электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 519.17 + 004.2

Построение велосипедных маршрутов

Горшков Н.А., *студент*
Россия, 105005, г. Москва, МГТУ им. Н. Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Степанов В.П., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н. Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»
vapals@yandex.ru

Введение. В последнее время популярность использования велосипедов в качестве средства передвижения или для активного отдыха растет. Поэтому многие велосипедисты нуждаются в сервисах, которые помогут им построить оптимальный, безопасный или просто интересный маршрут. Сервисы построения обычных автомобильных маршрутов [1-3], не позволяют строить велосипедные маршруты, так как в них нет информации о некоторых дорогах, по которым можно проехать на велосипеде, таких как дороги через лес. Такие данные предоставляет open source проект [4], который называется OpenStreetMap.

Постановка задачи. Исходными данными для решения задачи являются данные об электронной карте OpenStreetMap, представленные в виде XML-файла. Из этих данных строится граф дорог, а после этого вводятся исходная точка, конечная точка, некоторое количество промежуточных точек и вид маршрута. Вид маршрута выбирается по двум критериям. Первый критерий – это вид маршрута по типу траектории. Тип траектории маршрута может быть линейным и кольцевым. Линейный маршрут – это маршрут из исходной точки в конечную или через несколько заданных промежуточных точек. Кольцевой маршрут – это маршрут из исходной точки в исходную через несколько заданных промежуточных точек. Второй критерий выбора вида маршрута – это маршрут, соответствующий требованиям велосипедиста. Пользователь может выбрать один из четырех видов маршрута: короткий, безопасный, гоночный или интересный. Например, если пользователь приложения хочет построить маршрут, позволяющий ему доехать до работы, то он скорее всего выберет самый короткий путь, чтобы не опоздать. Если же велосипедист является спортсменом, у которого шоссейный велосипед, то ему необходим

маршрут, который не будет проходит по лесным дорогам, пешеходным тротуарам, поэтому построенный для него маршрут должен проходить по большим дорогам и шоссе. Безопасный маршрут может подойти среднестатистическому велосипедисту, который хочет проехать из одной точки в другую, максимально избегая больших дорог с автомобилями, где многократно повышается вероятность попадания в ДТП. Последний тип маршрута, который является самым интересным подойдет велосипедистам, которые любят путешествовать, посещая популярные достопримечательности. Интересных маршрутов может быть несколько, поэтому задачей настоящей работы является предложение алгоритма, который может найти некоторое множество интересных маршрутов, основой которых будет являться самый оптимальный маршрут. Для решения поставленных задач необходимо выбрать подходящие алгоритмы. Поэтому в настоящей работе анализируются некоторые алгоритмы построения кратчайших маршрутов из точки в точку, а также на основе этих алгоритмов создаются новые для построения безопасных, гоночных и интересных маршрутов, так как не существует готовых решений этих задач.

Анализ предметной области. Сервисов на подобную тематику не так много, так как веломаршрут построить достаточно сложно из-за разных предпочтений пользователей. Из-за несовершенства приложений построения велосипедных маршрутов, наибольшее количество велосипедистов создают маршруты вручную. Но все-таки несколько таких сервисов существует. Все они, в основном, используют данные OpenStreetMap. Наиболее удобным в использовании является сервис [5], так как в нем существует множество настроек маршрута, таких как выбор вида маршрута. Однако, в этом сервисе нет возможности построения наиболее интересного или субоптимального маршрута. Алгоритм построения самого интересного маршрута сделает приложение более полезным, так как большинство велосипедистов хотят путешествовать по самым красивым и запоминающимся местам.

Математическая модель. Множество всех возможных дорог, по которым можно проехать, представляется в виде графа: $G = (V, E)$, где V – множество вершин графа, E – множество ребер графа.

