к элементу SortedList осуществляется за O(logN), где N – это количество вершин в графе. А доступ к элементу в двумерном массиве происходит за O(1).

Реализован жадный алгоритм, а так же была проведена его оптимизация. Сравнительный анализ двух реализаций жадного алгоритма можно увидеть в таблице 2.

Таблица 2 – время выполнения последовательного и оптимизированного жадного алгоритма.

Количество вершин	Время выполнения последовательной реализации	Время выполнения оптимизированной реализации
2000	10 сек 903 мс	9 сек 826 мс
4000	1 мин 16 сек 64 мс	1 мин 14 сек 244 мс
5000	2 мин 25 сек 53 мс	2 мин 23 сек 440 мс
6000	4 мин 2 сек 781 мс	4 мин 1 сек 259 мс

Распараллелить жадный алгоритм не удалось, так как небольшие действия, выполняющиеся в одной итерации построения узла обхода, нельзя распараллеливать, потому что накладные расходы на создание потоков будут слишком большие по сравнению с выигрышем от небольшой операции. Полученные данные совпали результатами других исследователей.

Оптимизация алгоритма с помощью средств профилирования тоже заметно не улучшило показатели кода, так как обращение к элементам графа происходит в большинстве случаев единожды.

Был реализован классический генетические алгоритм, распараллелен, а так же была проведена оптимизация параллельной версии. Сравнительный анализ трех реализаций генетического алгоритма представлен в таблице 3.

Таблица 3 – время выполнения последовательного, параллельного и оптимизированного генетического алгоритма.

Количество вершин	Время выполнения последовательной реализации	Время выполнения параллельной реализации	Время выполнения оптимизированной реализации
40	789 мс	300 мс	44 мс
400	12 сек 429 мс	5 сек 872 мс	115 мс
1000	39 сек 71 мс	19 сек 979 мс	227 мс
5000	6 мин 29 сек 153 мс	3 мин 12 сек 447 мс	1 сек 815 мс

Причина полученных результатов такая же, как и для метода полного перебора.

Сравнительный анализ лучших версий каждого из алгоритмов, ими оказались последовательный жадный алгоритм, оптимизационные варианты

параллельных версий алгоритмов – полного перебора, генетического алгоритма. Вычисления проводились на процессоре Intel(R) Core(TM) i3-6100U CPU @2.30GHz. Количество ядер 2, количество потоков 4.

В таблице 4 приведены зависимость времени выполнения от количества вершин в графе.

Таблица 4 – Зависимость времени выполнения от количества вершин обрабатываемых последовательным жадным и оптимизированными параллельными полным перебором, генетическим алгоритмами. В скобках указаны затраты ресурсов памяти.

Время	Полный перебор		Жадный алгоритм		Генетические алгоритм	
	Количество вершин	Объем памяти (память процесса/ размер кучи)	Количество вершин	Объем памяти (память процесса/ размер кучи)	Количество вершин	Объем памяти (память процесса/ размер кучи)
5 секунд	11	11 мб / 0,078 мб	1500	59 мб / 29,633мб	10000	441 мб / 495,986 мб
1 минута	-		4000	318 мб / 273,833 мб	Более 22000	Более 1293 мб / 132,424 мб
4 минуты	12	12 мб / 0,088 мб	6000	886 мб / 845,521 мб		
10 минут	13	13 мб / 0,09 мб	8300	1032 мб / 979,886 мб		

По приведенным данным можно сделать вывод о том, что самым быстрым алгоритмом является генетический алгоритм, далее идет жадный алгоритм и самым медленным является полный перебор.

Потенциал оптимизированного параллельного генетического алгоритма огромный, но при этом он требует больших затратов памяти и вычислительных ресурсов. Поэтому при запуске генетического алгоритма с более чем 22 000 вершинами не хватает оперативной памяти, «out of memory exception», из-за этого не удалось выяснить какое количество вершин алгоритм обрабатывает за 1 минуту и дольше.

В таблице 5 приведены сравнения точности решения задачи коммивояжера разными рассматриваемыми алгоритмами. Веса над ребрами выбираются в диапазоне от 1 – 10 усл.ед, так как вершин мало. Большой разброс весов при малом количестве вершин в граф приведет к большим погрешностям в вычислении для жадного и генетического алгоритмов.

Таблица 5 – Результаты решения задачи коммивояжера полным перебором, жадным и генетическим алгоритмами в графе с 13 вершинами. Графы представлены в приложении Б.

Название алгоритма	Граф 1		Граф 2		Граф 3		Граф 4	
	Длина пути	Время выполнения	Длина пути	Время выполнения	Длина пути	Время выполнения	Длина пути	Время выполнения
Полный перебор	19	10 мин 12 сек 533 мс	20	10 мин 56 сек 264 мс	22	10 мин 49 сек 113 мс	29	11 мин 23 сек
Жадный алгоритм	28	1 мс	29	1 мс	28	1 мс	36	1 мс
Генетический алгоритм	34	25 мс	32	30 мс	31	26 мс	32	35 мс

Приведенные выше данные соответствуют теоретической базе, полный перебор находит точный ответ и выполняется дольше всех. Жадный и генетический алгоритмы находят лишь приближенные, но выполняются они гораздо быстрее.

Проведенные эксперименты позволили получить следующий вывод, который можно классифицировать по 3 признакам: время выполнение, корректность результата, количество и сложность кода.

