TRAVEL TIME ESTIMATION USING SPATIO-TEMPORAL INDEX BASED ON CASSANDRA

Zheng Wu[1], Chengming Li [1,*], Yinghao Wu [2], Fei Xiao [1], Lining Zhu [1], Jianming Shen [1]

[1] Chinese Academy of Surveying and Mapping, Beijing, China - wuzheng_gucas@163.com, cmli@casm.ac.cn, casmxiaofei@126.com, zhu_li_ning@163.com, jianmingsh@126.com
[2] School of Environment and Planning, Liaocheng University, Liaocheng, China - WuYingHaoHH@163.com

Commission IV, WG IV/10

KEY WORDS: Travel Time Estimation, Spatio-Temporal Index, GPS Trajectories, Cassandra

ABSTRACT:

Travel time estimation plays an important role in traffic monitoring and route planning. Taxicabs equipped with Global Positioning System (GPS) devices have been frequently used to monitor the traffic state, and GPS trajectories of taxicabs also used to estimate path travel time in an urban area. However, in most cases, it is difficult to find a trajectory that fits perfectly with the query path, as some road segments may be traveled by no taxicab in present time slot. This makes it hard to estimate the travel time of the query path. This paper proposes a framework to estimate the travel time of a path by using the GPS trajectories of taxicabs as well as map data sources. In this framework, the travel time is represented as a series of residence time in cells (one cell is the gird segmentation unit), thus the key issues of the estimation are: finding the local traffic patterns of frequently shared paths from historical data and computing the stay time in cells. There are three major processes in this framework: trajectories preprocessing, establishing the temporal-spatial index and cell-based travel time estimation. Based on the temporal-spatial index, an algorithm is developed that uses similar route patterns, the cell-based travel time over a period of history and road network information to estimate the travel time of a path. This paper uses GPS trajectories of 10,357 taxicabs over a period of one week to evaluate the framework. The results demonstrate that this paper's method is effective and feasible in city-wide scenarios.

1. INTRODUCTION

With the urban population increasing year by year and the expanding urban area, the serious traffic congestion has brought great pressure to the existing transportation facilities. The demand and supply of road traffic is seriously unbalanced, which increases the travel cost, especially the travel time. As an important indicator of road traffic state, the travel time can be used for traffic monitoring (Chawla et al., 2013), route planning (Yuan et al., 2010), taxi dispatching (Yuan, 2013) and ride-sharing (Wolfson et al., 2013). The travel time has been widely concerned by researchers. How to accurately estimate the travel time of a route at the current or future times is key technology of modern navigation and location-based applications (Zheng, 2015a).

Vehicle trajectories not only express vehicle running status, but also directly reflect the capacity of road network service after map matching. Using vehicle trajectories as the entry point of researching on traffic problem can not only make up for the shortage of other traffic data collection methods, get traffic data without interruption, but also can more fully express the space and time state of traffic in the whole road network (Zheng, 2015b). Therefore, using vehicle trajectories to study the travel time estimation problems has been widely accepted by researchers.

At present, there are lots of research results in the fields of travel time estimation by using trajectories. Different models and methods have been presented to estimate the travel time, which can be divided into two main categories: one is statistical method based on mass historical data, such as support vector regression (SVR) model for travel-time prediction using real highway traffic data (Wu et al., 2004), a model described probability distributions of travel times (Hofleitner et al., 2011), gradient-boosted regression tree model (Zhang et al., 2016); the other is using low-frequency floating car data and other auxiliary information (for example, points of interest (POI), road network information, weather and so on) to predict the travel time in real time, such as a dynamic travel time prediction models with real-time data collected by probe vehicles on path and its consisting link (Chen et al., 2001), a non-parametric method for route travel time estimation using low-frequency floating car data (FCD) (Rahmani et al., 2013), a model for estimating hourly average of urban link travel times using taxicab origin-destination (OD) trip data (Zhan et al., 2013), three dimension tensor model which includes geospatial, temporal and historical contexts (Wang et al., 2014).

