Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Фундаментальные науки
КАФЕДРА Прикладная математика

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Марцинович Софья Эльдаровна
фамилия, имя, отчество

Группа ФН2-41Б

Тип практики ознакомительная

Название предприятия НУК ФН МГТУ им. Н.Э. Баумана

Студент _____ Марцинович С.Э.
подпись, дата *фамилия и.о.*

Руководитель практики _____ Савельева И. Ю.
подпись, дата *фамилия и.о.*

Оценка _____

2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Кафедра «Прикладная математика»

З А Д А Н И Е
на прохождение ознакомительной практики

на предприятии НУК ФН МГТУ им. Н.Э. Баумана

Студент Марцинович Софья Эльдаровна
фамилия, имя, отчество

Во время прохождения ознакомительной практики студент должен

1. Изучить на практике основные возможности языка программирования C++ и системы MATLAB, закрепить знания и умения, полученные в курсах «Введение в информационные технологии», «Информационные технологии профессиональной деятельности».
2. Изучить способы реализации методов решения задачи коммивояжера.
3. Реализовать метод полного перебора и алгоритм имитации отжига.

Дата выдачи задания «1» марта 2020 г.

Руководитель практики _____ Савельева И. Ю.
подпись, дата *фамилия и.о.*

Студент _____ Марцинович С.Э.
подпись, дата *фамилия и.о.*

Оглавление

Введение

Основной целью ознакомительной практики 4-го семестра, входящей в учебный план подготовки бакалавров по направлению 01.03.04 — Прикладная математика, является знакомство с особенностями осуществления деятельности в рамках выбранного направления подготовки и получение навыков применения теоретических знаний в практической деятельности.

В рамках освоенного курса «Введение в информационные технологии» изучены основные возможности языка программирования C++ и сформированы базовые умения в области программирования на C++. Задачей практики является закрепление соответствующих знаний и умений и овладение навыками разработки программ на языке C++, реализующих заданные алгоритмы. На протяжении курса «Информационные технологии профессиональной деятельности» был получен опыт работы в системе компьютерной верстки TEX. Этот опыт оказался полезным при написании отчета по ознакомительной практике. Кроме того, практика предполагает формирование умений работы с системами компьютерной алгебры и уяснение различий в принципах построения алгоритмов решения задач при их реализации на языках программирования высокого уровня (к которым относится язык C++) и в среде MATLAB.

Формулировка индивидуального задания

Задача коммивояжера

Дан набор вершин и набор ребер, их соединяющих. Каждое ребро является направленным и характеризуется весом – положительным числом – стоимостью движения из его начала в конец. Для любой пары вершин существует два ребра, их соединяющих (в прямом и обратном направлении). Требуется найти маршрут минимальной стоимости, проходящий через все вершины ровно по одному разу и возвращающийся в первую вершину.

Необходимо выполнить следующее:

1. Решить задачу «полным перебором», прокладывая все возможные пути (не заходя дважды в одну и ту же вершину).
2. Решить задачу с помощью алгоритма имитации отжига.

1. История рассматриваемой задачи

Задача коммивояжёра (*англ.* Travelling salesman problem) — одна из самых известных задач комбинаторной оптимизации, заключающаяся в поиске самого выгодного маршрута, проходящего через указанные города хотя бы по одному разу с последующим возвратом в исходный город. В условиях задачи указываются критерий выгодности маршрута (кратчайший, самый дешёвый, совокупный критерий и тому подобное) и соответствующие матрицы расстояний, стоимости и тому подобного.

Точно неизвестно, когда задачу коммивояжера исследовали впервые. Однако в 1832 году была издана книга [1], в которой описана задача, но математический аппарат для её решения не применяется [1].

Ранним вариантом задачи может рассматриваться игра «Икосиан» (*англ.* Icosian Game) Уильяма Гамильтона [2] 19 века, которая заключалась в том, чтобы найти маршруты на графе с 20 узлами. Вскоре появилось известное сейчас название «задача странствующего торговца», которую предложил Хасслер Уитни [3] из Принстонского университета.

[1] «Коммивояжёр — как он должен вести себя и что должен делать для того, чтобы доставлять товар и иметь успех в своих делах — советы старого курьера» (*нем.* Der Handlungsreisende – wie er sein soll und was er zu tun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein – von einem alten Commis-Voyageur, 1832)

[2] Уильям Гамильтон (*англ.* William Rowan Hamilton, 1805—1865)— ирландский математик, механик-теоретик, физик-теоретик.

