МИНОБРНАУКИ РОССИИ

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

Кафедра теории функций и стохастического анализа

ОБОБЩЕННАЯ ЗАДАЧА КОММИВОЯЖЕРА

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 248 группы
направления 09.04.03 — Прикладная информатика

механико-математического факультета
Немоляева Ильи Владиславовича

Научный руководитель
доцент, к. ф.-м. н., доцент _____ М. Г. Плешаков

Заведующий кафедрой
д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2022

ВВЕДЕНИЕ

Современные практические задачи становятся все более требовательными к выполняемым условиям. Данный процесс охарактеризован развитием потребностей и появлением новых ограничений.

Задача коммивояжера о нахождении кратчайшего пути обхода городов в рамках современных потребностей человека не дает возможности удовлетворить его запрос. Например, при покупке товаров или ознакомлении с информацией, ее поиске, пользователь услуг может воспользоваться алгоритмами сортировки, фильтрации, что позволяет ему найти подходящий товар или ресурс. Также встречаются системы, которые по заданным критериям самостоятельно находят подходящий результат (агрегаторы поиска авиабилетов, алгоритмы поиска рекламы и т.д.).

У организации или отдельно взятого человека может возникнуть задача коммивояжера и сейчас, но нахождение ее решения теперь ограничена не только нахождением кратчайшего пути. Формируя данную задачу стоит учитывать цели, которых требуется достичь, тем самым оптимизируя процесс затрат организации или ресурсов отдельного человека.

Целью настоящей работы является описание обобщенной задачи коммивояжера и реализация программы для нахождения решения, используя методы эволюционных вычислений.

В процессе работы были поставлены следующие задачи:

— описать постановку обобщенной задачи коммивояжера;
— сформулировать алгоритм решения;
— реализовать программу нахождения оптимального пути в рамках установленных ограничений.

Работа прошла апробацию на различных конференциях, в частности, на ежегодной студенческой конференции «Актуальные проблемы математики и механики», которую проводил механико-математический факультет СГУ в апреле 2022 года, в секции «Анализ данных», в X Международной научно-практической конференции «Математическое и компьютерное моделирование в экономике, страховании и управлении рисками», ноябрь 2021 года.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Введение содержит в себе описание проблемы, актуальность работы, основные цели и задачи.

Первый раздел содержит описание задачи коммивояжера и постановку задачу её обобщенного видаю. Задача Коммивояжёра формируется следующим образом. Даны n городов и известны расстояния d_{ij}, где $i, j = 1, ..., n$. Коммивояжёр, выходящий из какого либо города, должен посетить $n - 1$ других городов по одному разу и вернуться в начало. Необходимо определить порядок обхода городов, чтобы коммивояжёр прошёл наименьшее из возможных расстояний.

В терминах теории графов классическая задача коммивояжёра формулируется следующим образом: найти гамильтонов цикл μ наименьшей длины на реберно-взвешенном графе $G =< V, D >$, вершины $v_i | i = 1, ..., N$ которого представляют города, а взвешенные рёбра $d_{ij} | i, j = 1, ..., N, i \neq j$ — расстояния между ними.

Обобщим задачу введя новые ограничения. Пусть $C = \{C_1, C_2, ..., C_n\}$ множество городов. Пусть множество $T = \{T_1, T_2, ..., T_k\}$ - это множество способов перемещения между городами. Каждый способ будет иметь некий набор характеристик. Существует заинтересованное лицо, которое определяет критерии оптимизации. Данные критерии будут иметь непосредственное отношение к характеристикам видов транспорта, которые будут использоваться при перемещении между городами.

Таким образом, в рамках условий описанных выше, необходимо найти оптимальный маршрут, который будет учитывать предпочтения (критерии оптимальности) заинтересованного лица. Это означает, что маршрут должен представлять порядок обхода городов с выбором транспорта, который будет соответствовать установленным критериям.

Второй раздел содержит описание алгоритмов для решения обобщенной задачи.

В качестве базы для нахождения оптимального маршрута будет использоваться генетический алгоритм.

