3. ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Р. Н. Алексеев, М. С. Цыганова

Тюменский государственный университет, г. Тюмень

УДК 519.178

РАЗРАБОТКА ИНСТРУМЕНТАРИЯ ПОСТРОЕНИЯ МАРШРУТОВ НА БОЛЬШИХ ГРАФАХ

Аннотация. В статье рассматривается задача построения маршрутов с использованием железнодорожного транспорта. Описан процесс сбора и агрегации данных, необходимых для поиска маршрутов, обоснован выбор оптимизационного алгоритма, представлен алгоритм создания адаптивной таблицы, используемой в качестве входной структуры данных, а также результаты работы этого алгоритма. Обозначены перспективы использования полученных результатов.

Ключевые слова: построение маршрута, поисковый сервис, графовое представление, адаптивная таблица, алгоритм A*.

1. Постановка задачи

В эпоху информационных технологий и развитой транспортной системы решение задачи построения маршрута от пункта А до пункта Б с использованием средств общественного транспорта является достаточно обыденной операцией. Однако, несмотря на широкую востребованность информационной поддержки и автоматизации поиска оптимальных маршрутов, на сегодняшний день можно констатировать практически полное отсутствие удобного инструментария решения подобных задач. Подавляющее большинство поисковых сервисов (например, [1], [2]) предлагают не поиск удобного маршрута, а выбор из уже заготовленных списков маршрутов. В случае отсутствия прямого сообщения (возможности

проезда на одном виде транспорта без пересадок) это часто приводит к появлению сообщения о том, что маршрут не существует.

В качестве примера можно привести запрос на построение маршрута от города Тюмень до города Ирбит по железной дороге. Расстояние между указанными городами составляет 180 км, но прямого железнодорожного сообщения между ними нет. Обратившись к сайту РЖД [1], получим ответ, представленный на рис. 1. Аналогичный ответ дает и сервис *Tymy.py*.

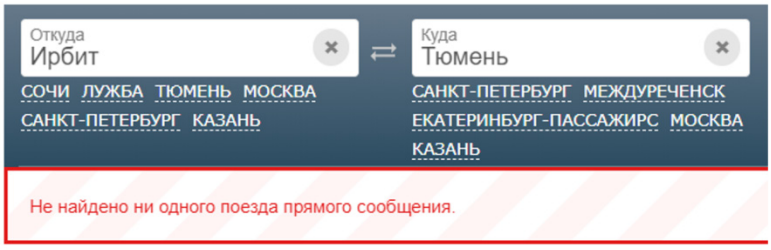

Рис. 1. Сообщение об отсутствии запрашиваемого маршрута на сайте РЖД.

Следует заметить, что в ответе поисковой системы говорится об отсутствии именно прямого сообщения, а не маршрута вообще. Разумное предположение состоит в том, что для любых двух железнодорожных станций общего пользования должен существовать маршрут/маршруты, соединяющие эти станции (возможно с одной или несколькими пересадками). Но доступных инструментов построения таких маршрутов и, тем более, выбора оптимального по какому-либо критерию маршрута, нет. Пользователю придется самостоятельно сформировать и оценить возможные варианты, руководствуясь знаниями географии и информацией, полученной из сторонних источников.

Еще сложнее выглядит задача построения маршрута, предусматривающая использование различных видов транспорта. Например, при формировании маршрута из города Курган в город Ноябрьск разумным выглядит привлечение автобусного сообщения на участке Курган-Тюмень вместо того, чтобы искать железнодорожный маршрут с пересадками,

заведомо неоптимальный по общей длине пути (тем более, что сайт РЖД и в этом случае не предлагает никаких вариантов).

Таким образом, решение задачи построения оптимального маршрута между пунктами А и Б с использованием общественного транспорта представляет большой практический интерес. Оптимальность может определяться различными критериями: общая длина пути, суммарная стоимость, суммарное время, затраченное на преодоление маршрута (включая время между пересадками), количество пересадок и т. п. Решение этой задачи в самой общей постановке (с использованием различных видов транспорта) сопряжено с множеством проблем (включая и объем вычислительных ресурсов). Поэтому в рамках данного исследования было принято решение ограничить задачу только одним видом транспорта – железнодорожным.