[3] Хасслер Уитни (*англ.* Hassler Whitney, 1907—1989)—американский математик.

2. Обзор методов решения

Все эффективные (сокращающие полный перебор) методы решения задачи коммивояжера — методы эвристические. В большинстве эвристических методов находится не самый эффективный маршрут, а приближённое решение.

Простейшими методами решения задачи коммивояжера являются:

- полный перебор;
- случайный перебор;
- жадные алгоритмы:
  - метод ближайшего соседа;
  - метод включения ближайшего города;
  - метод самого дешёвого включения;
- метод минимального остовного дерева;
- метод имитации отжига.

На практике применяются различные модификации более эффективных методов: метод ветвей и границ и метод генетических алгоритмов, а также алгоритм муравьиной колонии.

3. Метод полного перебора

3.1. Описание метода полного перебора

При реализации полного перебора рассматривались в качестве потенциальных путей все возможные перестановки из порядковых номеров узлов. При генерации перестановок использовалась рекурсивная функция [2].

На выходе имеется $n!$ перестановок. Такое решение задачи подходит только для графов с малым количеством узлов.

Область применимости алгоритма можно немного увеличить, если зафиксировать стартовую позицию. Перебор с учётом этого наблюдения сокращается незначительно: количество перестановок сокращается до $(n-1)!$. Экспериментальным путем было установлено, что программа работает на графах с количеством узлов $n \leqslant 11$.

По результатам работы рекурсии получается список всех возможных путей. Из них выбирается путь с минимальным весом.

3.2. Особенности реализации на языке C++

Для перебора всех возможных перестановок была написана рекурсивная функция `rec`. В векторе `p` хранится текущая перестановка, в матрице `a` хранятся все перестановки.

Создается вектор `used` со значениями логического типа `bool`, который хранит информацию о том, какие порядковые номера уже вошли в перестановку. Изначально он заполняется ложными значениями. Счетчик `i` цикла отвечает за перебор возможных вариантов для следующего шага. Если `used[i]` равно `false`, то `i` добавляется в конец перестановки `p` и в `used[i]` записывается `true`. Происходит переход на следующий уровень рекурсии. Если `used[i]` равно `true`, значит этот город уже вошел в перестановку, следовательно, осуществляется переход к следующей итерации цикла. Аргументом функции является число `idx` типа `int`. Оно обозначает глубину рекурсии.

3.3. Особенности реализации в системе MATLAB

В системе MATLAB существует мнгого различных возможностей для работы с матрицами и векторами. Для решении задачи полного перебора были реализованы те же самые идеи, что и при написании программы на языке C++. Функция `rec` была написана в отдельном файле `rec.m`. Были использованы глобальные переменные `global n p a used` . Также как и в результате работы программы на языке C++, найденный маршрут выводится в текстовый файл.

Обработка одних и тех же тестов в системе MATLAB и на языке C++ занимает разное количество времени: в системе MATLAB тесты обрабатываются дольше. Подробнее о результатах замеров времени выполнения будет изложено ниже в разделе «Примеры решения модельных задач».

4. Алгоритм имитации отжига

4.1. Описание алгоритма имитации отжига

Алгоритм имитации отжига (*англ.* Simulated annealing) основывается на имитации физического процесса, который происходит при кристаллизации вещества, в том числе при отжиге металлов. Предполагается, что атомы уже выстроились в кристаллическую решётку, но ещё допустимы переходы отдельных атомов из одной ячейки в другую, и что процесс протекает при постепенно понижающейся температуре. Переход атома из одной ячейки в другую происходит с некоторой вероятностью, причём вероятность уменьшается с понижением температуры. Устойчивая кристаллическая решётка соответствует минимуму энергии атомов, поэтому атом либо переходит в состояние с меньшим уровнем энергии, либо остаётся на месте [3].

Имитация отжига похожа на восхождение на холм или поиск градиента с несколькими модификациями. При поиске на основе градиента направление поиска зависит

от градиента, и поэтому оптимизируемая функция должна быть непрерывной функцией без разрывов. Имитация отжига не требует, чтобы функция была гладкой и непрерывной. Чтобы избежать проблемы застревания в локальных минимумах, иногда принимаются маршруты с расстоянием больше текущего маршрута. По мере снижения температуры вероятность принятия плохого решения уменьшается, и на заключительных этапах поиск наилучшего маршрута становится похожей на поиск на основе градиента.

