МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

РФ КГУ 090303.65.КР.101301107 1

Курсовой проект

По дисциплине «Теория графов и её приложения»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Листов 17

2016

АННОТАЦИЯ

Документ содержит общие сведения о предметной области, примеры алгоритмов и пример создания алгоритма.

# Содержание

Содержание 3

Введение 4

В чем же заключается задача? 4

Задача 4

Вводные данные 4

Последовательность действий 4

Описание метода решения задачи 5

Какую задачу следует решить? 5

Алгоритмы решения задачи 5

Выбор алгоритма решения задачи 5

Описание алгоритма решения задачи 7

Как использовать выбранный алгоритм? 7

Общий план решения задачи коммивояжера 7

Подробная методика решения задачи коммивояжера 7

Обработка входных параметров 11

Граф из категорий 11

Обработка выходных параметров 11

Решение задач 12

Задача #1 12

Задача #2 13

Задача #3 14

Задача #4 15

Задача #5 16

Выводы 17

Введение

В чем же заключается задача?

Задача

Основная задача курсового проекта – разработка ПО, решающего задачу нахождения оптимального пути для экскурсии по заданным категориям объектов.

Данная проблема, очевидно, довольно актуальна в настоящее время. Именно такое ПО поможет решить задачи составления оптимальных маршрутов для экскурсий, развоза товаров в магазины, объезда должников и даже осмотра больных для врачей, которые работают на выезде.

Вводные данные

Список названий категорий объектов, оптимальный маршрут посещения которых необходимо построить.

Пример:

1. Музеи Кургана
2. Церкви Кургана

Или:

1. Магнит Курган

Последовательность действий

1. Изучить алгоритмы решения задачи
2. Выбрать и алгоритмизировать один из них
3. Реализовать готовое ПО, обеспечивающее основной функционал, описанный в первом пункте введения.

Описание метода решения задачи

Какую задачу следует решить?

Очевидно, что для решения поставленной задачи необходимо перенести пункты, которые будут посещены, на граф. В графе, для того, чтобы обеспечить оптимальный маршрут по всем точкам с единичным посещением придется построить Гамильтонов цикл.

***Гамильтоновым*** *циклом* является такой цикл (замкнутый путь), который содержит все вершины (точки) данного графа и проходит по ним ровно один раз.

Для нахождения Гамильтонова цикла в графе решается задача Коммивояжера. Именно эту задачу и необходимо решить.

Алгоритмы решения задачи

Существует перечень алгоритмов решения данной задачи. Однако большая их часть либо очень сложны в вычислительном смысле, либо очень сложны в плане реализации. Список алгоритмов приведен ниже:

1. полный перебор
2. случайный перебор
3. жадные алгоритмы
   1. метод ближайшего соседа
   2. метод включения ближайшего города
   3. метод самого дешёвого включения
4. метод минимального остовного дерева
5. метод имитации отжига

Обычно задачу коммивояжера используют для нахождения заданного пути между городами. Поэтому далее я буду использовать в качестве примера именно города.

Выбор алгоритма решения задачи

Очевидно, что решать задачу методом полного или случайного перебора глупо, ведь мы сталкиваемся с проблемой алгоритмической сложности.

Поскольку коммивояжер в каждом из городов встает перед выбором следующего города из тех, что он ещё не посетил, существует   маршрутов для асимметричной и   маршрутов для симметричной задачи коммивояжера. Таким образом, размер пространства поиска зависит экспоненциально от количества городов.

Различные варианты задачи коммивояжера (метрическая, симметричная и асимметричная) NP-эквивалентны. Согласно распространенной, но недоказанной гипотезе о неравенстве классов сложности P и NP, не существует детерминированной машины Тьюринга, способной находить решения экземпляров задачи за полиномиальное время в зависимости от количества городов.

Также известно, что при условии  не существует алгоритма, который для некоторого полинома   вычислял бы такие решения задачи коммивояжера, которые отличались бы от оптимального максимум на коэффициент.

Однако, существуют алгоритмы поиска приближенных решений для метрической задачи за полиномиальное время и нахождения маршрута максимум вдвое длиннее оптимального. До сих пор не известен ни один алгоритм с полиномиальным временем, который бы гарантировал точность, лучшую чем 1,5 от оптимальной. По предположению, существует (неизвестная) константа, такая, что ни один алгоритм с полиномиальным временем не может гарантировать точность. Как было показано Арора, для евклидовой задачи коммивояжёра существует схема полиномиального времени PTAS для поиска приближённого решения.