Most of the research works are based on such a precondition: the trajectories on a subset of the roads are observed by several vehicles within a short time window. There is still no good way to estimate the travel time by using sparse trajectories. Massive vehicle trajectories are huge, grow and update dynamically. The traditional way of spatial data management cannot meet the actual application needs. How to use a spatio-temporal index to organize and query these dynamically growing trajectories is also a problem which has not been mentioned or solved in their work. Furthermore, it is only valuable for travel time estimation when trajectories have been matched on the road network, which is also difficult in doing map-matching with sparse trajectories. Based on above considerations, in this paper, we propose a framework for travel time estimation based on floating car data. This research work has three challenges: first one is trajectory sparsity, only a few subset of trajectories will be available in a specific period; secondly, outlier detection is also a problem, because the large variance in travel time

* Corresponding author

observations of the same path is obvious, which has a great impact on map-matching; finally, the spatio-temporal index should be well designed for querying in different situations. The dynamic accumulation and growth of trajectories have a significant impact on the performance of spatial and temporal queries and travel estimations. Therefore, our solution will need to handle data sparsity, various amount of uncertainty in travel time observations and the spatio-temporal index. To address these challenges, we use a cell-based approach, which decomposes a path into a sequence of cells on the road network by using the spatio-temporal index, and predicts the travel time in cells by using similar route patterns in current time slot and history. Our approach is different form link-based (Wang et al., 2014) and path-based (Zhan et al., 2013) methods, and the road network and trajectories are divided into fragments (called cells) by using Google S2 index (Eric et al., 2017), both querying and estimation are totally based on cells. The Google S2 index starts by projecting the points/regions of the sphere into a cube, and each face of the cube has a quad-tree where the sphere point is projected into. After that, some transformation occurs and the space is discretized, each face of the cube has been divided into grids, which called the cells. The cells are an hierarchical decomposition of the sphere into compact representations of regions or points. And then, the cells are enumerated on a Hilbert Curve. The Hilbert curve is a space-filling curve that converts multiple dimensions into one dimension that has an special spatial feature: it preserves the locality. So we can use the Google S2 index to encode the spatial info of trajectories as hexadecimal string, which is very convenient to be used as index in database. So the keys to our travel time estimation method are how to preprocess sparse trajectories, build spatio-temporal index and estimate the travel time based on cells.

In this paper, we use Cassandra database as our trajectories data storage. Apache Cassandra is an open-source column, family-oriented database. Its architecture is peer-to-peer, so each node in a cluster is assigned the same role, making it a decentralized database. In Cassandra, data partitioning schema plays an important role in data distribution across nodes (Vivek, 2015). Each row in Cassandra may contain one or more columns. A column is the smallest unit of data containing a name, value, and time stamp. Each row has a unique identifier key, which could be one column value or multiple column values. The keys may contains the partitioning keys and clustering keys. The partitioning key is used to determine which node the data is stored, and the clustering key is used to determine the order of data in the node. We design the row key schema, which contains the spatio-temporal index information, to query the trajectories.

The rest of the paper is organized as follows: Section 2 introduces the model and framework we used for travel time estimation. Section 3 presents the preprocessing of road networks and trajectories. Section 4 elaborates the method of constructing spatio-temporal index. The algorithm of travel time estimation is given in section 5. Section 6 presents the experiments and we conclude the paper in Section 7.

2. MODEL AND FRAMEWORK

2.1 Data Model

This paper proposes a model to estimate the travel time for a path and a framework based on the model.

Definition 1: Cell. A road network is divided into several fragments Ψ according to Google S2 index. One cell C is one grid unit on a certain level. Each Cell can be subdivided into four cells just like what quardtree does.

Definition 2: Cell Route Pattern. A cell route pattern R_c is a set of road segments linked in the cell C. Each pattern is one sub-path of road network in the cell. Different road segments linked according to the rule of road network constitute the route patterns in the cell.

Definition 3: Cell Stay Time. The cell stay time T_c is defined that the time cost by passing through or staying in the cell C. As Figure 1 illustrated, from the begin point B to the end point E, there are several paths in different color. The whole road network is divided into cells. In the cell C, it contains four cell route patterns: R_1 ($S1 \rightarrow S2$), R_2 ($S9 \rightarrow S10$), R_3 ($S3 \rightarrow S4 \rightarrow S5$) and R_4 ($S6 \rightarrow S7 \rightarrow S8$). Each pattern R_i includes several road segments S_i which are linked as subpaths of road network. For example, R_1 is consisted of S_1 and S_2, R_3 is consisted of S_3, S_4 and S_5.The time costed on R_1 can be considered as the time of passing through the cell, is denoted by $T_{c,r1}$, while the time costed on R_2 can be denoted by $T_{c,r2}$. $T_{c,r2}$ is cell stay time in C, but the route pattern R_2 does not cross C. So the travel time $T^t_{B,E}$ form B to E at time slot t can be estimated by a serial of T_c in cells on the path.

$$T^t_{B,E} = \sum_{C \in \Psi} T^t_{C,R_C} \quad (1)$$

Where Ψ is the set of cells on the route from B to E, T^t_{C,R_C} denotes the time pass through or stay in the cell C at t time slot. In this paper, we analyse the travel time of history data based on the cells, and estimate the travel time of a path based on the route pattern in each cell.

