NANYANG TECHNOLOGICAL UNIVERSITY

SCE16-0446
Time-Dependent Shortest Path Queries on Mobile Devices

Submitted in Partial Fulfillment of the Requirements
for the Bachelor of Computer Science
of the Nanyang Technological University

by

Wei Yumou

School of Computer Science and Engineering
2017

SCE16-0446

Time-Dependent Shortest Path Queries on Mobile Devices

by

Wei Yumou

Submitted to the School of Computer Science and Engineering
on 27 March 2017, in partial fulfillment of the
requirements for the degree of
Bachelor of Computer Science

Abstract

In this thesis, I designed and implemented a compiler which performs optimizations that reduce the number of low-level floating point operations necessary for a specific task; this involves the optimization of chains of floating point operations as well as the implementation of a "fixed" point data type that allows some floating point operations to simulated with integer arithmetic. The source language of the compiler is a subset of C, and the destination language is assembly language for a micro-floating point CPU. An instruction-level simulator of the CPU was written to allow testing of the code. A series of test pieces of codes was compiled, both with and without optimization, to determine how effective these optimizations were.

FYP Supervisor: Xiao Xiaokui
Title: Associate Professor, Assistant Chair (Strategic Research)

Acknowledgments

I would like to express my special thanks of gratitude to my FYP supervisor (Assoc Prof. Xiao Xiaokui) who gave me the golden opportunity to do this wonderful project on the topic (GPS Trajectory Mining), which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them. Secondly i would also like to thank my parents and friends who helped me a lot in finalising this project within the limited time frame.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

Finding a practically shortest route on a large road network in a metropolis is not only of algorithmic interests, but also of economic and environmental values. Less travelling time means less fuel consumptions and less carbon emissions. However, finding shortest routes can be challenging, especially when the road traffic is known to be *time-dependent* or *dynamic*, namely, when the road conditions change with respect to time. It may take 10 minutes on average to traverse a particular road at 10a.m, but it is possible that the expected travel time increases to 20 minutes at 5p.m. Moreover, two different roads may have different time-varying patterns. For instance, one road may have a peak traveling time at 12p.m but the other may have two peaks at 8 a.m. and 6 p.m., respectively.

The formal definition of a dynamic road network is given as follows.

Definition 1 (*Dynamic road network*)**.** A dynamic road network is a weighted, directed graph $G = (V, E)$ where E represents a set of road segments and V denotes the set of intersections of these road segments. It has a weight function $w : E, t \to \mathbb{R}$, where t represents an instant in time.

With the definition of a dynamic road network, the generalised time-dependent

shortest path problem can be formally defined as follows.

Definition 2 (*Generalised time-dependent shortest path problem*)**.** In a dynamic road network $G = (V, E)$, given a source node u, a destination node v and a departure time t from u, find a path p that satisfies:

$$w(p) = \delta(u, v) = \begin{cases} \min \left\{ w(p) : u \overset{p}{\rightsquigarrow} v \right\} & \text{if there is a path from } u \text{ to } v, \\ \infty & \text{otherwise.} \end{cases} \tag{1.1}$$

where $w(p)$ is the weight of the path p and defined as sum of the weights of its constituent edges, and $\delta(u, v)$ is called the **shortest-path weight** from u to v.

A typical Bellman-Ford[1] or Dijkstra's algorithm[2] for finding shortest paths assume the cost of traversing each edge in the abstract graph is constant with respect to time and therefore, do not work on time-dependent road networks without appropriate modifications. Fortunately, most online mapping services such as Google Maps or Apple Maps are able to recommend shortest routes by incorporating real-time traffic information. This project seeks to investigate an alternative approach of finding shortest routes on a dynamic road network based on mining a GPS[1] trajectory database aggregated from thousands of taxis in Beijing, China.

Chapter two describes the preliminary data processing. Chapter three introduces the approach for building a landmark graph. Chapter four explains how to estimate the travel time for each landmark graph edge. Chapter five presents methods for evaluating travel time estimations.