Кроме того, решать задачу жадным алгоритмом тоже нельзя, ведь нам нужен **действительно** оптимальный путь.

***Жадный алгоритм***— алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Нас, конечно, какое-то локальное решение не устраивает. Именно поэтому для решения задачи был выбран **метод ветвей и границ**. Основное преимущество данного метода в том, что, имея достаточно времени он может вычислить **действительно** кратчайший путь, а о скорости его работы мы поговорим дальше.

Описание алгоритма решения задачи

Как использовать выбранный алгоритм?

Общий план решения задачи коммивояжера

1. Построение матрицы с исходными данными
2. Нахождение минимума по строкам
3. Редукция строк
4. Нахождение минимума по столбцам
5. Редукция столбцов
6. Вычисление оценок нулевых клеток
7. Редукция матрицы
8. Если полный путь еще не найден, переходим к пункту 2, если найден к пункту 9.
9. Вычисление итоговой длины пути и построение маршрута.

Подробная методика решения задачи коммивояжера

Сначала необходимо длины дорог, соединяющих города представить в виде следующей таблицы:

В нашем примере у нас 4 города и в таблице указано расстояние от каждого города к 3-м другим, в зависимости от направления движения (т.к. некоторые ж/д пути могут быть с односторонним движением и т.д.).

Расстояние от города к этому же городу обозначено буквой M. Также используется знак бесконечности. Это сделано для того, чтобы данный отрезок путь был условно принят за бесконечно длинный. Тогда не будет смысла выбрать движение от 1-ого города к 1-му, от 2-ого ко 2-му, и т.п. в качестве отрезка маршрута.

Находим минимальное значение в каждой строке (di) и выписываем его в отдельный столбец.

Производим редукцию строк – из каждого элемента в строке вычитаем соответствующее значение найденного минимума (di).



В итоге в каждой строке будет хотя бы одна нулевая клетка.

Далее находим минимальные значения в каждом столбце (dj). Эти минимумы выписываем в отдельную строку.



Вычитаем из каждого элемента матрицы соответствующее ему dj.

В итоге в каждом столбце будет хотя бы одна нулевая клетка.

Для каждой нулевой клетки получившейся преобразованной матрицы находим «оценку». Ею будет сумма минимального элемента по строке и минимального элемента по столбцу, в которых размещена данная нулевая клетка. Сама она при этом не учитывается. Найденные ранее di и dj не учитываются. Полученную оценку записываем рядом с нулем, в скобках.  


И так по всем нулевым клеткам:



Выбираем нулевую клетку с наибольшей оценкой. Заменяем ее на «М». Мы нашли один из отрезков пути. Выписываем его (от какого города к какому движемся, в нашем примере от 4-ого к 2-му).  


Ту строку и тот столбец, где образовалось две «М» полностью вычеркиваем. В клетку, соответствующую обратному пути ставим еще одну букву «М» (т.к. мы уже не будем возвращаться обратно).



Если мы еще не нашли все отрезки пути, то возвращаемся ко 2-му пункту и вновь ищем минимумы по строкам и столбцам, проводим их редукцию, считаем оценки нулевых клеток и т.д. Если все отрезки пути найдены (или найдены еще не все отрезки, но оставшаяся часть пути очевидна) – переходим к пункту 9.

Найдя все отрезки пути, остается только соединить их между собой и рассчитать общую длину пути (стоимость поездки по этому маршруту, затраченное время и т.д.). Длины дорог, соединяющих города, берем из самой первой таблицы с исходными данными. В нашем примере маршрут получился следующий: 4 → 2 → 3 → 1 → 4. Общая длина пути: L = 30.

Применение задачи коммивояжера на практике довольно обширно. В частности, ее можно использовать для поиска кратчайшего маршрута при гастролях эстрадной группы по городам, нахождения последовательности технологических операций обеспечивающей наименьшее время выполнения всего производственного цикла и пр.

Обработка входных параметров

Граф из категорий

На входе мы получаем список категорий объектов, маршрут между которыми нам нужно построить. Но что с ними делать? Необходимо получить набор точек для графа. Как?