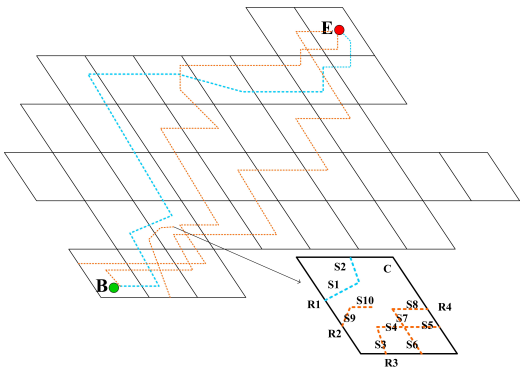

Figure 1. The model demonstration

2.2 The Framework

Generally, the trajectories collected by GPS cannot be directly used for analysis. This paper proposes a framework to purify the data, organize and index the data based the above model, and analyze data, estimate the result finally. Figure 2 presents the framework which is comprised of three main parts: the preprocessing of road network data and trajectories, the construction of spatio-temporal index and the estimation model of the travel time.

In this framework, the road network data and trajectories have been processed separately. The preprocessing procedure of road network data includes decomposition and reconstruction by using the spatio-temporal index; the trajectories have been processed by three steps: detection, filtering and map-matching. After these procedures above, the data have been imported into Cassandra database, and indexed by the spatio-temporal index strategy proposed in this paper. It is convenient to analyze the trajectories and to gather statistics by using the spatio-temporal

index. In this paper we use the historical data, current data and the road network information to predict the travel time of a path.

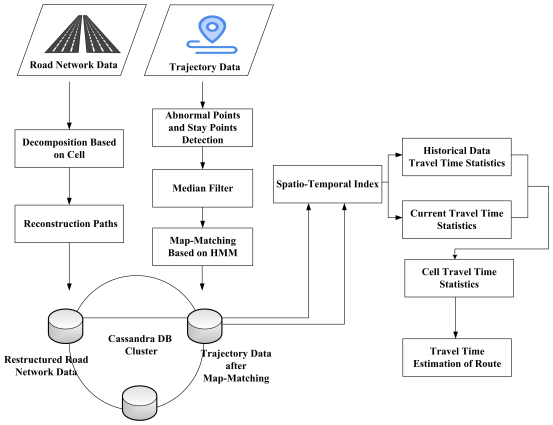

Figure 2. The framework of travel time estimation

3. DATA PREPROCESSING

3.1 Preprocessing of Road Network Data

The purpose of the preprocessing of road network data is to obtain the route patterns in a cell. So firstly, we need to decompose the road map data according to cells. Each road segment in a cell keeps its attribute information, such as the classification, the direction and its adjacent sections, which are important to reconstruct the topology information in the cell. Then, we need to find the intersection points of the road segments and the edges of the cell. The intersection points may be the start point or end point of route patterns, which can be used to search the paths among the road segments. Finally, from each intersection point, we do a depth-first search to find all possible paths: the paths cross the cell which the end point is another intersection point or the paths terminate in the cell.

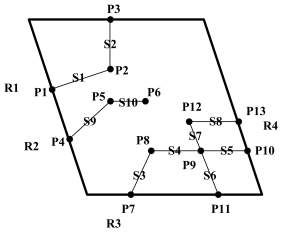

Figure 3. The cell route pattern demonstration

In Figure 3, the points P_1, P_3, P_4, P_7, P_{10}, P_{11} and P_{13} are the intersection points between the routes and the edges of cell, and there are 4 route patterns:
R_1:{S_1:$P_1 \rightarrow P_2$;S_2:$P_2 \rightarrow P_3$},
R_2:{S_9:$P_4 \rightarrow P_5$;S_{10}:$P_5 \rightarrow P_6$},
R_3:{S_3:$P_7 \rightarrow P_8$;S_4:$P_8 \rightarrow P_9$,S_5:$P_9 \rightarrow P_{10}$},
R_4:{S_6:$P_{11} \rightarrow P_9$;S_7:$P_9 \rightarrow P_{12}$,S_8:$P_{12} \rightarrow P_{13}$}.
The reconstructed road network data have been stored in database, which is denoted as $DB_{roadnetwork}$.