Обобщённая структура генетического алгоритма может быть представлена следующим образом:

Рисунок 1 – Блок-схема генетического алгоритма

1. Создание начальной популяции с заданной численностью;
2. Воспроизводство потомков с наследственными генетическими свойствами родителей;
3. Мутация потомков, для изменения генетических свойств, полученных от родителей;
4. Замена текущей популяции новой.

В соответствии с рисунком 1, получается циклический алгоритм. Условием выхода из цикла служит поставленный вопрос задачи.

Отличия генетического алгоритма от других алгоритмов случайного поиска заключаются в следующем:

— в генетическом алгоритме используются наборы параметров, а не сами параметры. Параметры объединяются в цепочки конечной длины, в процессе итераций эти цепочки тестируются и видоизменяются;
— поиск происходит не в точках пространства, а на основе популяции точек;
— генетический алгоритм использует знания о прошлых полученных решениях. Он обеспечивает концентрацию решений в наиболее эффективных областях пространства решений;
— генетический алгоритм использует только оценки решений, а не их производные или другие параметры.

Далее выделим три самых часто используемых способов перемещения из точки A в точку B: автомобиль, поезд и самолет. Каждый вид транспорта по отношению к паре городов будет иметь следующие характеристики исходя

из условий поставленной задачи:

— s - расстояние между городами, характеризуется используемой в рамках задачи проекцией. Например, это может быть декартова система координат или естественные единицы измерения расстояния;
— p - стоимость поездки, характеризуется условной денежной единицей;
— t - время поездки, характеризуется количеством часов;
— k - степень комфорта, характеризуется тремя классами комфорта: эконом, средний, бизнес.

В рамках решения задачи необходимо определить оптимальный способ премещения между городами. Для оптимизации воспользуемся формулировкой критерия ПаретоIten. Вектор решения $\vec{x}' \in S$ называется оптимальным по Парето, если не существует $\vec{x} \in S$ такого, что $f_i(\vec{x}) \leq f_i(\vec{x}')$ для всех $i = 1, ..., k$ и $f_i(\vec{x} < f_i(\vec{x}'))$ для хотя бы одного i.

Наиболее подходящим методом оптимизации в контексте данной задачи является метод изменения ограничений. В рамках данного метода выбирается главная характеристика, по которой будет происходить нахождение минимума или максимума. По остальным характеристикам могут устанавливаться ограничения. К примеру, предположим что необходимо перемещаться между городами как можно быстрее, при этом не тратя на каждую поездку больше фиксированной суммы. Таким образом получим следующее формула нахождения оптимизации будет состоять из одного минимума по главному критерию и списка неравенств, ограничений по другим критериям

Алгоритм решения обобщенной задач:

Шаг 1. Создание входного двумерного массива данных перед обработкой многокритериальной оптимизации. Данный массив должен состоять из n строк и столбцов, в ячейках которого будет храниться набор ребер, соответствующий паре узлов.

Шаг 2. Обработка ячеек входного массива. Решение многокритериальной оптимизации методом изменения ограничений для каждого из набора ребер, выделение оптимального вида транспорта для перемещения между городами.

Шаг 3. Отправка преобразованного массива на вход генетического алгоритма с передачей названия главного критерия оптимизации. В рамках ра-

боты генетического алгоритма функция выживаемости будет подсчитывать минимальное значение по главному критерию, тем самым решая классическую задачу.

В результате получится маршрут, который будет соответствовать введённым ограничениям, так как на этапе преобразования не удовлетворяющие условию задачи ребра будут исключены. Также генетический алгоритм найдёт локальный минимум оптимизируя маршрут по главному критерию.

Третий раздел содержит в себе более подробное описание реализации шагов алгоритма для программного пакета.

Рассмотрим обобщенную задачу на примере нахождения оптимального маршрута из 5 городов, $C = \{C_1, C_2, C_3, C_4, C_5\}$. Пусть между каждой парой городов есть 3 способа перемещения $T_{ij} = \{T_1, T_2, T_3\}$. Каждый из способов имеет в свою очередь 3 характеристики $\{x, y, z\}$. Имеющиеся данные представляют собой граф, у которого каждая вершина соединена со всеми другими вершинами. При чем между ними находится по 3 ребра. Ребра представляют один из способов перемещения между городами, а их характеристики - вес ребра.