Я воспользовался Yandex.Maps API для того, чтобы найти объекты, которые меня интересуют, а затем получить их координаты. После получения координат составляем все возможные пары объектов исключая только петли. Дальше вычисляем расстояния между ними объектами в парах (на выходе из API у нас географические координаты, воспользуемся специальной формулой). И последним пунктом строим матрицу графа из полученных пар (у нас есть нужные индексы, полученные при заполнении пар и расстояния, вычисленные на предыдущем шаге).

После этого применяем алгоритм на полученную матрицу и получаем маршрут, который, для наглядности, необходимо нарисовать на карте в виде графа (мы не учитываем расположение дорог, основной задачей было вычислить оптимальный маршрут исходя из расстояний между объектами).

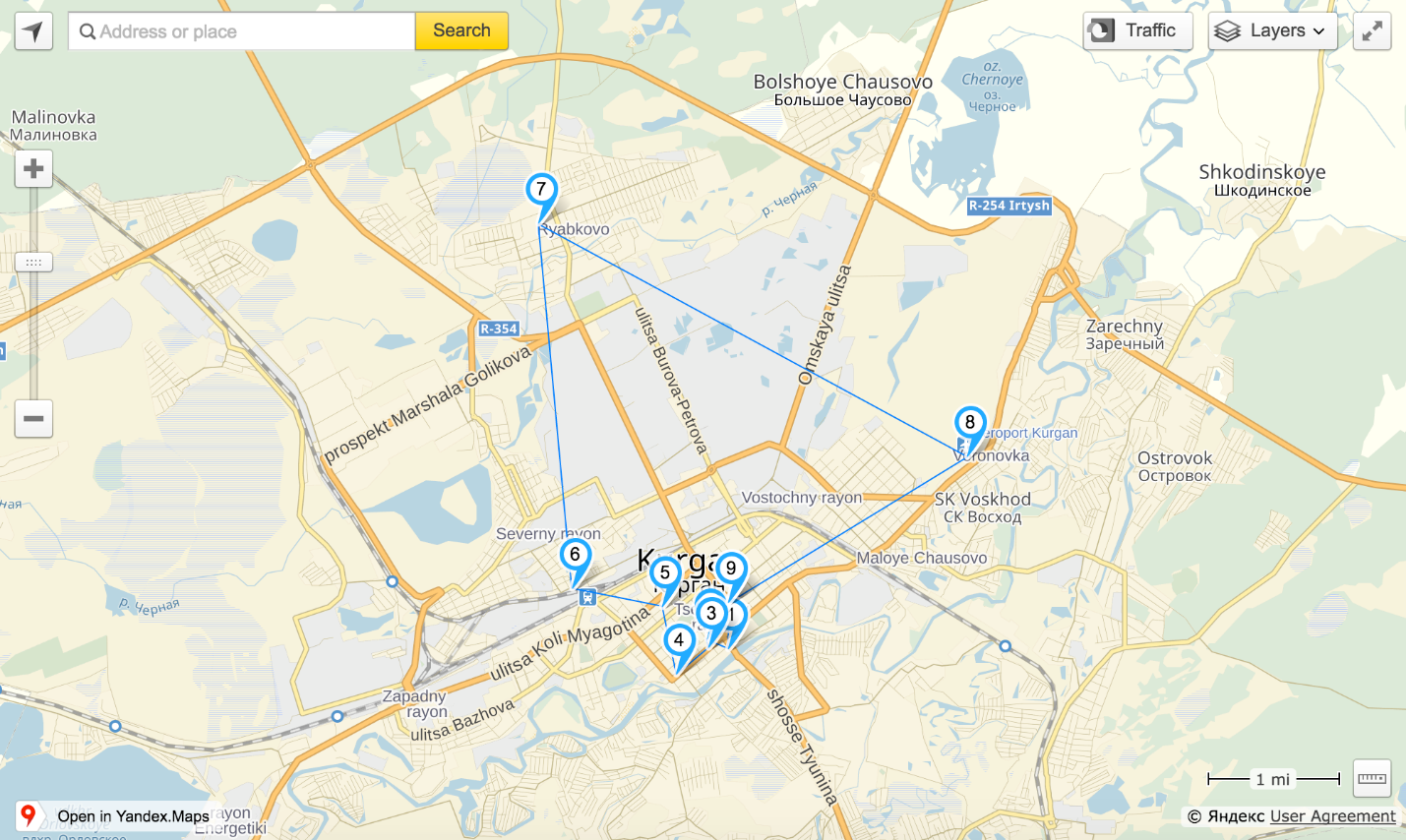
Обработка выходных параметров

На выходе программы мы получим карту с метками, которые между собой соединены ребрами.