3.2 Preprocessing of Trajectories

Before using trajectories, we need to deal with a number of issues, such as abnormal points and stay points detection, noise filtering and map-matching. This stage is a fundamental procedure of trajectories analysis tasks. In this framework, this stage consists of three steps:
Step 1: Abnormal Points and Stay Points Detection. The abnormal points are the points at a strange position relative to the near points in a time slot. These points make the direct impact on results of filtering. So we need to check out these exceptions and remove them by using a certain distance threshold and a time interval threshold. Then, we need to find out the stay points in the trajectories. In our model, each vehicle has two states: driving or stopping. If the duration of the stop state exceeds a time threshold(30 minutes used in this paper), the vehicle needs to be marked as starting a new route. In this way, the original trajectory will be divided into multiple trajectories. In the stay points detection algorithm, we identifies the location where a vehicle has stayed for a while within a certain distance threshold. These stay points may stand for a traffic jam or parking. We further classify stay points to distinguish traffic congestion, which means the vehicle in the original route, not a new route. The purpose of doing this is to refine the time consuming of each road section and exclude the parking time.
Step 2: Filtering. We use median filter to remove from a trajectory some noise points that may be caused by the poor signal of location positioning systems. The reason we chose median filter is that it can keep the shape and width of the trajectory unchanged while filtering out the noise. Suppose that the data consists of a set x_k ($k \in [1,n]$,), the window size is m, the sequence seq: $\{x_{i-v},x_{i-v+1},...,x_i, x_{i+1},...,x_{i+v}\}$, i is the central position of the window, $v = (m-1)/2$, sort seq in ascending order, then the middle of seq is selected as the output value.

$$p_i = Med\{x_{i-v}, x_{i-v+1}, \quad , x_i, x_{i+1}, \quad , x_{i+v}\} \qquad (2)$$

where $i \in [1,n]$, $v = (m-1)/2$. In this paper, we use $m = 5$.
Step 3: Map-Matching. This step aims to project a sequence of position measurements onto a corresponding road segment where the point was truly generated. Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved states. It assumes that the state is not directly visible, but the output, dependent on the state, is visible. Each state has a probability distribution over the possible output. This is the same in map matching where a sequence of position measurements, contains implicit information about an object's movement on the map. Therefore, the sequence of output generated by HMM gives some information about the sequence of states. So we can use the sequence of position measurements and the road network data to estimate its position on the road. Two HMM map matching methods were proposed, offline map-matching (Newson et al., 2009) and online map-matching (Goh et al., 2012). Assume that a sequence of position measurements m_t at time slot t, $0 \leqslant t \leqslant T$. The matching candidates (the estimated positions on the road segment) at time slot t are denoted by $S_t = \{s_t^1, \quad , s_t^n\}$, each pair of S_t and S_{t+1} has a transition probability $p(s_t \mid s_{t-1})$. So the measurement probabilities can be formulated as follow:

$$P(m_t \mid S_t = s_t^i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left\|m_t - s_t^i\right\|^2_{great-circle}}{2\sigma^2}} \qquad (3)$$

where $\left\|m_t - s_t^i\right\|_{great-circle}$ denotes the great circle distance on the surface of the earth between the true location of a vehicle and the trajectory point. The parameter σ is the standard deviation of the position measurements. For our test data, this default value is 5 meters, which is a reasonable value for GPS noise. The

transition probabilities can be set to be proportional to the distance between S_t on the road network, as follow:

$$P(S_{t+1} = s_{t+1}^j \mid S_t = s_t^i) = \frac{1}{\beta} e^{\frac{-d_t}{\beta}} \quad (4)$$

$$d_t = \left| \left\| m_t - m_{t+1} \right\|_{great-circle} - \left\| s_t^i - s_{t+1}^j \right\|_{route} \right| \quad (5)$$

where the *route* distance can be computed by the diver distance between s_t^i and s_{t+1}^j on the road network, β is a rate parameter of the negative exponential distribution.
After steps above, the matched trajectories have been imported into database $DB_{trajectory}$.

4. SPATIO-TEMPORAL INDEX STRATEGY

To query the trajectories efficiently, we designed the row key based on Cassandra database (Cassandra Development Team, 2017). Cassandra is NoSQL Database over P2P framework. It has high scalability and availability without compromising performance, which is very suitable for the storage of streaming data, such as GPS trajectories.