Таким образом, была поставлена цель разработать удобный инструментарий поиска оптимального маршрута между любыми двумя выбранными железнодорожными станциями РФ по заданным критериям оптимальности (с учетом возможности использования как поездов дальнего следования, так и электропоездов).

Для достижения поставленной цели необходимо решить ряд задач:

- собрать и агрегировать информацию о маршрутах движения поездов дальнего следования и электропоездов;
- выполнив обработку собранной информации, сохранить полученную модель железнодорожной системы РФ в виде взвешенного графа;
- сформировать графовое представление транспортной системы, учитывающее географические координаты вершин (железнодорожных станций);
- разработать и реализовать алгоритм построения маршрута на полученном графе, оптимального с точки зрения заданного критерия;
- реализовать визуальное представление полученного маршрута.

2. Сбор данных

В качестве инструмента реализации сбора данных была выбрана свободно распространяемая программная платформа Node.js [3], обладающая большой функциональностью, и удобная для разработчика. Немаловажным фактором выбора платформы явилось наличие у нее множества готовых решений и библиотек. В качестве основного источника информации был выбран первоисточник – официальный сайт РЖД – как содержащий наиболее полную и точную информацию.

На этапе получения данных возникло серьезное затруднение: закрытость базы данных РЖД и отсутствие API для разработчиков. Поэтому единственным возможным решением стал прямой парсинг сайта. Для этих целей была использована библиотека node-html-parser [4], которая позволила скачать структуру сайта и выгрузить информацию по ключевым тегам.

Список всех железнодорожных станций был получен с помощью поисковой формы, предлагавшей возможные названия по первым двум введенным символам. Успешное выполнение 1089 запросов обеспечило наполнение словаря в количестве 16 194 станций и остановок.

Полный список всех станций еще не позволяет сформировать какую-то структуру железнодорожной сети. Первым шагом на пути построения такой структуры стало получение для каждой станции ее географических координат. Возможный способ решения этой задачи – использование картографических сервисов. Были проанализированы возможности подключения трех основных сервисов: Яндекс.Карты, Google Maps и 2ГИС, которые позволяют получить информацию об огромном количестве географических объектов РФ, в том числе и о железнодорожных вокзалах. Все указанные сервисы предоставляют API для разработчиков, однако обработка большого количества единовременных запросов (более 500) во всех случаях является платным функционалом. Авторы выражают глубокую благодарность сервису Яндекс.Карты, предоставившему нам возможность единовременной бесплатной обработки 100 тысяч запросов.

Благодаря удобному API [5] были определены географические координаты (широта и долгота) станций, выгруженных с сайта РЖД. В качестве инструментария на этом этапе также использовалась платформа Node.js. По итогам поиска было получено 11 312 станций с достоверными координатами, 4639 станций без координат и 243 станции с явно ошибочными координатами (указывающими на точки, не попадающие на территорию материка Евразия). Для отсеивания станций с ошибочными координатами использовался API сервиса Яндекс.Карты, позволяющий удалить «лишние» (не удовлетворяющие заданному критерию) координаты из словаря.

После всех описанных операций сбора данных было получено 11 312 записей о железнодорожных станциях, содержащих 3 поля: название станции, широта и долгота. Это открыло большие возможности для поиска маршрутов и различных операций с графами. В частности, с учетом информации о существующих маршрутах движения поездов, этого уже достаточно для формирования маршрутов по критерию «минимальная длина».