G является ориентированным мультиграфом. Вершинам такого графа соответствуют пересечения дорог, а ребрам – сами дороги. Таким образом, вершина – это точка на карте, которая имеет географические координаты. Ребро является самой дорогой, соединяющей 2 вершины. Каждое ребро имеет несколько свойств: длина, тип дороги,

количество полос, направление движения. Тип дороги необходим для построения безопасных и гоночных маршрутов.

В дальнейшем под $s \in V$ будем понимать начальную точку движения, под $t \in V$ конечную, под $m_i \in V$ – промежуточную точку, где $i \in \overline{1,n}$, n – количество промежуточных точек. Также для упрощения объяснения всех алгоритмов обозначим текущую вершину, в которой алгоритм находится в данный момент как $v \in V$, а метку этой вершины как $f(v)$.

Рассмотрим алгоритмы для реализации задачи построения линейного маршрута.

Алгоритм Дейкстры. Задачу нахождения кратчайшего линейного маршрута можно решить с помощью наиболее известного алгоритма, указанного в [6]. Алгоритм находит кратчайшее расстояние от одной вершины до всех остальных. Для нахождения расстояния до конкретной вершины необходимо просто остановить алгоритм при достижении этой вершины.

Алгоритм работы состоит из следующих шагов. Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до s. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены (или алгоритм дошел до конечной вершины).

Инициализация. Метка самой вершины s полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от s до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

Шаг алгоритма. Если дошли до конечной вершины, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина v, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых v является предпоследним пунктом. Вершины, в которые ведут рёбра из v назовём соседями этой вершины. Для каждого соседа вершины v, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки v и длины ребра, соединяющего v с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину v как посещённую и повторим шаг алгоритма.

Так как алгоритм не имеет представления о том, где находится конечная вершина, то он просматривает все возможные вершины, пока не найдет нужную. Алгоритм просматривает множество вершин, которые ни при каких обстоятельствах не могут находиться на кратчайшем пути от s до t. Это существенно уменьшает производительность этого алгоритма для поиска пути из исходной точки в конечную.

Для оптимизации алгоритма можно использовать свойства, которые присущи дорожным графам:

- Степень вершин дорожного графа редко превышает 5, это означает, что граф является разреженным;
- Дорожный граф является планарным, так как его можно представить на плоскости без пересечения ребер;
- Для каждой вершины дорожного графа известны ее координаты (долгота, широта).
- Существуют более важные дороги и менее важные, каждому виду дороги присваивается приоритет.

Последние два свойства можно использовать для оптимизации алгоритма Дейкстры. Например, при движении между двумя большими городами автомобили обычно двигаются по одной большой дороге без съездов на вспомогательные. Это значит, что можно увеличить метки на вершинах, находящихся на вспомогательных дорогах, это, в свою очередь, уменьшит вероятность захода алгоритма на эти вершины. Также, можно использовать известные географические координаты для подсчета минимального расстояния до конечной вершины, а потом выбирать из всех соседних вершин ту, расстояние от которой до конечной меньше, так как вероятность нахождения этой вершины на кратчайшем пути больше.

Такой подход используется в алгоритме, указанном в [7]. Он использует эвристические функции для поиска кратчайшего расстояния.

Алгоритм A* основан на алгоритме Дейкстры, но он к каждой метке добавляет специальную эвристическую функцию, обозначающую минимальную оценку расстояния от текущей вершины до конечной. В данной работе это расстояние будет равно расстоянию между двумя этими точками на поверхности Земли, так как это расстояние является минимально возможным:

$$cos(d) = sin(\varphi_A) \cdot sin(\varphi_B) + cos(\varphi_A) \cdot cos(\varphi_B) \cdot cos(\lambda_A - \lambda_B)\ ,$$

где φ_A и φ_B— широты, λ_A и λ_B — долготы заданных начальной и конечной точки соответственно, d — расстояние между этими точками, измеряемое в радианах.

Расстояние между точками, измеряемое в километрах, определяется по формуле:

$L = d \cdot R$, где $R = 6371$ км — средний радиус земного шара.