4.1 Spatio-Temporal Index

Google S2 index is essentially a hierarchical space filling curve strategy. It combines the quadtree with Hilbert Curve. In this paper, we use S2 index to encode the spatial information of trajectory after map-matching. In our framework, the 12th level(average cell area is about 5.067 km^2) is used to divide the map into cells for trajectories indexing, and the 14th level(average cell area is about 0.3166 km^2) is used to divide road network data into cells for estimating the travel time, as shown in Figure 4 and Figure 5. Each cell has a unique ID, which we used as the identification of spatial information of trajectory in a region. For example, one point of the map-matching trajectory is (Lng:116.500393, Lat:39.906996), the cell id at 12th level is 35f1ac3, the time stamp is 2008-02-06 18:10:58, we convert it to a long value which is the milliseconds of the time from January 1, 1970 to that time (only contains the year, month, day and hour fields), the minute and second fields is also converted to milliseconds, its id is 6275, so its spatio-temporal key can be presented as 1202292000000-35f1ac1-658000-6275, which can be directly used as row key in NoSQL database. Similarly, for reconstructed road network data, we use the cell id, the year and the month as the spatio-temporal key.

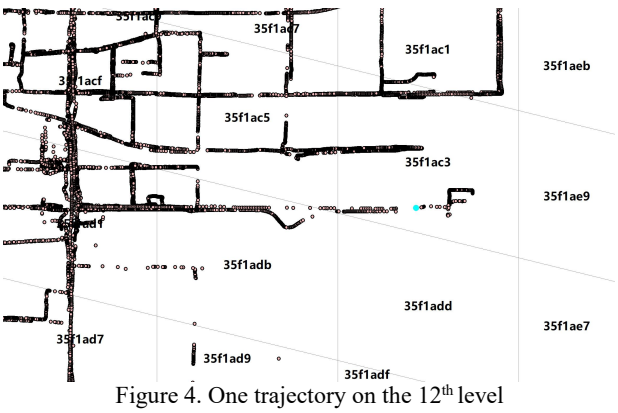

Figure 4. One trajectory on the 12th level

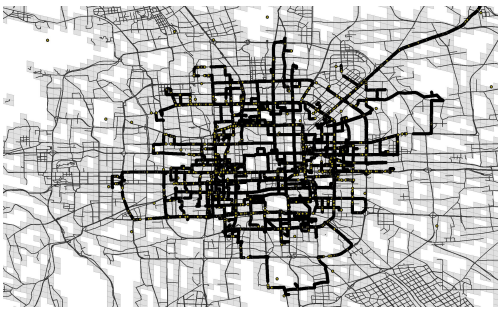
Figure 5. Major road network and one trajectory on the 14th level

In our design, the row key of matched trajectories is composed of two parts: partition key and clustering key, the priority of partition key is higher than clustering key's. The time stamp (in the format of yyyy-MM-dd HH:mm:ss) is split into the year-month-day-hour field (yyyy-MM-dd HH) and the minute-second (mm:ss) field. The partition key includes the date time which is accurate to the hour, and the cell ID of S2 index at the 12th level. The clustering key includes the 14th level cell ID, the minute, second and object ID. Figure 6 shows the format of the row key which we used in Cassandra Database. Correspondingly, the row key of reconstructed road network data in one cell is shown in Figure 7.

Row Key				Column			
Partition Key		Clustering Key					
yyyy-MM-dd HH	12th level Cell ID	mm:ss	Object ID	Route ID	X	Y	Status

Figure 6. The row key of map-matched trajectories

Row Key				Column			
Partition Key		Clustering Key					
yyyy-MM	12th Level Cell ID	14th Level Cell ID	Route Pattern ID	Road Name Set	Road Type Set	Single or two-way Set	Geometry(Multi-LineString)

Figure 7. The row key of reconstructed road network data

According to Distributed Hash Table (DHT), the row key is mapping to Chord (Hash Ring Model) by its hash value, then determine the location of the storage node in a clockwise direction. As shown in Figure 8, the object *i* gets its hash value *Key*(*i*) = *hash*(*rowkey*), then find the position on the Chord, it is between the Node *N* and Node *1* , so the object should be stored on the Node *1*.