Предварительно выбирается маршрут, в котором города стоят по порядку. Далее при каждой итерации алгоритма меняются местами два случайных города. Длина нового маршрута сравнивается с длиной текущего маршрута. Если длина нового маршрута меньше, то текущий маршрут заменяется на новый. В случае, когда новый маршрут оказывается длиннее предыдущего, происходит замена с вероятностью P, которая вычисляется по формуле

$$P = e^{\left(-\frac{\Delta E}{T}\right)}, \tag{1}$$

где ΔE является разницей длин нового и текущего маршрутов, а T — температура в данный момент времени.

Алгоритм имитации отжига не гарантирует нахождения минимума функции, однако преимуществом метода является то, что он вытаскивает ее из локальных минимумов.

На рис.1 приведена блок-схема данного алгоритма [4].

4.2. Особенности реализации на языке C++

При реализации алгоритма имитации отжига было принято решение хранить информацию в виде матрицы смежности. На главной диагонали матрицы стоят нули. Нумерация городов идет с нуля. Так как выбор точки старта не принципиален, то всегда стратовой вершиной выбирался город с индексом нуль.

Были реализованы следующие функции:

- `Swap()` — данная функция меняет местами два случайных города. Она принимает в качестве аргументов вектор, содержащий текущий маршрут, и два порядковых номера городов, которые следуетпоменять местами;
- `GetTotalDistance()` — это функция, предназначенная для подсчета суммарной длины маршрута. Ее аргументом является вектор с текущим маршрутом. Функция возвращает число типа `double` — длину маршрута;
- `GetNextArrangement()` — данная функция создает новый маршрут на основе текущего. Она принимет на вход вектор, содержащий текущий маршрут;