3. Построение графа

«Лобовой» вариант построения неориентированного графа на основе имеющейся информации не вызывает трудностей. Однако, как показал опыт построения маршрутов на таком графе, использование стандартных алгоритмов в этом случае приводит к серьезным проблемам. В частности, нет механизмов отбрасывания заведомо неприемлемых вариантов (например, маршрута Ирбит-Тюмень длиной более 1000 км), что приводит к огромным затратам ресурсов на анализ всех подобных вариантов. Кроме того, процесс построения маршрута невозможно контролировать. В связи с этим, было принято решение подобрать наиболее эффективный в условиях решаемой задачи алгоритм поиска оптимального маршрута, и только после этого сформировать структуру, на которой выбранный алгоритм будет действительно работать эффективно.

Поиск алгоритма

Одним из наиболее известных алгоритмов построения кратчайшего пути между двумя вершинами графа является алгоритм Дейкстры [6]. Достоинством этого алгоритма является то, что он хорошо работает с нагруженными графами и, при некоторых модификациях, может быть использован для различных критериев оптимальности. Однако в контексте решаемой задачи серьезным недостатком этого алгоритма (как и ряда других классических алгоритмов) является отсутствие «направленности» в поиске: при построении маршрута от Тюмени до Новосибирска будут рассматриваться, в частности, и варианты движения, например, через Москву. Известно [6], что оценка сложности данного алгоритма составляет O(n^2), что при имеющемся значении *n* делает этот алгоритм малопригодным.

Избежать указанной проблемы позволяет применение алгоритма A* – алгоритма поиска пути по первому наилучшему совпадению на графе, который находит маршрут с наименьшей стоимостью от начальной вершины до конечной [7]. Этот алгоритм учитывает «направление» конечной вершины и при обходе вершин всегда движется в ее сторону. Порядок обхода определяется значением функции «расстояние + стоимость». За счет направленности обхода алгоритм A* работает очень быстро, но при этом имеет следующую особенность: он обрабатывает не граф, а двумерный массив. Данный алгоритм и был разработан в первую очередь для быстрых расчетов на очень больших картах с явно заданной структурой (в виде двумерного массива) [7].

Выбором алгоритма A* в качестве основного алгоритма поиска маршрута определился следующий шаг в решении исходной задачи – построение структуры данных, которую будет обрабатывать этот алгоритм.

Построение структуры данных

Для успешного применения алгоритма A* к решению поставленной задачи необходимо представить карту Российской Федерации с известными железнодорожными станциями в виде двумерного массива. Для обеспечения

требуемой точности дальнейших вычислений необходима высокая степень дробления карты (минимальный размер ячейки – порядка 10 $м^2$ реальной площади). При площади территории Российской Федерации около 17,1 млн. $км^2$ такое дробление приведет к созданию массива колоссального размера, причем большая часть ячеек этого массива (соответствующих территориям, расположенным вдали от железнодорожных линий) не будет содержать никакой информации, кроме своей структуры. Отсев таких ячеек не может быть разумным выходом, поскольку в этом случае будет нарушена структура карты, и поиск на такой структуре станет невозможным.

Решением описанной проблемы может стать построение адаптивной таблицы – структуры, в которой размер ячейки варьируется в зависимости от плотности расположения железнодорожных станций: для малонаселенных районов, где количество станций небольшое, а расстояния между станциями значительные, размер ячейки может быть существенно увеличен; наоборот, для густонаселенных районов следует использовать небольшие ячейки. Идея применения адаптивной сетки (или адаптивного уточнения сетки) известна в численном анализе: она используется для построения разностных уравнений, когда требуемая точность вычислений варьируется в различных областях моделируемой области [8]. В настоящей работе предлагается использовать эту идею для построения массива данных о железнодорожных станциях на территории России и СНГ. Далее предлагается алгоритм построения адаптивной таблицы.

- Инициализация: создание двумерного массива, размером 10×10. Первоначальный размер выбран для удобства дальнейшего обращения к элементам массива; он не влияет на ход дальнейших вычислений.
- При добавлении очередной вершины выполнять следующие действия:

1) объявить рассматриваемую область в виде пустого двумерного массива;

2) если ячейка в рассматриваемой области является одиночной, то перейти на шаг 5);

3) если координата вводимой вершины меньше срединной точки, то отсечь правую половину рассматриваемой области, иначе отсечь левую половину рассматриваемой области;

4) перейти на шаг 2);

5) разделить ячейку на две части и записать вершину в область слева, если ее координата меньше срединной точки этой вершины, в противном случае – в правую область.