Метки вершин вычисляются по формуле: $f(v) = g(v) + h(v)$, где $f(v)$ – итоговая метка, $g(v)$ – расстояние, уже посчитанное от вершины s до вершины v, $h(v)$ – оценка расстояния от вершины v до вершины t, посчитанное по формуле расстояния между точками на поверхности Земли.

Приведем описание алгоритма, который состоит из следующих шагов:

1. Добавляем начальную вершину s в открытый список.
2. Повторяем следующее:
   a) Ищем в открытом списке вершину с наименьшей стоимостью $f(v)$. Делаем ее текущей вершиной v.
   b) Помещаем ее в закрытый список, удаляем из открытого.
   c) Для каждой вершины, смежной с текущей:
      - Если вершина находится в закрытом списке, то игнорируем ее и переходим к следующей смежной вершине.
      - Если вершина не находится в открытом списке, то добавляем ее туда. Делаем вершину v родительской для этой вершины. Рассчитываем $f(v) = g(v) + h(v)$ для этой вершины.
      - Если вершина уже находится в открытом списке, то проверяем не короче ли будет путь до нее через текущую вершину. Для сравнения используем расстояние $g(v)$. Если сумма $g(v)$ текущей вершины и веса ребра до этой вершины будет меньше, чем $g(v)$ вершины, то меняем родителя этой вершины на текущую вершину.

   d) Останавливаемся, если:
      - Добавили конечную вершину в открытый список.
      - Открытый список пуст, значит мы не нашли путь.

3. Сохраняем путь. Двигаясь назад от конечной вершины, проходя от каждой вершины к ее родителю до тех пор, пока не дойдем до начальной вершины. Это и будет наш путь.

С помощью этого алгоритма можно не только увеличить скорость нахождения кратчайшего пути, но можно его модифицировать для поиска безопасных, гоночных и интересных маршрутов. Модификация алгоритма Дейкстры для поиска множества субоптимальных путей описана в работе [3]. Задачу нахождения безопасного и гоночного маршрутов можно решить, добавив новую эвристическую функцию $p(v)$, которая обозначает приоритет выбора направления движения к данной вершине. Далее предложены алгоритмы, являющиеся модификациями алгоритма A*, для решения задач, описанных выше.

Модифицированный A*. Для выбора значения функции $p(v)$ необходимо рассмотреть структуру данных OpenStreetMap, который предоставляет данные, по которым будет строится граф дорог. Исходные данные представлены в виде XML-файла. Все данные можно условно разбить на три основные группы:

1. Типы данных, описывающие в виде иерархической связи сам объект, как некую пространственную сущность, имеющую свой конечный результат — известные координаты всех частей объекта;
2. Информационная часть — это описательная характеристика объекта, не имеющая к пространственной географической структуре объекта прямого отношения. К этой части относятся его название, физические, логические и прочие свойства;
3. Служебные атрибуты объекта, необходимые для организации процесса хранения и обработки информации в виде набора данных, такие как уникальный идентификатор, состояние объекта в базе, время последней правки объекта в базе и т.д.

Базовых типов всего три: точка (node), линия (way) и отношение (relation).

Точка (node) — это минимальный набор данных, который содержит в себе информацию о паре координат: широта, долгота (lat, lon) и является базовым в иерархической модели. Это единственный тип данных, который хранит саму географическую информацию — координаты, в виде широты и долготы. В дальнейшем мы будем считать, что координаты — это не информационная составляющая объекта точки, а неотъемлемая часть его структуры.

```
<node id='19' lat='58.888047127548994' lon='49.747870758186764' />
```

Линия (way) — это совокупность указателей на объекты типа точка (node). Как минимум, линия состоит из одной точки, т.е. должна содержать как минимум одну ссылку на уже существующий объект типа точка.

```
<node id='23' lat='58.875047918145675' lon='49.785240674006126' />
<node id='22' lat='58.86687448573524' lon='49.737090974777324' />

<way id='24'>
  <nd ref='22' />
  <nd ref='23' />
</way>
```

Тип отношение (relation) рассматривать не будем, так как в данной работе он не нужен.