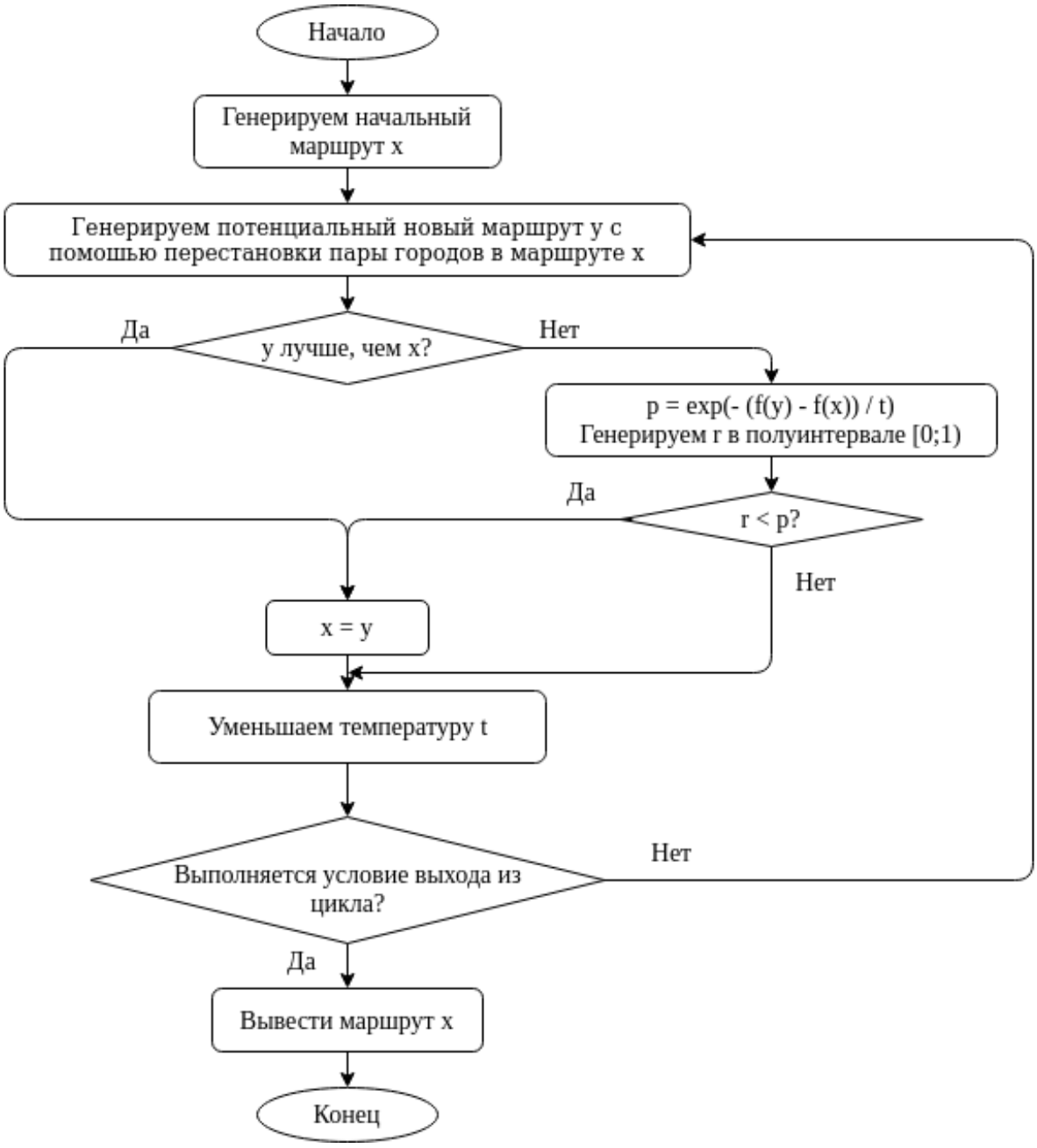

Рис. 1. Блок-схема алгоритма имитации отжига

• `GetProbability()` — это функция, которая считает вероятность по формуле (1).

Сам алгоритм был реализован как функция `Anneal()`. Она не требует входных аргументов. Внутри задаются начальная температура (`temperature`), абсолютная температура (`absoluteTemperature`) и скорость охлаждения (`coolingRate`). С помощью цикла `while` функция находит наилучший путь, понижая температуру при каждой итерации. Результат работы функции это вектор с порядком городов соответствующим кратчайшему маршруту.

Поскольку в процессе работы алгоритма перестановки получаются случайным образом, то в финальном маршруте стартовая и финишная точки также оказываются случайными. Для удобства проверки правильности работы программы необходимо пронумеровать города с единицы и «передвинуть» маршрут так, чтобы он начинался с узла под номером 1.

4.3. Особенности реализации в системе MATLAB

В системе MATLAB были реализованы те же самые идеи, что и в системе C++, но с использованием внутренних функций работы с матрицами.

Основным отличием разработки программы стала возможность визуализировать процесс поиска оптимального маршрута. При каждой итерации с использованием функции `plot` строился новый путь, что позволяло следить за работой программы в режиме реального времени.

По условию задачи задается только матрица смежности, в то время как для построения картинки требуются координаты городов. Тесты были модифицированы таким образом, чтобы из них считывалась и матрица смежности, и вектор координат. В качестве одного из тестов, были рассмотрены координаты 29 городов Баварии и длины реальных дорог между ними. Также были разработаны тесты, состоящие только из набора координат городов. В этом случае матрица смежности вычислялась программным образом: расстояние между городами рассчитывались с помощью евклидовой метрики.

5. Примеры решения модельных задач

Приведенные ниже примеры позволяют сравнить работу двух алгоритмов на тестах с различными входными данными. Для наглядности для некоторых тестов были приведены иллюстрации, построенные с помощью системы MATLAB.

В файле с входными данными в первой строке записано количество городов `n`, и следующие `n` строк занимает матрица смежности.

Пример 1. Рассматривается тест, состоящий из 4 городов, которые необходимо обойти. Это простейший случай задачи коммивояжера. На данном тесте можно проиллюстрировать формат входных и выходных данных и наглядно показать выбранный маршрут на графе.

На рис. 2 изображен взвешенный граф. Синим цветом отмечен маршрут коммивояжера с минимальной длиной.

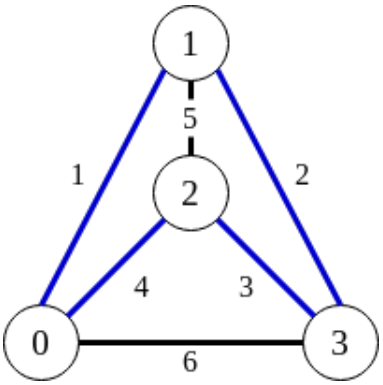

Рис. 2. Иллюстрация примера 1

Входные данные:

```
4
0  1  4  6
1  0  5  2
4  5  0  3
6  2  3  0
```

В таблице 1 приведено сравнение результатов работы программ, реализованных методом полного перебора и методом имитации отжига. Можно заметить, что в обоих случаях минимальный маршрут определен правильно.