Эти действия производятся одновременно по двум координатам (x, y). По завершении работы алгоритма обеспечивается большая дробность ячеек в требуемых областях при сохранении прежних размеров изначального массива.

При поиске вершины выполняются те же действия, но на этапе 5) происходит не деление ячейки, а считывание из нее информации.

Реализация описанного алгоритма была выполнена на языке javaScript с нестрогой типизацией и массивами без фиксированного размера [9]. На рис. 2 представлен наглядный пример расстановки вершин и ребер графа. Желтые кружки, связанные между собой – пример маршрута А, красные кружки, связанные между собой – пример маршрута Б.

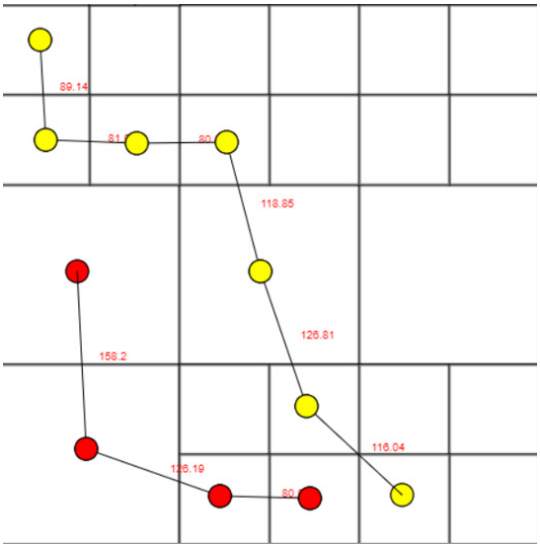

Рис. 2. Пример адаптивной таблицы.

Можно заметить, что в полученной таблице легко определить соседей какой-либо из вершин – достаточно взять ячейку, в которой находится вершина, и обойти соседние ячейки с помощью обычного цикла.

Описанный алгоритм применялся для получения адаптивной таблицы, содержащей карту Российской Федерации с учетом плотности расположения железнодорожных станций. Визуализация полученного результата представлена на рис. 3. Можно заметить, что чем больше плотность станций, тем больше дробление ячеек и, соответственно, более темный цвет данного участка. В районах, где станций не имеется, либо количество их незначительно, соответствующие ячейки занимают огромное пространство. При этом структура карты не теряется и расстояния между вершинами измеряется корректно.