1. *Время выполнения*

Лучшим на маленьких графах с количеством вершин менее 13 оказался жадный алгоритм – время выполнения его лучшей реализации меньше, чем у ГА, примерно в 20 раз, не учитывая тот факт, что решения приближенны. Алгоритм ищет первый попавшийся путь, а, значит, не рассматривает множество других вариантов прохождения по графу, за счет чего сэкономил много времени.

Далее следует оптимизированный параллельный генетический алгоритм, однако он является лучшим при работе с большим графами. Такой результат получился потому, что классический генетический алгоритм является эвристическим алгоритмом, то есть тот, который находит приближенный результат, затрачивая при этом меньше ресурсов.

Худший результат по времени выполнения дал полный перебор – его время возрастало факториально и уже для 14 вершин дождаться результата не удалось.

2. Корректность результата

По корректности результата алгоритмы можно разделить следующим образом: полный перебор – самый точный метод, потому что он находит наименьший вес пути, и с ним можно сравнивать результаты других методов.

Жадный алгоритм, ГА – приближенные методы. Они находят не самое лучшее, но с другой стороны сильно сокращают время выполнения по сравнению с точными методами. Жадный алгоритм, несмотря на то, что он искал первый попавшийся путь, показал достаточно неплохие результаты. Его вес пути во всех случаях отличался от наименьшего веса лишь на 6-9 единиц.

По сравнению с ними ГА выдавал вес пути на 3-15 единиц отличающийся от минимума, то есть отнюдь не самый хороший (в 2 раза больше). Но это можно обосновать тем, что начальная популяция состояла всего из 50 или 100 случайных особей, в то время как общее число возможных путей измерялось десятками или сотнями тысяч. То есть начальная популяция, полностью зависела в данном случае от разработчика, и ее можно было бы выбрать и лучшим образом – либо подвергнуть какому-либо отбору вход в нее, либо увеличить размер популяции. Тогда возросла бы вероятность найти хорошее решение.

3. Количество и сложность кода

Если сравнивать все реализованные методы по количеству кода, то меньше всего строк потребовалось на полный перебор, жадный алгоритм.

Каждый из методов был реализован в 1-2 функции, и их код несложен и интуитивно понятен.

Самым сложным в результате анализа был признан генетический алгоритм. ГА реализует сложную схему естественного отбора по подобию происходящего в живой природе. Также ГА имеет достаточно объемный код, в котором много проверок на крайние случаи, и около десятка функций; алгоритму необходимо выполнить несколько десятков или сотен итераций для того, чтобы добиться хотя бы приближенного результата (если решение не приближается к оптимуму и не вырождается).

ЗАКЛЮЧЕНИЕ

В проделанной работе были решены все поставленные задачи, что позволило исследовать различные методы решения задачи коммивояжера. В частности, были изучены методы решения задачи коммивояжёра, относящиеся к разным типам: точный метод (метод полного перебора), приближенный метод (жадный алгоритм), эвристический метод (классическая реализация генетического алгоритма, в котором предполагается, что генотип имеет фиксированную длину), реализовать изученные методы решения задачи коммивояжера на языке C# в среде Visual Studio, проведены серии экспериментов, по использованию данных методов для решения задачи коммивояжера для различных значений n (количества городов), проведены сравнительные анализы результатов, полученных в ходе экспериментов, с точки зрения затрат времени и памяти при различных значения n, рассмотрены возможность оптимизации программного кода реализованных методов за счет распараллеливания, кеширования и выбора структуры данных для представления взвешенного графа, а также оценить эффективность выполненных оптимизаций.

В процессе выполнения выпускной квалификационной работы было проведен ряд экспериментов на случайных графах от 10 до нескольких десятков тысяч вершин, измерено время выполнения методов решения задачи коммивояжера и затраченные ресурсы памяти.

Проведенный сравнительный анализ по показателям эффективности позволил определить, какую оптимальность имеет каждый из методов, какие у них преимущества и недостатки. Лучшие по точности – полный перебор. Лучшими по времени выполнения был признан оптимизационный генетический алгоритм. Жадный алгоритм оказался самым оптимальным он выполняется быстрее, чем полный перебор и находит решения лучше, чем генетический алгоритм.

Отдельные части бакалаврской работы были опубликованы:

1. Батталова Н.И. Оптимизация программного кода с использованием средств профилирования visual studio // Информационные технологии в образовании: Материалы IX Всерос.научно-практ. конференции. – Саратов: ООО «Издательский центр «Наука», 2017. – С. 149-154.

2. Батталов А.И. Использование средств профилирования visual studio для оптимизации программного обеспечения // А.И. Батталова, Н.И. Батталова Научное обозрение. Технические науки. – 2018. – № 1. – С. 5-9;

Основные источники информации:

1. Applegate D.L. The Traveling Salesman Problem: A Computational Study/ V. Chvátal & W.J. Cook (2006).

2. Кобак В.Г. Исследование влияния сильных мутаций при решении задачи коммивояжера модифицированной моделью голдберга// В.Г. Кобак, И.Ш. Рудова Известия ЮФУ. Технические науки. 2017. № 3 (188). С. 140-148.

3. Бураков М.В. Генетический алгоритм: теория и практика Учебное пособие / М.В. Бураков. – Спб.: ГУАП, 2008. – 164 с.: ил.

4. Гавриляко В.М. Задача коммивояжера и генетические алгоритмы // В.М. Гавриляко М.: Наука, 1970. - 272с.

5. Могилко А.А. Параллельный алгоритм поиска ближайшей точки в радиусе // А.А. Могилко Россия, МГТУ им.Н.Э.Баумана – 11.11.2013.