Таблица 1. Пример работы алгоритмов для 4 городов

	Маршрут	Длина пути	Относительная погрешность, %	Затраченное время, с
Вывод перебором	1 3 4 2 1	10	0,0	0,0005
Вывод методом имитации отжига	1 2 4 3 1	10	0,0	0,0094

Пример 2. В тесте содержится 11 городов — это максимальное количество городов, которое может быть обработано методом полного перебора.

В этом примере мы пытались добиться расхождения результатов работы двух методов. Поскольку на таком относительно небольшом тесте метод имитации отжига часто дает точный ответ, то для получения неточного ответа обработка теста была проведена 50 раз. В таблице 2 приведены средние значения длин найденных путей для разных значений параметра `coolingRate`. При увеличении этого параметра относительная погрешность уменьшается.

Таблица 2. Пример работы алгоритмов для 11 городов

	Средняя длина полученного пути	Относительная погрешность, %	Затраченное время, с
Полный перебор	0,0150	0,0	5,4337
Алгоритм имитации отжига при `coolingRate`=0.9	0,0318	112,0	0,0010
Алгоритм имитации отжига при `coolingRate`=0.99	0,0182	21,3	0,0094
Алгоритм имитации отжига при `coolingRate`=0.999	0,0152	1,2	0,1211

Также этот пример позволяет сравнить время, затраченное на решение задачи двумя разными методами. Для работы метода полного перебора требуется заметно больше времени, чем для работы алгоритма имитации отжига.

Пример 3. Тест состоит из 60 точек, которые расположены на двух концентрических окружностях радиуса 1 и 1.05. Аналитически была посчитана длина минимального пути:

$$30 \cdot 0.05 + 1 \cdot \pi + 1.05 \cdot \pi \approx 7,94026494.$$

В таблице 3 приведено сравнение полученной длины пути и относительной погрешности при разных значениях параметра `coolingRate`. В отдельной колонке показывается минимальная длина пути, которая была достигнута за все время работы программы. Можно заметить, что эта величина всегда меньше или равна финальному ответу.

Ниже на рисунке 3 изображена случайная перестановка, с которой может начинаться работа алгоритма. На рисунках 4–8 изображены результаты работы алгоритма при увеличении значения параметра `coolingRate`. При `coolingRate` = 0.99999 ответ практически совпадает с теоретически полученным.

Таблица 3. Пример работы алгоритма для 60 городов

`coolingRate`	Длина полученного пути	Достигнутый локальный минимум	Относительная погрешность, %	Время работы на C++, с	Время работы в MATLAB, с
0,9	53,8579	12,0199	578,2885	0,0040	0,0245
0,99	31,6376	10,4945	298,4451	0,0219	0,1409
0,999	15,7187	12,3693	97,9619	0,2298	0,6690
0,9999	12,5532	12,5532	58,0955	1,9769	6,2904
0,99999	7,9409	7,9404	0,0017	20,4957	66,9513