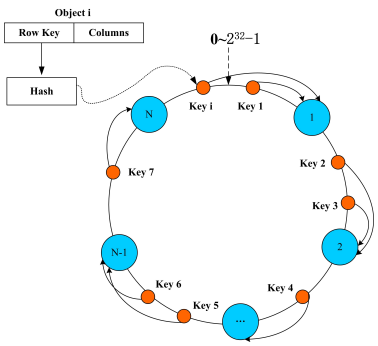

Figure 8. The Hash Mapping

4.2 Spatio-Temporal Query

In the model presented above, it is convenient to retrieve the trajectory of one vehicle at a certain time or period, such as: range queries and K-Nearest Neighbor (KNN) queries (Zheng, 2015c).
Range queries: Retrieve the trajectories falling into a spatio-temporal range, as shown in Figure 9. For example, if we want

to retrieve the trajectories of vehicles passing a given rectangular region R between 10 a.m. - 12 p.m. in one week. The retrieved trajectories (or route patterns) can then be used to derive features, such as the travel speed and traffic flow for estimate the travel time. First, we can split the time interval into hours, in this example are 10 a.m., 11a.m. and 12 p.m., and compute the set of cell IDs which covered the region R, then according to the time value set (long type values) and the cell ID set, generate the partition keys to query the database. These queries can be executed in parallel in different node on the Chord.

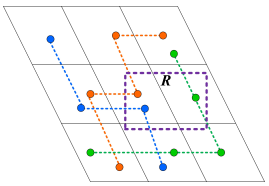

Figure 9. Range queries

KNN queries: Retrieve the top K trajectory points with the minimum aggregate distance to a certain point or a specific trajectory at certain time. For the first case, we just to find the cell C which contains the point at the certain time and the neighbor cells which are adjoining cells of C, we only need the time and the cell IDs of the cells to compute the partition key; then we get all the trace points, the rest of work is just do a regular KNN query to find the nearest points, as shown in Figure 10(a). For the second case, we can get the cells which cover the specific trajectory at that time, and then find K trajectories which have the minimum aggregate distance to this trajectory, as shown in Figure 10(b). If we cannot find the trajectories, we include neighbor cells to extend the query range. The minimum aggregate distance depends on the definition a similarity or distance function between two trajectories. In this model, the aggregate distance is calculated as follow:

$$D_{R_j,R_i} = \sum_{r \in R_j} d_{\min}(r, R_i) \tag{6}$$

$$d_{\min}(r, R_i) = \min\{d(r, S^i_{m,n})\} \tag{7}$$

$$d(r, S^i_{m,n}) = \begin{cases} \|r,m\|, \text{ if } \theta_{\overrightarrow{mn},\overrightarrow{mr}} \geq 90° \\ \|r,n\|, \text{ if } \theta_{\overrightarrow{nm},\overrightarrow{nr}} \geq 90° \\ d_{\perp}(r, S^i_{m,n}), \quad \text{otherwise} \end{cases} \tag{8}$$

where r is a point on route R_j, d_{min} is the minimum distance from point r to route R_i, the distance between R_j to R_i is defined as the sum of the d_{min}. $S^i_{m,n}$ is the segment of R_i, and its endpoints are m and n. So the distance from r to $S^i_{m,n}$ depends on the included angle among $\overrightarrow{mn}$, $\overrightarrow{mr}$, $\overrightarrow{nm}$ and $\overrightarrow{nr}$. $d_{\perp}(r, S^i_{m,n})$ denotes the perpendicular distance from r to $S^i_{m,n}$.

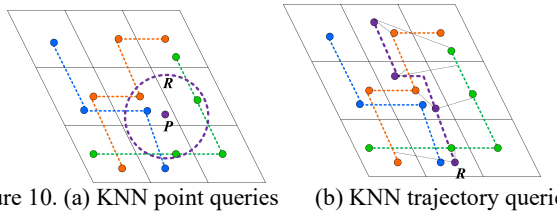

Figure 10. (a) KNN point queries (b) KNN trajectory queries

5. ESTIMATION OF TRAVEL TIME

To estimate the travel time, firstly we compute the historical travel time based on the trajectories, which have already been store in the database. Then we compute the travel time of each route pattern by splitting travel time of whole trajectory into segments according to cells. So we can obtain the travel time of route patterns in each cell if there is route pattern similar to the historical trajectory. Finally, based on the information obtained above, we can do range queries or KNN queries to estimate the travel time of any path.

5.1 Cell Route Pattern Search

In the preprocessing stage, we have already reconstructed the route patterns in one cell, which can be easily retrieved from the Cassandra database by using the row key designed above. To find a route pattern matching a certain trajectory in one cell, one trajectory is divided into fragments by cells. Each fragment is a set of segments of road network, which could be a subset of route patterns. So we need to find out the route pattern matched the given fragment. We firstly use the minimum bounding box and length of the segment to make a preliminary comparison, and then we do a buffer operation to judge a route pattern exactly equal to the given segment. If we cannot find a route pattern which matches the given fragment, the KNN trajectory queries mentioned above can be used to find a similarity route pattern as a candidate. So a trajectory $P_{B,E}$ (from B to E) can be presented as a set of route pattern in cells: $P_{B,E} = \{R_C, C \in \Psi\}$, where R_C is a route pattern in cell C, Ψ is the cell set of $P_{B,E}$.