[1] Global Positioning System

1.1 Motivations for mining taxi GPS trajectories

Taxi drivers or any experienced car drivers, more often than not, possess some *implicit* knowledge or intuitions about which route from source u to destination v is the best in terms of travelling time at a particular moment. Such knowledge or intuitions come from everyday experiences. For example, a taxi driver may observe that there are always traffic jams during 6 p.m. to 7 p.m. on a particular street and hence avoids travelling on that street during that period whenever possible. But observations of this kind, albeit valuable, are just too subtle to be captured by any general algorithms and oftentimes, even the drivers themselves may not be aware of that.

However, mining their GPS trajectories can reveal such knowledge to some extent. In a metropolis such as Beijing or New York, taxi drivers are required by regulations to install GPS devices on their cars and to send time-stamped GPS information to a central reporting agency periodically for management and security reasons. Such information typically includes latitudes, longitudes, instantaneous speeds and heading directions. Therefore, the GPS data is readily available and little effort is needed to collect it. By means of mapping to a real road network all GPS data points of a particular taxi during a specific period of time, a GPS trajectory can be obtained to represent the driver's intelligence.

1.2 Practical limitations

There are some practical limitations that are worth mentioning.

Arbitrary sources and destinations

In a typical map-query use case, a user is able to select an arbitrary source to start with and an arbitrary destination to go to. But this may not be possible in the GPS-

based approach, since the taxi GPS trajectories do not necessarily cover every part of a city's map. It is likely that there are no trajectories passing through the source and the destination.

Low sampling rate

Taxis report their locations to the central reporting agency at a relatively low rate to conserve energy. For the data set used in this project, the expected sampling interval is one minute. But oftentimes, the GPS device may not be working properly or may be occasionally shut down due to various reasons, which causes the actual sampling interval to fluctuate.

Even if the sampling interval *is* strictly kept at one minute, for a taxi moving at a typical speed of $60km/h$, it means the distance between two consecutive sample points is $1km$. Such a large distance increases the uncertainty of the *exact* trajectory that the taxi has moved along. The picture[8] below demonstrates a problem caused by low sampling rate and long inter-sample-point distance.

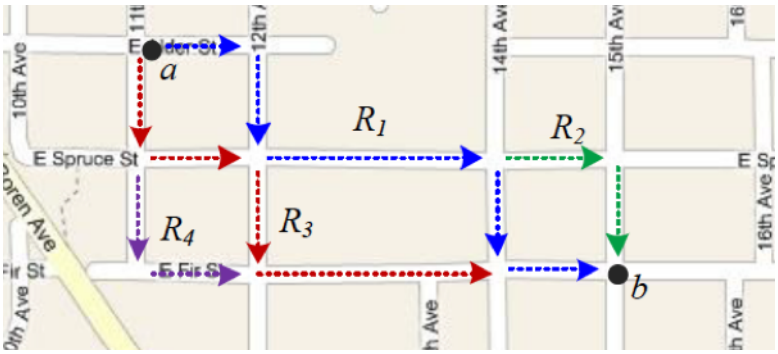

Figure 1-1: Example of low sampling rate problem

The taxi is known to have traversed from point a to point b. But there are four possible trajectories from a to b. The exact route cannot be determined without additional information in this case.

Limited GPS accuracy

After decades of development, the GPS service has achieved great accuracy, but it is still not completely error-free. A report[6] in 2015 showed that GPS-enabled smartphones typically have an accuracy of 5 metres *under open sky*. But in a metropolis like Beijing, the actual accuracy may be lower than this value due to the reflection of signals amongst high buildings. Moreover, the data set used in this project was collected in 2009 when GPS devices had lower accuracy than that of today's.

The limited accuracy in GPS devices makes the exact mapping from a GPS data point to a street impossible. In Beijing, there is usually a side road running in parallel with a main road. Due to that limited accuracy, the taxi might be *actually* on the side road but the GPS data point is shown on the main road, or vice versa.

1.3 Related work

The incentive for carrying out this project comes from a similar project[8], and similar procedures are followed in this project but with some modifications.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Preliminary Data Processing

2.1 Data Collection and Cleaning

The taxi GPS data used in this project is collected from the Computational Sensing Lab[9] at Tsinghua University, Beijing, China. The data set contains approximately 83 million time-stamped taxi GPS records collected from 8,602 taxis in Beijing, from 1 May 2009 to 30 May 2009. The original data set consists of seven fields as shown in Table 2.1. Longitude and latitude in the data set are defined in the WGS-84[1] standard coordinate system, which is the reference coordinate system used by the GPS.