Тип объекта описывает географические (пространственные) свойства объекта, но ничего не говорит о свойствах самого объекта, его характеристиках, назначении и прочем. Для это существует информационная часть структуры данных OSM, основанная на принципах тегирования объектов, т.е. назначении им определённых меток и указанием свойств этих меток. Теги задаются в виде пары ключ=значение, что в нотации XML для нашей линии 24 выглядит так:

```
<way id='24'>
  <nd ref='22' />
  <nd ref='23' />
  <tag k='highway' v='primary' />
</way>
```

В данном случае добавляется свойство нашей линии, а именно указал тег highway со значением primary, что в принятой схеме тегирования обозначает что линия является основной дорогой, то есть дорогой, которая классом ниже магистрали, но выше второстепенной.

На тегах highway и будет основано вычисление эвристической функции $p(v)$. В OpenStreetMap существует множество значений тега highway. . Каждому тегу будет назначено свое значение, которое будет показывать приоритет посещения, в зависимости от выбранного типа маршрута. Таким образом, если есть несколько вариантов дальнейшего пути, то сначала будут отброшены дороги с наименьшим приоритетом. Предлагаемый метод заключается в том, что во время выполнения алгоритма A* необходимо локализовать выбор дальнейшего маршрута, то есть сначала отбросить все вершины, $f(v)$ которых отклоняется от наименьшей $f(v)$ на определенную величину. Затем отбросить вершины, $h(v)$ которых отклоняется на определенную величину от минимального значения. После локализации решений выбор следующей вершины зависит от новой функции $p(v)$, которая показывает приоритет выбора данного ребра для дальнейшего прокладывания маршрута. $p(v)$ зависит от типа ребра и выбранного вида маршрута.

Двунаправленный A*. Алгоритм A* можно оптимизировать с помощью его одновременного запуска из исходной и конечной точек. Поиск пути заканчивается тогда, когда некоторая вершина будет посещена из обоих направлений, то есть полученный оптимальный путь будет проходить через эту вершину.

Таким образом, благодаря возможности современных компьютеров распараллеливать процессы, можно получить увеличение скорости нахождения пути 2 раза по сравнению с обычным алгоритмом. Благодаря этому, такой алгоритм является достаточно эффективным, поэтому для задачи поиска кратчайшего расстояния из одной точки в другую целесообразно выбрать именно его, двунаправленный модифицированный алгоритм A*.

Заключение.

1. Формализована задача построения велосипедных маршрутов в виде задачи на графах.

2. Предложены модификации существующих алгоритмов для построения маршрутов разных типов, таких как безопасные, гоночные и наиболее интересные маршруты, а также маршрутов, являющихся субоптимальными.

Список литературы

1. Google Maps. Режим доступа: https://www.google.ru/maps/ (дата обращения 02.03.2015).
2. Яндекс Карты. Режим доступа: http://maps.yandex.ru/ (дата обращения 02.03.2015).
3. Степанов В. П. Оптимизация маршрутов на дорожной сети // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 5. Режим доступа: http://technomag.bmstu.ru/doc/369475.html (дата обращения 10.03.2015).
4. Open Street Map. Режим доступа: http://www.openstreetmap.org/ (дата обращения 05.03.2015).
5. Open Route Service. Режим доступа: http://openrouteservice.org/ (дата обращения 12.03.2015).
6. Dijkstra E.W. A Note on Two Problems in Connexion with Graphs. Available at: http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf, accessed 15.03.2015.
7. Hart P.E., Nilsson N.J., Raphael B. A Formal Basic for the Heuristic Determination of Minimum Cost Paths. Available at: http://ai.stanford.edu/~nilsson/OnlinePubs-Nils/PublishedPapers/astar.pdf, accessed 17.03.2015.