Для приведения задачи к классическому виду необходимо на основе критерия выбора сформировать целевую функцию и для каждой пары городов выделить способ перемещения соответствующий критерию оптимальности. Результатом функции будет считаться новым значением веса ребра, а не удовлетворяющие условию ребра будут исключены.

Для генетического алгоритма определим следующую логику выполнения операций.

В качестве целевой функции задачи коммивояжёра будем считать сумму расстояний между городами, слагаемые которой определяются последовательностью городов маршрута $f(x) = \sum d_{ij}$.

В качестве наборов популяции рассматривается последовательность городов маршрута. Таким образом, популяцию можно представить в виде математических объектов, векторов, координатами которых являются номера городов.

Для каждого из маршрутов подсчитывается целевая функция. Получив её значения, в зависимости от заданных критериев, выбираются дальнейшие

действия.

Для этапа селекции применяется метод рулетки, который позволяет на основе приспосабливаемости имеющихся наборов, определять их вероятность стать родителем новых потомков. Таким образом, сохраняется разнообразие, так как менее подходящие особи также имеют шанс на создание потомства. Для определения вероятности стать родителем будет использоваться следующая формула: $p_i = \frac{f_i^{-1}}{\sum f_k^{-1}}$.

Операция скрещивание в рамках задачи коммивояжёра определяет дополнительное условие. Последовательность городов в маршруте не должна содержать дубликатов (одинаковых номеров городов) кроме старта, так как в ходе её прохождения необходимо вернуться в ту точку, из которой был начат путь. Из этого следует что простое скрещивание, когда берётся часть генов родителей и соединяется в новую особь, потомок, не подходит. При таком скрещивании велика вероятность нахождения дубликатов в последовательности, что не соответствует условию задачи.

В соответствии с рисунком 2, выбран алгоритм скрещивания порядка во время работы которого случайным образом с вероятностью $\frac{1}{n-1}$, где n — общее число городов, выбираются две точки a_1 и a_2 ($a_1 < a_2$; $a_1, a_2 \quad [1, n-1]$), называемые точками кроссовера, которые разрезают генотипы родителей на три части.

Рисунок 2 – Схема скрещивания

Четвёртый раздел содержит описание архитектуры приложения и ис-

пользуемых технологий в его реализации.

Программа для решения поставленной задачи имеет вид веб-приложения. Веб-приложение - это программа, имеющая две части исполнения. Визуальная часть — веб-страница, через которую происходит взаимодействие с логикой самой программы. Серверная часть хранит в себе бизнес-логику. Бизнес-логика — это код исполняемый на стороне сервера, недоступный для просмотра или редактирования пользователю. Сервер выполняет запросы пользователя на запись, чтение, удаление и обработку информации, которую он предоставляет.

Приложение выполнено в исполнении микросервисной архитектуры. Микросервисная архитектура — вариант сервис-ориентированной архитектуры программного обеспечения, направленный на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей — микросервисов. Данная архитектура позволяет разбивать большое приложение на слабо зависимые программные модули. Между этими модулями устанавливаются правила общения, протоколы.

В соответствии с рисунком 3, архитектура разбита на два модуля: визуальный интерфейс для работы с картой и задания условий; веб-сервер для работы с данными, которые он получает из первого модуля. Передача данных из одного модуля в другой происходит с помощью протокола HTTP. HTTP — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

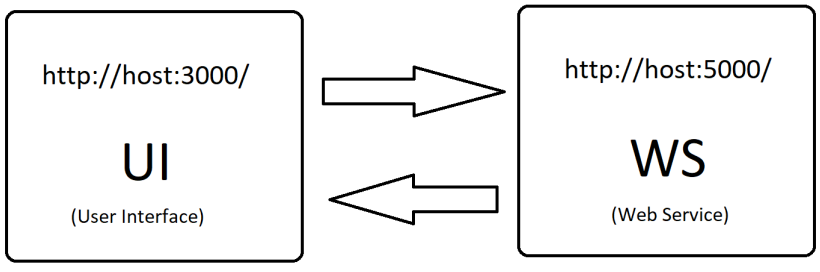

Рисунок 3 – Архитектура приложения

Модуль визуализации данных базируется на технологиях HTML, CSS, JavaScript.