The original data set came in a binary file format. After the data set was decoded and imported into a MySQL database, the first step in data cleaning was **to delete all records with zero value in the SPEED field**, since when a taxi is stationary it yields no valuable information about the *trajectory* it is moving along. While being stationary could be due to a traffic jam, this kind of information is well captured by the time difference between the last *non-stationary* data point and the next *non-*

[1]World Geodetic System

Field	**Explanation**
CUID	ID for each taxi
UNIX_EPOCH	Unix timestamp in milliseconds since 1 January 1970
GPS_LONG	Longitude encoded in WGS-84 multiplied by 10^5
GPS_LAT	Latitude encoded in WGS-84 multiplied by 10^5
HEAD	Heading direction in degrees with 0 denoting North
SPEED	Instantaneous speed in metres/second (m/s)
OCCUPIED	Binary indicator of whether the taxi is hired (1) or not (0)

Table 2.1: Fields in the original data set

stationary data point.

In addition, all records must have a *unique* pair of CUID and UNIX_EPOCH fields, since it is not possible for a taxi to appear in two different locations at the same moment in time. This kind of error is likely due to some errors in aggregating the original data set.

2.2 Reverse Geocoding

After the data set was cleaned, the next step was to map each GPS data point to a road segment, which is also known as *reverse geocoding*. A number of algorithms[4] have been proposed for this purpose, but most of them require an additional GIS[2] database of the road network in Beijing. This project adopted an alternative strategy which leveraged on the existing public APIs[3] for reverse geocoding.

Currently, a number of online mapping platforms provide reverse geocoding services as part of their developer APIs. Amongst others, Google Maps and Baidu Maps

[2]Geographic Information System
[3]Application Programming Interface

offer relatively stable and fast reverse geocoding services. However, due to the "China GPS shift problem"[7] where coordinates encoded in WGS-84 format are required by regulations to be shifted by a large and variable amount when displayed on a street map, Google Maps is not able to display a GPS data point correctly because it only supports WGS-84 formats. Figure 2-1 demonstrates the effect of such shift, with the correct location displayed on the right.

Figure 2-1: China GPS shift problem

Baidu Maps, on the other hand, has been using their own coordinate system, BD-09, which is an improved version of the Chinese official coordinate system, GCJ-02. Baidu provides a set of APIs to convert WGS-84 coordinates into BD-09 ones. Therefore, to reverse-geocode the data points, the coordinates must be converted to BD-09 format. To store the converted coordinates as well as the street names obtained from reverse geocoding, four new fields were added to the original data set as shown in Table 2.2.

In order to use Baidu APIs for coordinate conversion, the following system architecture was set up as shown in Figure 2-2. The Apache HTTP server hides the MySQL database and sends HTTP POST request to Baidu Maps Web API to get

Field	**Explanation**
DataUnitID	Nominal primary key for each record
BD09_LONG	Longitude encoded in BD-09 format
BD09_LAT	Latitude encoded in BD-09 format
STREET	Street name

Table 2.2: Additional fields added to data set

converted coordinates. Then it updates the database through PHP *mysqli* utilitity.

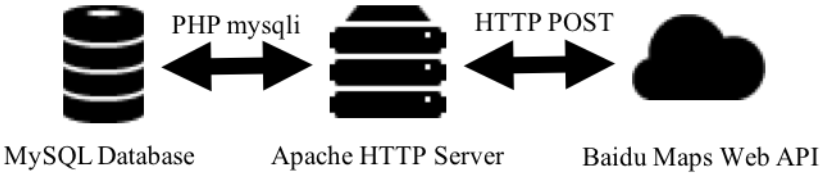

Figure 2-2: Basic system architecture

After the coordinates were converted from WGS-84 format to BD-09 format, Baidu Maps Web API was used to reverse geocode all GPS data points. However, the system architecture was slightly changed, to accommodate the change in technology used. For reverse geocoding, AJAX[4] was used to communicate with the Baidu Maps Web API for speed and unlimited number of requests per day. Therefore, one additional layer was added to the existing system architecture as shown in Figure 2-3.