Практическое применение ПО

Решение задач

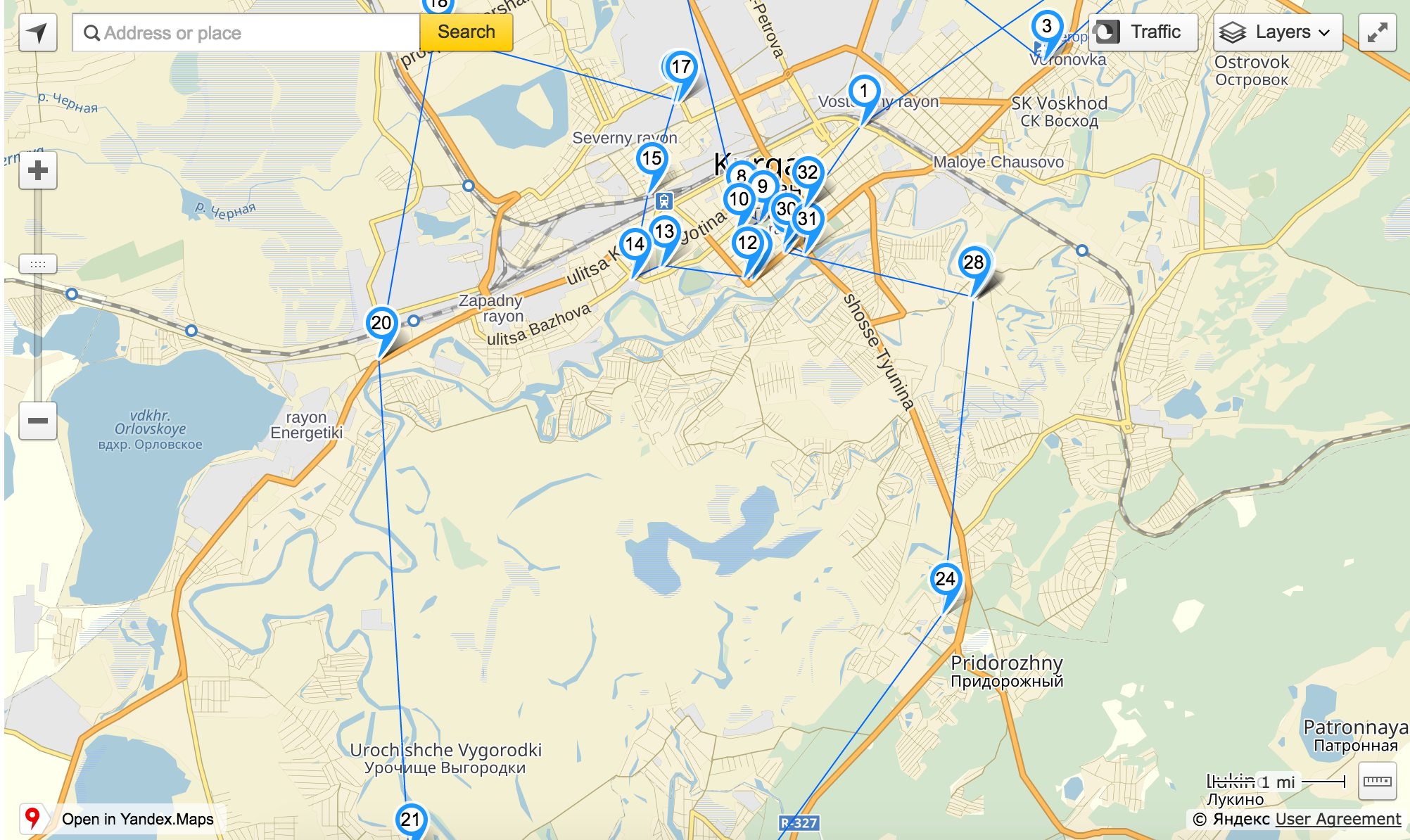
Задача #1

Вычислить оптимальный маршрут для посещения музеев города Кургана.

Задача #2

Усложним задачу, добавим второй параметр поиска.