За сборку и запуск модуля отвечает платформа Node.js. Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода.

Самый большой блок взаимодействия представляет собой карта. Основным компонентом программного пакета OpenLayers является объект Map. При инициализации карты и добавлении её на страницу он служит в качестве основы и определяет блок на страницы, в который она будет помещена.

Сам Map не отвечает за такие вещи как центр, уровень масштабирования и проекцию карты. Для управления данными параметрами существует объект View, передаваемый в Map. Также он хранит в себе сферическую проекцию карты. В приложении используется проекция по умолчанию EPSG:3857, в качестве единиц измерения карты используются метры. EPSG:3857 — это система координат сферической проекции Меркатора, популяризированная веб-сервисами, такими как Google, а затем и OpenStreetMap.

За изображение карты отвечают Source и Layer. Первый определяет источник данных для получения изображений карты. Второй выполняет построение карты в соответствии с определенной проекцией, благодаря чему изменение положение карты обновляются самостоятельно. Также необходимо отметить, что для получения изображения карты необходимо подключение к сети Интернет.

В соответствии с рисунком 4, в распоряжении пользователя есть возможности масштабирования карты, установка маркеров, их перемещения по карте, прокрутка карты. По умолчанию центр отображения карты установлен на Московскую область.

На модуль визуализации данных возложены следующие задачи:

— определение главного критерия и ограничений для поиска оптимального маршрута;
— генерация доступных рейсов и их характеристик между городами;
— визуализация и управление картой;

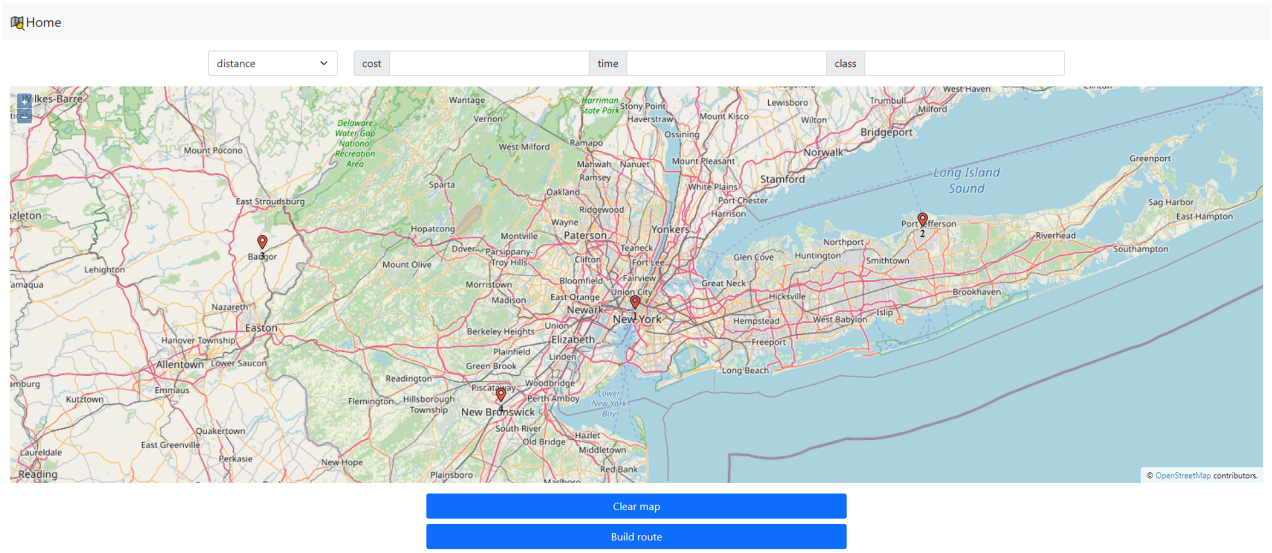

Рисунок 4 – Возможности для работы с картой

— обработка установленных на карте маркеров;
— формирование входных данных задачи отправка из во второй модуль;
— обработка и визуализация полученного ответа.