5.2 Cell Stay Time Estimation

According to the cell stay time defined above, it can be obtained by the historical statistics. In section 5.1, each trajectory can be divided into fragments in cells as sets of route patterns, so the travel time of matched route pattern can be calculated according to the travel time of the fragments in history. Ideally, after a large number of trajectories statistics, most of route patterns have an estimated travel time at a certain time slot. But there are a number of route patterns cannot estimate travel time only based on sparse trajectories generated by a sample of vehicles, as a driver can only travel a few road segments in a short time period. In our model, it follows such rules: firstly use the statistics of the corresponding time in previous time slot; if we cannot find such route pattern in current time slot, then use the statistics in last few days; if there are no statistics available for this route pattern, use the similarity route pattern referred above as an approximation or the average travel time of this cell; if there is still nothing that can be used, just use the speed limit of the road to compute the travel time or refer to the adjacent cells. In particular, for the start cell and the end cell which the trajectory may not pass through cells generally, the travel time need to be calculated proportionately according to attributes of roads, such as the length and speed limit of a road segment.

5.3 Travel Time Estimation Base on Cells

To estimate one path $P_{B,E}$ from the starting position B to the destination E at time slot T, it can be described as the sum of cell stay time on $P_{B,E}$, which has already defined in Formula (1). So to get the travel time estimation, it mainly contains two key operations: search the route patterns which $P_{B,E}$ passed, calculate the travel time of each route pattern. It usually follows steps below:

Step 1: Range query. Use the minimum bounding rectangle MBR_P of $P_{B,E}$ to find the cell set Ψ which covers $P_{B,E}$. Then use

the cells Ψ to get the trajectories S at time slot T from $DB_{trajectory}$, which is a series of range query operations. Similarly, use the cells Ψ to retrieve the route patterns R from $DB_{roadnetwork}$.

Step 2: Find the cell route patterns. Divide $P_{B,E}$ into fragments by cells. In order to find the matched route pattern $P_{B,E}$, the buffer spatial operation or the KNN trajectory queries may be used which have already been described in section 5.1.

Step 3: Estimate the cell stay time. After retrieving S and R, the stay time in cells can be calculated by rules mentioned in section 5.2. Then the travel time estimation of $P_{B,E}$ can be obtain as the sum of stay time in cells.

6. EXPERIMENTS

In this section, we evaluate the effectiveness of the framework proposed in this paper.

6.1 Experimental Data

Taxi trajectories. We test our algorithm using T-Drive data set (Yuan et al., 2010), which are trajectories of GPS-equipped taxis in Beijing provided by Microsoft Research (Yuan et al, 2011). This data set contains the GPS trajectories of 10,357 taxis during the period of Feb.2 to Feb.8, 2008 within Beijing. The total number of GPS points reaches about 15 million and the total distance is about to 9 million kilometers. We select 610 taxis as our experiment data to ensure that the selected taxis have relative stable sampled trajectories, thus the trajectories can be matched on to the road networks. During the preprocessing stage, the GPS points in the trajectories are marked with binary status labels: driving or stopping, and the route ID which denotes a new route after a long time stop state.

The travel time during working days and weekends are very different, it is largely influenced by resident trip rules. We only have trajectories in one week period, so we can only use the trajectories from Monday to Thursday as historical data and use the ones on Friday as the current data, to verify our method.

Road networks. The road networks of Beijing are extracted from the OpenStreetMap (OSM) (OpenStreetMap, 2015), which are used for map matching. The road class information(e.g. motorway, primary, secondary and tertiary road), the road priority(different road class has different priority, which can be used as one guidance for the selection in map-matching) and the max speed limit in OSM data are also extracted for map-matching and travel time estimation.

6.2 Evaluation Metrics

Mean Absolute Error (MAE) and Mean Relative Error (MRE) are used to evaluate estimation accuracy over queries (Yuan, 2010):

$$MAE = \frac{\sum_i \left| t_i - \hat{t}_i \right|}{n} \tag{9}$$

$$MRE = \frac{\sum_i \left| t_i - \hat{t}_i \right|}{\sum_i \hat{t}_i} \tag{10}$$

where t_i and $\hat{t}_i$ are the estimated travel time and the actual travel time of the *i-th* path.

