GPS Navigation Algorithm Based on OSM Data

Daniel Khachay

Ural Federal University, Yekaterinburg, Russia
daniil.khachay@gmail.com

Abstract. A new pedestrian GPS navigator providing the shortest-cost safest-crossing route on the basis of Open Street Map (OSM) cartographic data is proposed. Also, Java implementation and use case example are discussed.

Keywords: GPS-navigation application, OSM data retrieval, shortest-path search algorithm

1 Introduction

Satellite navigation algorithms are used everywhere in modern life. Every computing system, even a smartphone, is equipped with some kind of navigation application (at least, Google Maps). Such an application is able to build a route from one point to another, show it on a map, etc. Each navigation application can be proprietary or open-source. Among wide variety of open-source projects, Open Street Map (OSM) project seems to be the most interesting. I've decided to study this format in more details. I know that the best way to understand a new technology better is to apply it for something useful. I'm fond of walking the streets of my city. So I decided to develop a simple pedestrian navigator based on OSM data.

2 Problem Statement and Related Works

Each navigation application contains implementation of some routing algorithm as it's main building block.

General routing problem has the following setting. Input: starting and target location points given by their GPS coordinates and topographic map of the search area, defining restrictions over feasible routes. The goal is to determine the optimal (shortest w.r.t some predefined metric) route.

For instance, a car driver navigator constructs a minimum trip-time route subject to given road map and traffic constraints.

Traditional approach to mathematical solution of the above problem consists of two stages. On the first stage, the initial problem is reduced to well-known Shortest Path Problem (SPP), which is defined on the appropriate weighted graph. On the second stage, SPP is solved by one of classical combinatorial optimization algorithms: Dijkstra [1] or it's heuristic extension - the A^* [2].

Therefore, any navigation application differs from another only by the following features:

(i) setting of the initial problem (car driver navigation, bicycle navigation, etc.),
(ii) method of the reduction to SPP,
(iii) format of cartographic data.

Functionality (hover for description)	OpenRoute-Service [1]	YOURS [2]	CycleStreets [3]	Cloudmade	Routino [4]	BBBike @ World [5]	MapQuest [6]	OSRM [7]	TripGo [8]	BRouter	OpenTrip-Planner [9]	HoofMarker	GraphHopper [10]
Coverage	Europe only	Global	UK only	Global	UK only	Selected Cities	Global	Global	Selected Cities	Global	Global	Germany	Global
Modes of transportation													
Car (fastest)	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Experimental	No	No	Yes
Car (shortest)	Yes	Yes	No	Yes	Yes	No	Yes	No	Yes	Experimental	No	No	No
Bicycle (shortest)	Yes	Yes	Yes	No	Yes	Yes	No	No	Yes	Yes	Yes	No	No
Bicycle (fastest)	Partial	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes
Bicycle (safest)	Yes	No	No	No	Yes[1]	Yes	No	No	No	Yes	Yes	No	No
Bicycle (quietest)	Safest	Partial[2]	Yes	No	Yes[1]	No	No	Yes	No	Yes	No	No	No
Pedestrian	Yes	Yes	No	Yes	Yes	No	Yes	No	Yes	Shortest	Yes	Yes	Yes
Pedestrian (safest crossings)		No	No		No	No	No	No	No	No	No	No	No

Fig. 1. The comparison matrix taken from the OSM project's official site [6]

.

Nowadays, there are many open-source applications based on OSM data format. Among them, the following applications seem to be the most popular:

a **Open Source Routing Machine** is a nice online routing application. It's seems to be [3] mostly fast and precise car driver navigator ever. Unfortunately, it have no standalone version and provides no services for bicyclers and pedestrians.
b **CycleStreets** is a mobile routing application for iOS and Android platforms [4]. The iOS version of the application is seems to be the most valuable for bicyclers. However, it operates only in United Kingdom and has no services for pedestrians like the previous application.
c **GraphHopper** is Java implemented cross-platform multi-service routing application [5], which seems to be the most interesting. The application provides simultaneously a car driver, a bicycler and a pedestrian navigation services. But the pedestrian navigator provides no a safest-crossings routing.

As can be seen from Fig. 1 there is no OSM-based routing application providing full service to pedestrians (no safest-crossings support). In this paper we describe our OSM-based Java implemented standalone application, providing this type of service.