Executed in a web browser environment, AJAX sent HTTP POST requests to the Apache HTTP server to fetch the converted coordinates in BD-09 format which were subsequently sent to the Baidu Maps Web API server *asynchronously* via HTTP GET requests. Once the server responded with the name of the road segment, AJAX updated the database by sending another HTTP POST request to the Apache HTTP

[4]Asynchronous JavaScript and XML

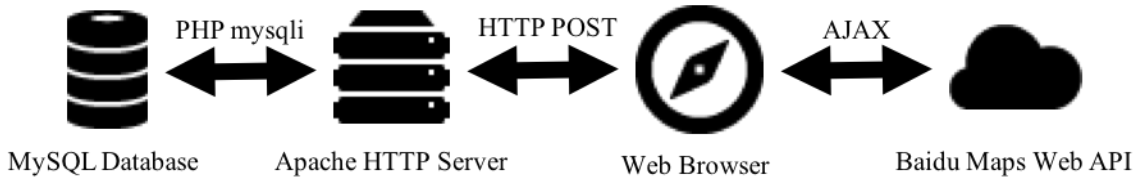

Figure 2-3: Augmented system architecture

server. The asynchronous nature of this architecture, however, caused a few problems which are addressed in Section 2.3.

2.3 Outlier Detection

2.3.1 Motivation for Outlier Detection

The Baidu Maps Web API for reverse geocoding is stable and fast, but does not produce no errors. Sometimes, a GPS data point may be mapped to a main road but actually it is on the side road, which is one of the limitations mentioned in Section 1.2 or it is actually mapped to a street that Baidu Maps does not recognise. In neither case will Baidu Maps produce a correct reverse geocoding. Moreover, the reverse geocoding process is asynchronous, which means that it is being performed in the background in parallel with the main application thread. Therefore, it is inevitable that some street names may get lost when the records are being updated or a record is updated with a wrong street name. Figure 2-4 shows a drastic example.

In this example, the Baidu Maps believes that all data points plotted belong to a particular street. But when plotted on a 2-D plane, these data points almost represent the *entire* road network in Beijing. The actual, correct street is represented in the figure as the *thickest* line on the right half of the figure with a longitude ranging from 116.45° to 116.65°. Erroneous records like those not on the thickest line are known as