Веб-сервер написан на языке Python. Python — это простой в освоении мощный язык программирования. Он имеет эффективные высокоуровневые структуры данных и простой, но эффективный подход к объектно-ориентированному программированию. Элегантный синтаксис и динамическая типизация Python, а также его интерпретируемый характер делают его идеальным языком для написания сценариев и быстрой разработки приложений во многих областях на большинстве платформ.

Модуль состоит из трех исполняемых файлов: обработчик запросов, алгоритм многокритериальной оптимизации и генетический алгоритм.

В обработчике описаны три запроса на получение и обработку данных: host:5000/properties, host:5000/set_main_property, host:5000/process.

Первые два отвечают за получение критериев, доступных для учета и обработки бизнес-логикой, и сменой порядке следования критериев. По умолчанию первый критерий считается основным критерием оптимизации.

Для работы над решением задачи обработчик выполняет метод по третьему запросу. В соответствии с рисунком 5, данный метод преобразует данные под типизацию языка Python затем отправляет их сначала в алгоритм

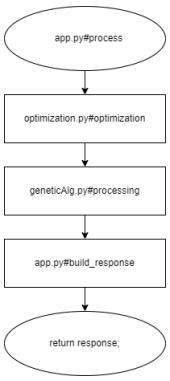

Рисунок 5 – Схема работы веб-сервера над решением задачи

многокритериальной оптимизации. После того, как решение задачи многокритериальной оптимизации будет выполнена, преобразованные данные отправляются в генетический алгоритм. После получения оптимального маршрута обработчик возвращает последовательность узлов и последовательность ребер маршрута.

Алгоритм многокритериальной оптимизации проходит по ячейкам матрицы и исключает из массива в ячейке те рейсы, что не удовлетворяют условию. Таким образом в ячейке остаётся только один вариант перемещения или одно ребро.

Генетический алгоритм работает с матрицей полученной в ходе многокритериальной оптимизации. В процессе выполнения в него передается главный критерий, по которому будет происходить минимизация, то есть по выбранной в первом модуле характеристике.

Полученный результат обработчик готовит к отправке в модуль визуализации данных формируя последовательность ребер с указанием координат точек на карте.

Пятый раздел содержит пример решения задач и проверку зависимости времени выполнения от объема входных данных.

Решим обобщенную задачу для маршрута, состоящего из 10 городов: Саратов, Москва, Самара, Волгоград, Ростов-На-Дону, Нижний Новгород, Тамбов, Воронеж, Санкт-Петербург, Уфа. Рассмотрим данную задачу для случаев когда определены ограничения по всем критериям. В качестве главного критерия на этот раз выберем время, стоимость установим — 7000 денежных единиц; максимальное расстояние — 1200 километров; класс комфор-

та — не ниже 2. В соответствии с таблицей 1, видно, что рейсы были выбраны с классом комфорта равным 2 или 1, что означает комфорт и бизнес-класс. Более того, заметно что рекордное время прибытие показывает самолет. Также цена поездки для каждого из рейсов не превышает установленных ограничений. В соответствии с рисунком 6, маршрут до Санкт-Петербурга строится как можно короче, соединяясь ребрами с Москвой и Нижним Новгородом. Время выполнения — 0,045 секунды.