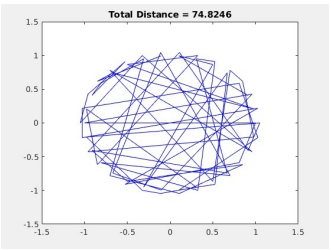

Рис. 3. Случайная перестановка

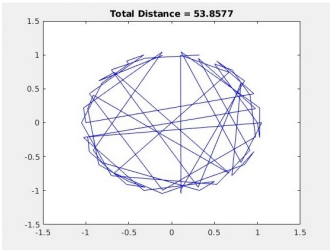

Рис. 4. Результат при `coolingRate` = 0.9

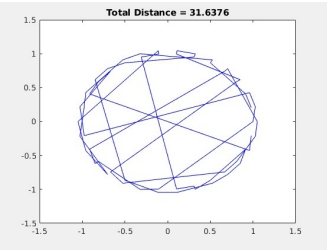

Рис. 5. Результат при `coolingRate` = 0.99

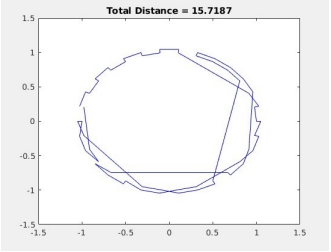

Рис. 6. Результат при `coolingRate` = 0.999

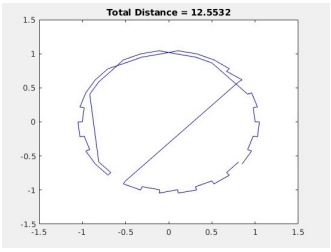

Рис. 7. Результат при `coolingRate` = 0.9999

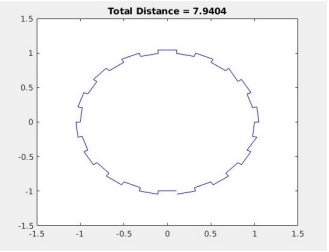

Рис. 8. Результат при `coolingRate` = 0.99999

Пример 4. Алгоритм был испытан на тесте, взятом из источника [6]. Рассматриваются 29 городов Баварии и длины дорог между ними. На рис. 9 изображен результат, полученный в результате работы алгоритма. На рис. 10 изображена художественная иллюстрация найденного пути.

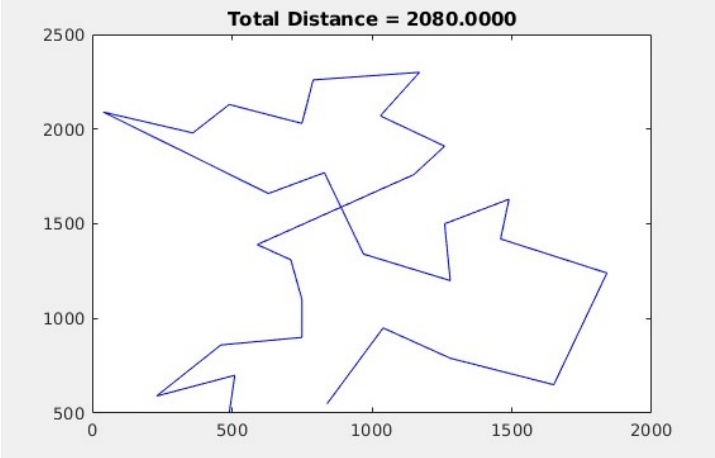

Рис. 9. Найденный маршрут

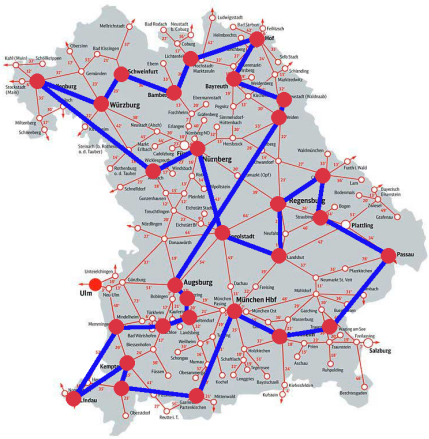

Рис. 10. Иллюстрция маршрута на карте Баварии

Заключение

Результатом выполнения практического задания стало знакомство с теорией графов, а конкретно с задачей коммивояжера. Для реализации метода перебора и алгоритма имитации отжига были использованы сначала язык программирования C++, а после система MATLAB.

Разобранный алгоритм имитации отжига имеет более чем полувековую историю и до сих пор находит широкое применение в информационных технологиях. Благодаря этому практическому заданию были освоены способы реализации этого алгоритма на практике, а также выявлены их преимущества и недостатки.

Список использованных источников

1. Задача о коммивояжера // Википедия. Свободная энциклопедия.
   URL: https://ru.wikipedia.org/wiki/Задача_коммивояжёра
   (дата обращения: 05.06.2020).
2. Геререация перестановок. Перебор // Платформа Coursera.
   URL: https://www.coursera.org/lecture/sportivnoe-programmirovanie/1-3-gienieratsiia-pieriestanovok-5BAS4
   (дата обращения: 05.06.2020).
3. Введение в оптимизацию. Имитация отжига // Хабр.
   URL: https://habr.com/ru/post/209610/
   (дата обращения: 05.06.2020).
4. List-Based Simulated Annealing Algorithm for Traveling Salesman Problem // Hindawi.
   URL: https://www.hindawi.com/journals/cin/2016/1712630/
   (дата обращения: 05.06.2020).
5. Simulated Annealing — Solving the Travelling Salesman Problem (TSP) // CodeProject.
   URL: https://www.codeproject.com/Articles/26758/Simulated-Annealing-Solving-the-Travelling-Salesma
   (дата обращения: 05.06.2020).
6. MP-TESTDATA — The TSPLIB Symmetric Traveling Salesman Problem Instances // TSPLIB.
   URL: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/
   (дата обращения: 05.06.2020).