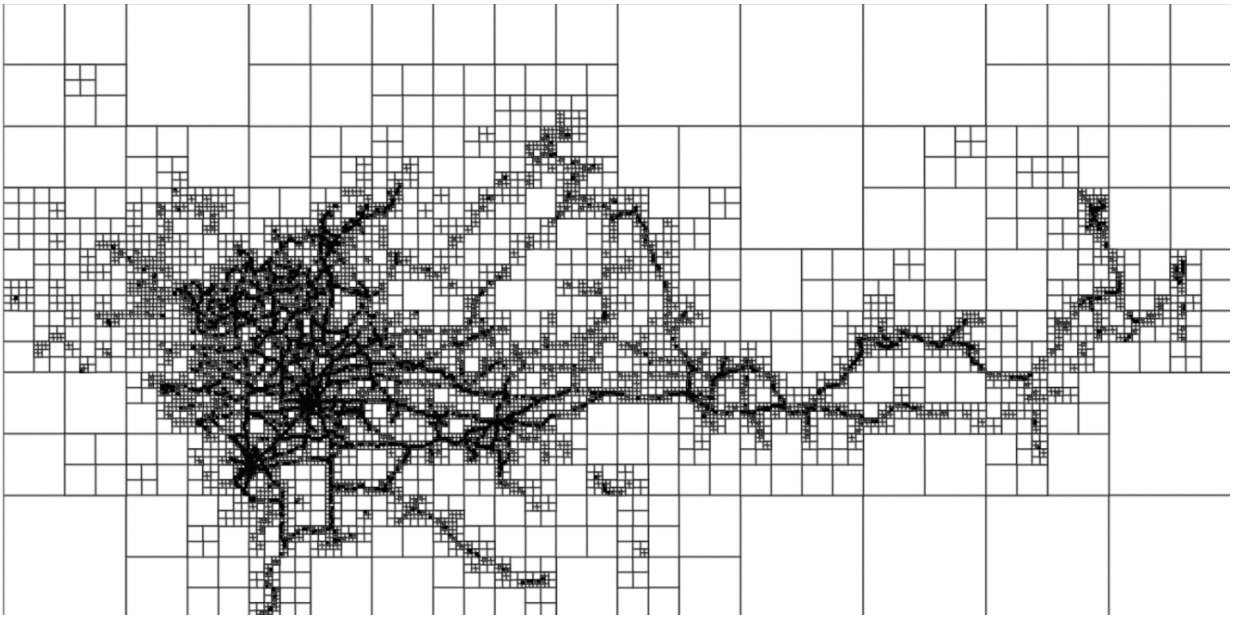

Рис. 3. Адаптивная таблица, содержащая все железнодорожные станции России и СНГ.

Благодаря использованию разработанной структуры данных при дальнейшем построении маршрутов можно будет без затруднений оперировать соединениями вершин и учитывать сразу несколько критериев оптимизации, что позволит существенно сократить объем требуемых вычислительных ресурсов.

4. Подведение итогов

В рамках настоящего исследования

- выполнен сбор и агрегирование данных обо всех железнодорожных станциях России и СНГ;
- обоснован выбор алгоритма, позволяющего с относительно небольшими затратами ресурсов осуществлять поиск маршрутов, оптимальных по одному из заданных критериев;
- разработана и заполнена адаптивная таблица, содержащая информацию для построения оптимальных маршрутов.

Для завершения проекта необходимо

- доработать алгоритм построения оптимального маршрута;
- собрать и агрегировать информацию о стоимости билетов и расписании движения поездов;
- разработать удобный пользовательский интерфейс;
- разместить поисковый сервис на сервере в сети Интернет, проверить устойчивость к нагрузкам и провести стресс-тесты.

СПИСОК ЛИТЕРАТУРЫ

1. Поисковый сервис официального сайта Российских железных дорог: [Электронный ресурс. https://pass.rzd.ru/main-pass/public/ru] (дата обращения: 05.05.2019).
2. Поисковый сервис Туту.ру: [Электронный ресурс. https://www.tutu.ru/] (дата обращения: 05.05.2019).
3. Официальный сайт Node.js: [Электронный ресурс. https://nodejs.org/en/] (дата обращения: 05.05.2019).
4. Документация библиотеки node-html-parser: [Электронный ресурс. https://www.npmjs.com/package/node-html-parser] (дата обращения: 05.05.2019).

5. API Яндекс.Карты: [Электронный ресурс. https://tech.yandex.com/maps/jsapi] (дата обращения: 05.05.2019).
6. Новиков, Ф.А. Дискретная математика для программистов: [учебное пособие для студентов вузов, обучающихся по направлению подготовки дипломированных специалистов «Информатика и вычислительная техника»]/ Ф. А. Новиков. – 3-е изд. – Санкт-Петербург: ПИТЕР, 2009. – 384 с.
7. Описание алгоритма A*: [Электронный ресурс. https://www.redblobgames.com/pathfinding/a-star/introduction.html
8. Berger, M.J., Colella, P. Local adaptive mesh refinement for shock hydrodynamics / J. Comput. Phys. (Elsevier), 1989, 82: pp. 64–84.
9. Учебник по программированию на javaScript: [Электронный ресурс. https://learn.javascript.ru/] (дата обращения: 05.05.2019).