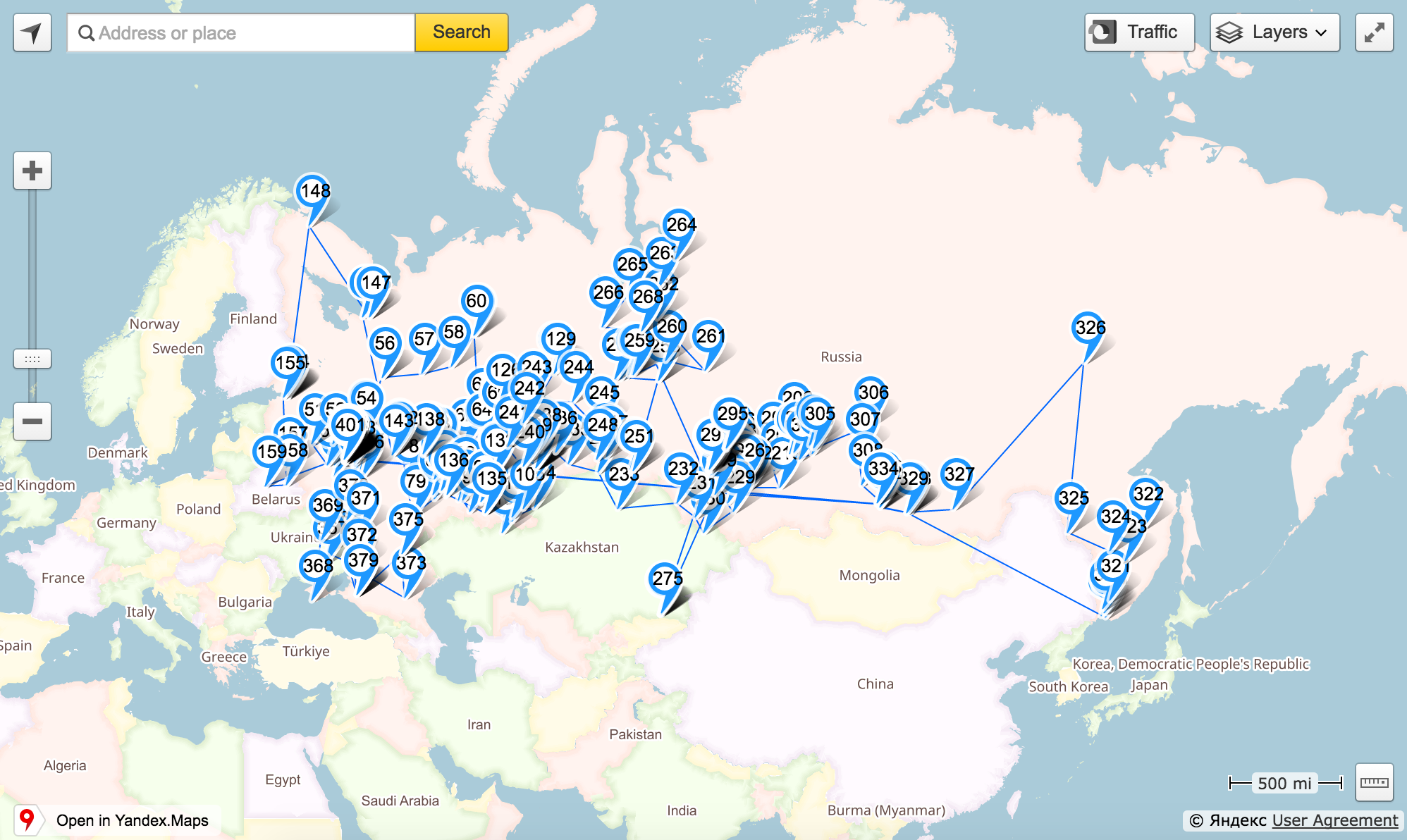
Вычислить оптимальный маршрут для посещения музеев и церквей города Кургана.



Задача #3

Попробуем увеличить вычислительную сложность задачи.

Вычислить оптимальный маршрут для посещения школ России и ближнего зарубежья.