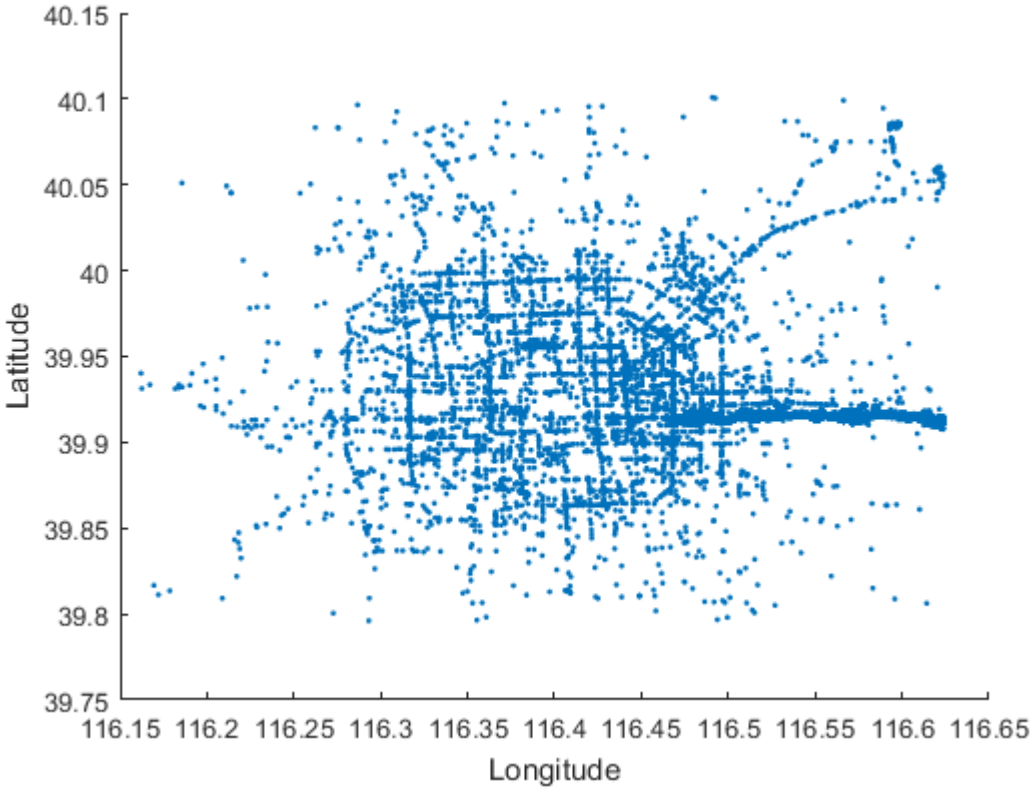

Figure 2-4: Example of outliers

outliers and must be properly identified and removed. This project proposes a novel outlier detection approach based on unsupervised learning whose principle behind is based on Theorem 1.

Definition 3 (*Reasonable reverse geocoder*)**.** A reasonable reverse geocoder always gives its best matching from a GPS data point to a street whenever possible and has an accuracy more than 50%.

Theorem 1 (*Majority Clustering Theorem*)**.** If a *reasonable reverse geocoder* is used to reverse geocode a set of GPS data points which are mapped to a particular street *in reality*, then, when plotted on a 2-D plane, majority (more than 50%) of the points

must be clustered together to form a rough shape that is similar to the shape of the street that they are supposed to be mapped to.

Proof. Proof by contradiction. Assume, for the purpose of contradiction, majority (more than 50%) of the data points that are *indeed* located on the same street are scattered arbitrarily on a 2-D plane after being reverse-geocoded by a reasonable reverse geocoder. In particular, when plotted on a 2-D plane, majority of them do not form a similar shape to that of the street they are supposed to be mapped to. Then, the majority must have been erroneously mapped to some other streets because no single street covers the whole city area. Thus, the reasonable reverse geocoder has only achieved an accuracy less 50%, which contradicts the Definition 3 of a reasonable reverse geocoder. □

2.3.2 Outlier Identification

Apparently, Baidu Maps provides a reasonable reverse geocoder because it is of industrial-grade quality and has an accuracy larger than 50%. Therefore, if a set of points belong to a particular street, after reverse-geocoded by Baidu Maps, majority of them should be clustered to assume a rough shape of that street according to Theorem 1. Based on that, an unsupervised learning technique — clustering can be used to separate the correctly mapped data points from outliers.

Many clustering techniques are available[5]. Since each record can be represented graphically by a point on a 2-D plane with longitute as the x axis and latitude as the y axis, a **self-organising feature map**[3](SOFM) seems to be an appropriate technique to use.

A self-organising feature map is a form of artificial neural network. It consists of a pre-defined number of interconnected neurons distributed over a 2-D plane as shown in Figure 2-5. Prior to training, the neurons are randomly scattered among the data

Figure 2-5: An example of SOFM

points and gradually move to the centroids of the data clusters they represent as they learn the *features* of the training data. Upon termination of the training, all data points near to a particular neuron, in terms of Euclidean distance[5], are assigned to the cluster that neuron represents. Figure 2-6 shows the results after the clustering is completed.

It is clear from the figure that while some neurons represent the clusters of outliers, majority of the neurons are clustered to *cover* the correct street they should represent. A 10×10 SOFM was used in this project, so there were at most 100 neurons or

[5]Other distance measures are also possible.