Let us recall some basics of the OSM format structure. First of all, OSM file [7] is just a special type of an XML document and contains hierarchical collection of elements. Some of these elements may have attributes and additional data. Basic elements of any OSM file are called *nodes* and *ways*. A *node* is just a model of some location point, defined by geographic coordinates (latitude and longitude). In the OSM format, each spacial topographical object is presented

```
<way changeset="7806002" id="39240803" timestamp="2011-04-08T17:25:58Z"
uid="133332" user="AMDmi3" version="5" visible="true">
  <nd ref="470133843"/>
  <nd ref="804028626"/>
  <nd ref="470133865"/>
  <tag k="highway" v="pedestrian"/>
  <tag k="name" v="Tolmacheva st."/>
</way>
```

Fig. 2. An example of the pedestrian way description in OSM format

by some *way* element containing a collection of *nodes* and accompanied by informational elements. Each informational child element can be considered as a (key, value)-pair, presenting some feature of the containing *way* element (see Fig. 2).

3 Results

3.1 Reduction to the Shortest Path Problem in Weighted Graph

As we've seen above, in the OSM format, every topographical object (street, bicycle path, footpath, building, etc.) is presented by a piecewise linear way consisting of nodes. Here's an another reason for usage exactly the OSM format for the construction of the graph corresponding to the current navigation problem.

During the reduction to SPP, these nodes are just taken as vertexes of the constructed graph. Further, we assume two vertexes to be adjacent if they correspond to neighboring nodes of the same way on the map.

Because of our intention to develop the pedestrian (safest crossing) navigation application, we should consider only such ways, that describes special types of roads, among them footways, sidewalks, pedestrian crossings, etc. During the graph construction we use only these ways. When the weighted graph is constructed, we apply an heuristic extension of well-known Dijkstra algorithm - the A* algorithm [2] to construct a minimum cost path.

3.2 Java Implementation

We implement the mentioned above application in object-oriented Java application[8]. Let's describe the main classes. Main - the main class that runs an application. Footway - the "footpath" class, it contains an array of nodes from OSM file. AStar - the class that implementing an A* search algorithm. MapReader - this class is intended for parsing OSM file. RouteWriter - this class appends the calculated route (in osm-file) in XML-format so the augmented map could be visualized. WeightedPoint - this class determines the coordinates of nodes.

3.3 Example of Application Usage

Suppose, we are asked to construct a shortest pedestrian path from the main building of Institute of Mathematics and Computer Science (IMCS) UrFU to the

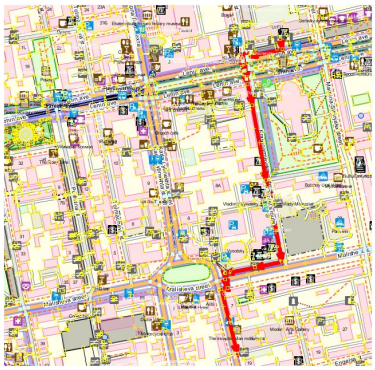

Fig. 3. The shortest pedestrian (safest crossings) path from IMCS UrFU to Belinsky public library is found and red-highlighted

main entrance of Belinsky public library. First we need to know GPS-coordinates of both way-points. Second we need a special OSM file that contains required piece of map. Then using the AStar class, our application constructing a shortest path and using a RouteWriter class, application write down an XML file with the shortest pedestrian route. To get the constructed route, we can open this updated XML file by any text-reading application. In our case (see Fig.3), we use JOSM (Java Open Street Map, Java implemented OSM editor) for graphical visualization.

We conduct a specific numerical experiment consisting of constructing of 100 routes for independently chosen random location points on the Ekaterinburg city map. Expected relative value of graph construction run-time is equal to 97.8% of total run-time within standard deviation of 0.3%.

4 Conclusion

A new type of OSM-based routing application for constructing shortest-cost safest-crossing pedestrian paths is proposed. The application is Java-implemented and can run on every Java-compatible platform. Run-time of the application can be significantly reduced by leveraging some of standard cashing techniques for a previously constructed graph.

References

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms (3-rd edition). MIT press. 2009.
2. Judea Pearl. Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley. 1984.
3. Open Sourse Routing Machine application page. http://map.project-osrm.org/

4. CycleStreets application page. http://www.cyclestreets.net
5. GraphHopper application page. http://graphhopper.com
6. OSM-based routing applications comparison matrix. http://wiki.openstreetmap.org/wiki/Routing/online_routers
7. Jonathan Bennett. OpenStreetMap. Packt Publishing. Birmingham. 2010.
8. Daniel Y. Liang. Introduction to Java programming: comprehensive version. Pearson Education, Inc. 2007.

Алгоритм GPS навигации по данным OSM

Даниил Хачай

Уральский федеральный университет, Екатеринбург, Россия
daniil.khachay@gmail.com

Аннотация Предлагается новое приложение — GPS-навигатор, находящее кратчайший маршрут с учетом правил дорожного движения по данным OpenStreetMap (OSM). Обсуждаются программная реализация на языке Java и пример использования.

Ключевые слова: GPS-навигатор, обработка OSM данных, алгоритм кратчайшего пути.