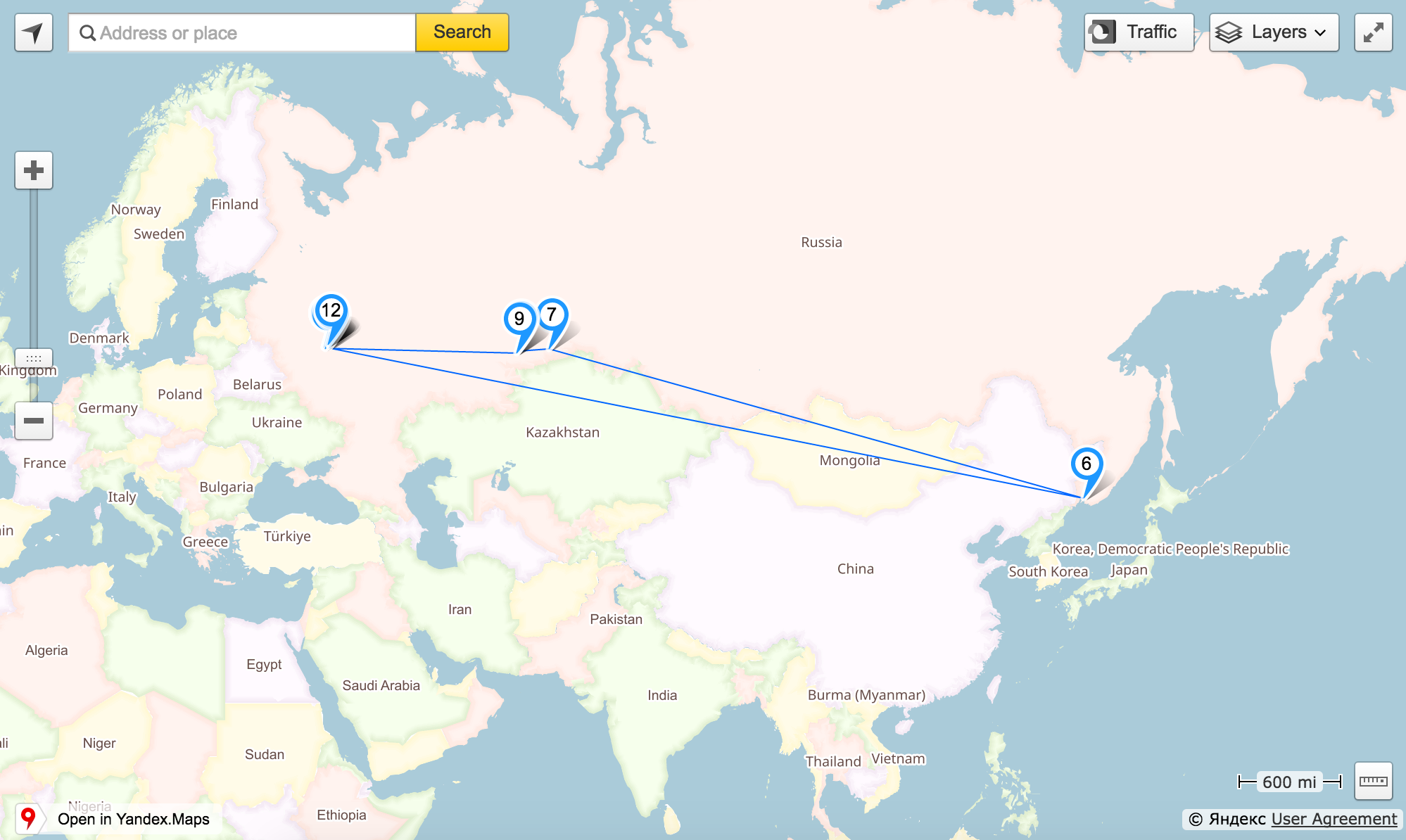


Из-за ограничений Yandex.Maps API – в программе невозможно обрабатывать больше 1000 точек. Именно поэтому не вся Россия, а только регионы, относительно не далеко расположенные от столицы.

Задача #4

Проверим универсальность алгоритма.

Построить маршрут между аэропортами Кургана, Челябинска, Москвы и Владивостока.