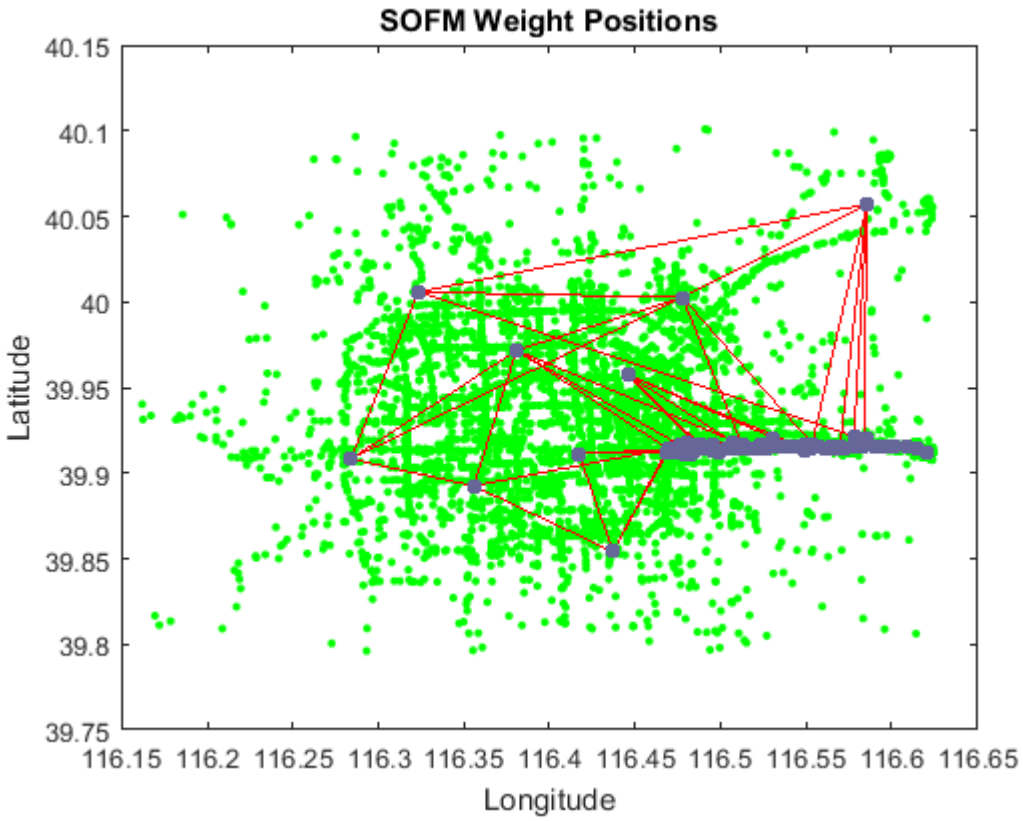

Figure 2-6: Neuron positions after training

equivalently, 100 clusters. Each cluster had a different size. To ensure a thorough removal of the outliers, **only the top 50% largest clusters were considered as clusters of correct data points which are called "legal clusters". All other clusters were deemed as clusters of outliers**.

2.3.3 Outlier Removal

Once the legal clusters were identified, a distance threshold was set to remove outliers so that **whenever the minimum distance between a data point and all centroids of the legal clusters was above the threshold, that data point**

would be considered as an outlier and removed. The python-like pseudocode in Listing 2.1 describes this idea in details.

For this project, two thresholds were selected: 30 metres and 50 metres. The thresholds were set in a way that it ensured there was sufficient data for subsequent machine learning tasks while the estimates were as least affected as possible by outliers. If the threshold were set to a too small value, the remaining data could not have been sufficient; on the other hand, however, if the threshold were set to a too big value, the accuracy of the final results would have been subject to outliers.

Listing 2.1: Pseudocode for outlier detection

```
1 for record in records:
2         min_distance = math.inf // infinity
3         for centroid in centroids:
4                 min_distance = min(min_distance, \
5                         get_distance(record, centroid))
6         if min_distance > threshold:
7                 remove(records, record)
```

Appendix A

Tables

Table A.1: Armadillos

Armadillos	are
our	friends

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Figures

Figure B-1: Armadillo slaying lawyer.

Figure B-2: Armadillo eradicating national debt.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, third edition edition, 2009.

[2] E. W. Dijkstra. A note on two problems in connections with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[3] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, pages 43, 59–69, 1982.

[4] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate gps trajectories. In *ACM SIGSPATIAL GIS 2009*. ACM SIGSPATIAL GIS 2009, November 2009.

[5] Lior Rokach and Oded Maimon, editors. *Data Mining and Knowledge Discovery Handbook*, chapter 15, pages 321–352. Springer US, 2005.

[6] Frank van Diggelen and Per Enge. The world's first gps mooc and worldwide laboratory using smartphones. In *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, pages 361–369, Tampa, Florida, September 2015.

[7] Wikipedia. Restrictions on geographic data in china, March 2017.

[8] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: Driving directions based on taxi trajectories. ACM SIGSPATIAL GIS 2010, November 2010.

[9] Bing Zhu, Peter Huang, Leo Guibas, and Lin Zhang. Urban population migration pattern mining based on taxi trajectories. In *Mobile Sensing Workshop at CPSWeek 2013*, Philadelphia, April 2013.