Таблица 1 – Последовательность маршрутов и их характеристик для решения задачи с ограничениями по всем критериям

№ ребра	Вид транспорта	Время	Стоимость	Расстояние	Уровень комфорта
1	Авто	705.44	2998.12	1199.25	2
2	Авто	361.63	1536.94	614.77	2
3	Авто	143.76	1221.93	244.39	1
4	Поезд	1123.21	6739.28	1123.21	1
5	Авто	307.06	2609.98	522	1
6	Авто	466.86	3968.35	793.67	1
7	Авто	245.39	2085.85	417.17	1
8	Самолет	68.5	4548.44	548	1
9	Авто	656.38	5579.25	1115.85	1
10	Самолет	79.03	5247.48	632.23	1

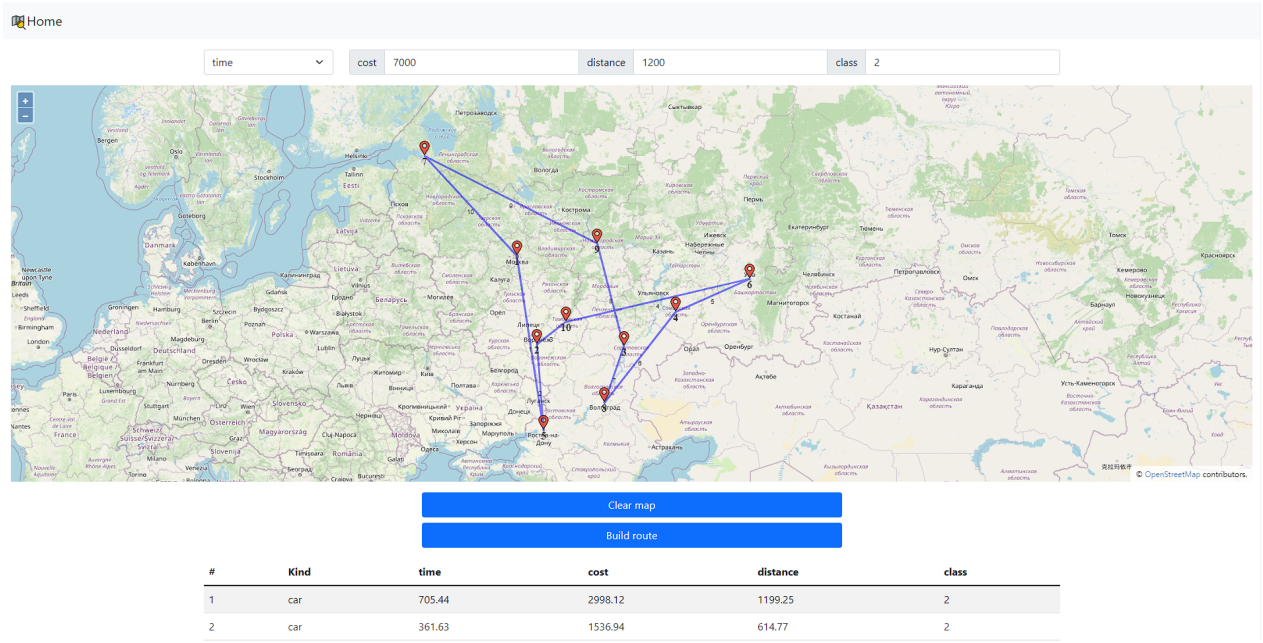

Рисунок 6 – Маршрут для решения задачи с ограничениями по всем критериям

Далее рассматривается время выполнение программы при 10, 25 и 50

точках.

Так как количество итераций фиксировано и равно одной тысяче, делаем вывод, что на время решения задачи будут влиять входные данные. Причиной тому размерность двумерных массивов и количество итераций цикла производимых с целью просмотра его ячеек. Временная сложность обхода двумерного массива должна равняться $O(n^2)$. Для прохода по одномерному массиву сложность выполнения операции составит $O(n)$, где n — число элементов в массиве. Так как в случае обобщенной задачи в двумерном массиве $n \times n$ элементов, то это будет означать, что для каждой итерации для просмотра объекта в массиве необходимо будет запустить еще один цикл на n элементов.

Время выполнения программы для 10 точек составило — 0,035 секунды; для 25 точек — 0,072 секунды; для 50 точек —

В соответствии с рисунком **??**, среднее время построения маршрута для 10 точек с данными ограничениями составляет — 0,035 секунды. 0,14 секунды. В соответствии с рисунком 7, показан результат построения маршрута для 50 точек.