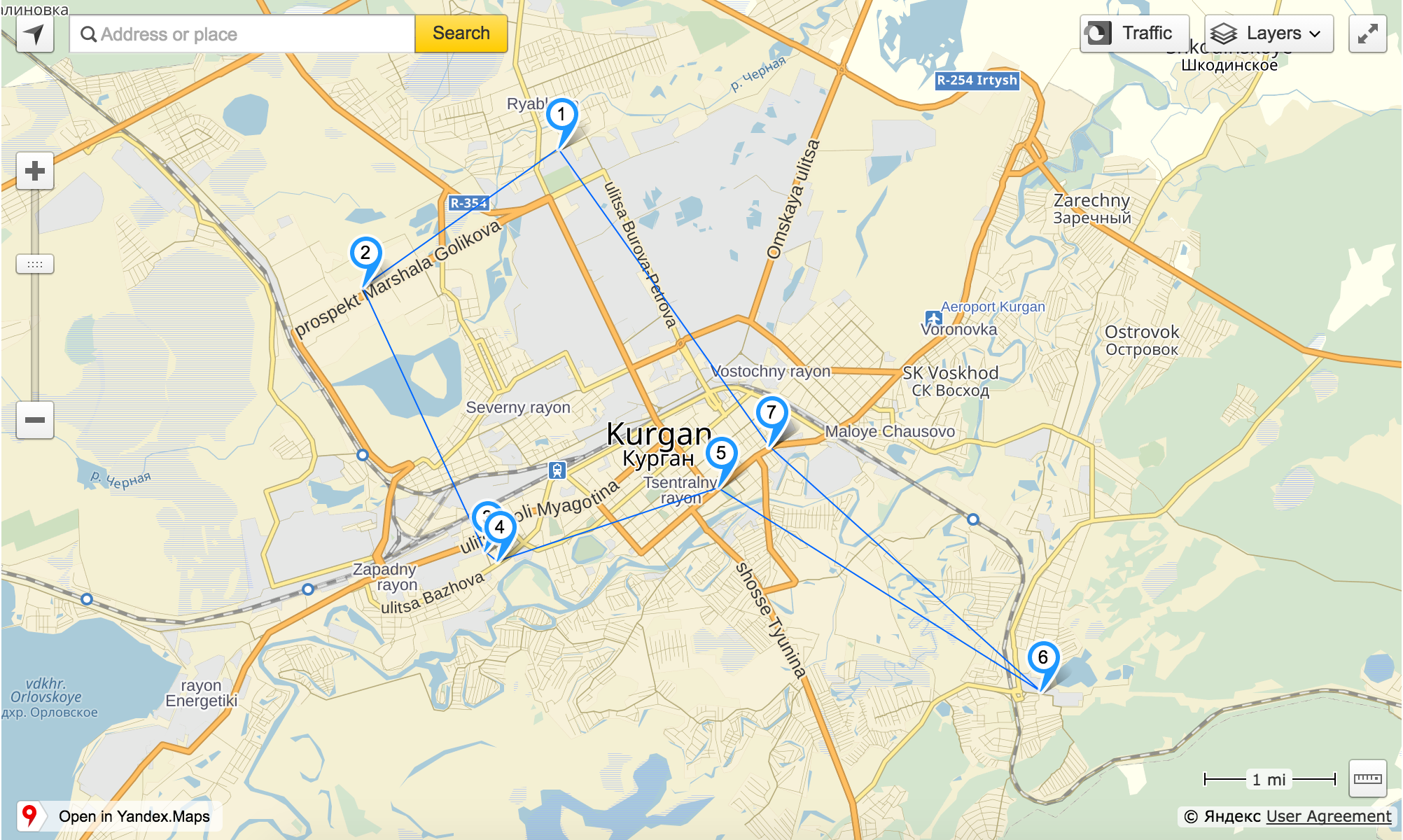


Как можно заметить, алгоритм не делает лишних циклов и посещает каждый город, в том числе, по одному разу, ведь расстояния между городами сильно больше, чем внутри них.

К сожалению, поиск Яндекса не работает со странами дальнего зарубежья (нельзя поискать аэропорты Берлина, например), поэтому я не могу продемонстрировать работу приложения не в странах СНГ.

Задача #5

Проложить маршрут доставки товара в сети магазинов “Магнит” и “Метрополис” города Кургана.



Выводы

В процессе изучения проблемы мной были предложены алгоритмы её решения. Кроме того, был описан алгоритм решающий задачу за приемлемое время.

Задачами предложенной курсовой работы послужили:

1. Изучение алгоритмов решения задачи
2. Выбор и алгоритмизация одного из них
3. Реализация готового ПО, обеспечивающего основной функционал, описанный в первом пункте введения.

Основой доступной модификацией данного проекта считаю добавление расчета расстояний с учетом дорог.