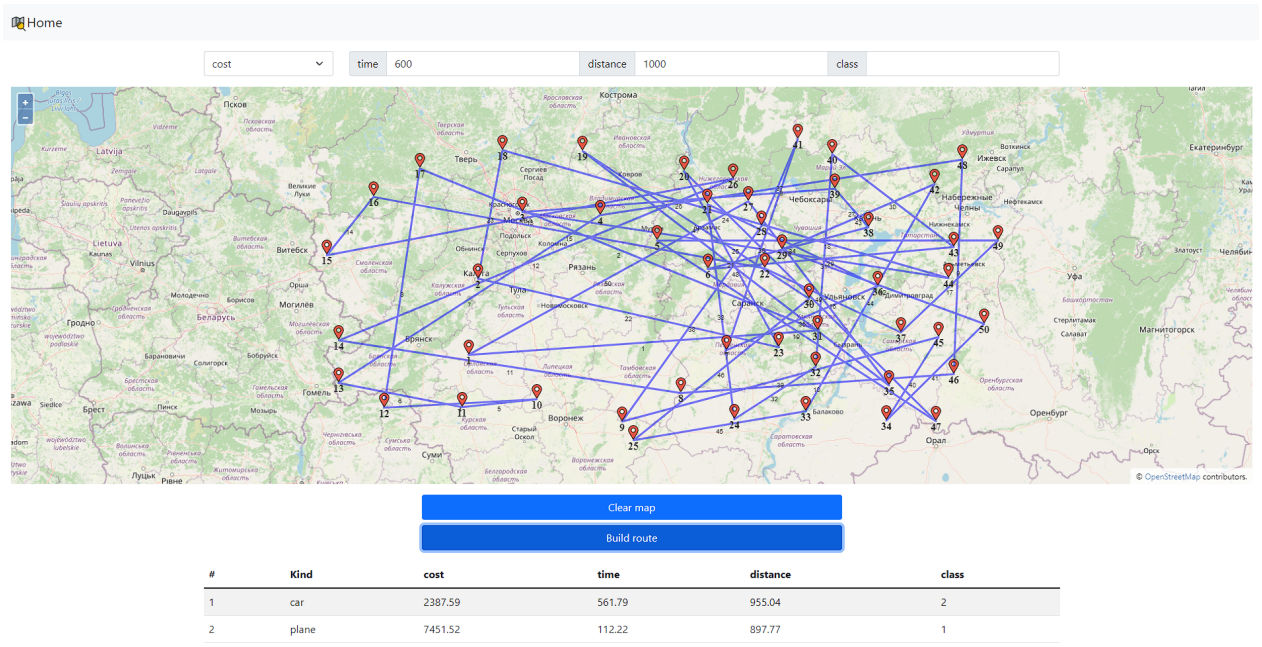

Рисунок 7 – Маршрут для 50 точек

Из полученных результатов можно сделать вывод, что время выполнения растёт линейно. Таким образом зависимость объема данных и время

выполнения — прямая, что характерно для задач такого типа. А это значит, что алгоритм достаточно эффективно справляется с поставленной задачей.

ЗАКЛЮЧЕНИЕ

В рамках данной работы был сформулирован и реализован алгоритм решения обобщенной задачи коммивояжера. Способом обобщения служит наличие ребер графа с многомерной характеристикой. Помимо этого две вершины данного графа могут иметь более одного ребра, что усложняет задачу нахождения гамильтонова цикла. Также существует заинтересованное лицо, устанавливающее критерии оптимизации характеристик маршрута.

Для нахождения решения определён способ приведения условия задачи к виду классической транспортной задачи с более простым видом графа. Путем выполнения многокритериальной оптимизации методом ввода ограничений исключаются ребра не удовлетворяющие условиям. В качестве способа решения уже преобразованной задачи используется генетический алгоритм, позволяющий достаточно быстро найти решение, которое удовлетворяет введённым ограничениям.

Реализовано веб-приложения, в рамках эксплуатации которого была продемонстрирована эффективность сформулированного подхода. Скорость выполнения легко прогнозируется и зависит напрямую от объема входных данных.