6.3 Travel Time Estimation Result

We estimate the travel time with cell stay time in current time and history. The cell stay time can be used to judge the level of the road traffic congestion and the traffic flow state in local area. We use the statistics of cell stay time form Monday to Thursday to compute the traffic performance index (TPI), which comprehensively reflects the conceptual value of the smooth or congested road network. TPI means "smooth" between 0 and 2 (can run according to the speed limit of the road), "basic smooth" between 2 and 4 (takes 0.2 to 0.5 times more than smooth), between 4 and 6 as "mild congestion" (takes 0.5 to 0.8 times more than smooth), "moderate congestion" between 6 and 8 (takes 0.8 to 1.1 times more than smooth), and "serious congestion" between 8 and 10 (takes more than 1.1 times). Figure 11 and Figure 12 shows TPI within Beijing's Fifth Ring Road at 10 a.m. and 16 p.m., respectively.

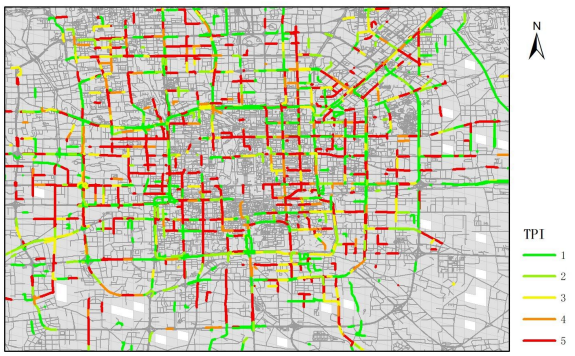

Figure 11. TPI at 10 a.m. from Monday to Thursday

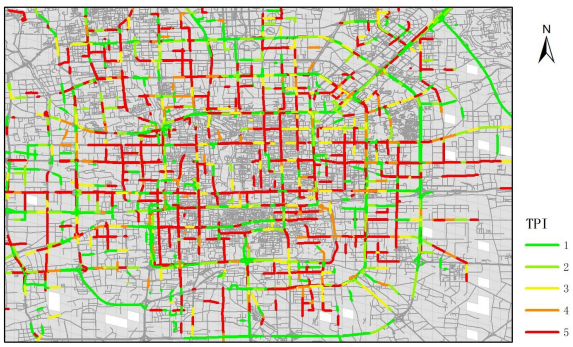

Figure 12. TPI at 16 p.m. from Monday to Thursday

To express the traffic state in the local area, we compute the average speed in fourteenth level cells, as shown in Figure 13 and Figure 14. The average speed of most areas within Beijing's Fifth Ring Road is about 40-60 km/h at 10 a.m. and 20-40 km/h at 16 p.m. from Monday to Thursday. It is basically consistent with the actual traffic situation in that week.

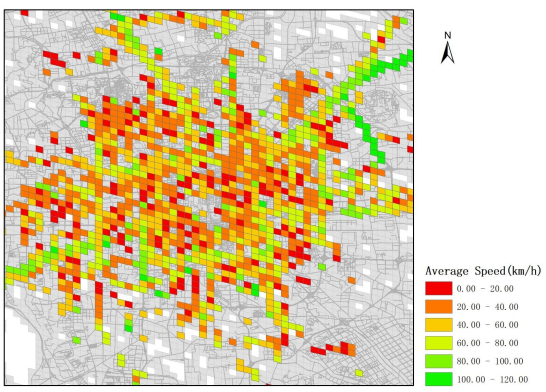

Figure 13. Average speed in 14th level at 10 a.m. from Monday to Thursday

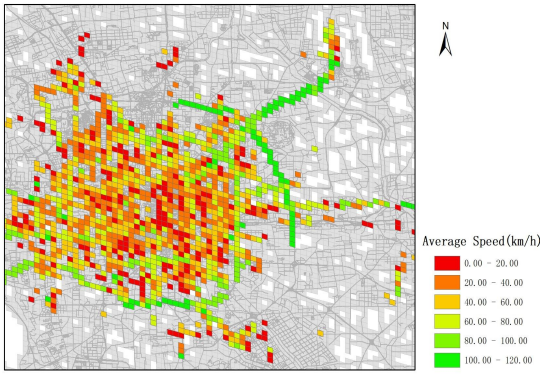

Figure 14. Average speed in 14th level at 16 p.m. from Monday to Thursday

In order to verify the accuracy of estimating the travel time of road segments based on cells, we randomly query the travel time within 610 taxis from 9 a.m. to 17 p.m. on Friday. We compare the overall travel time estimation results with their original values as a ground truth to calculate MAE and MRE. For example the travel time estimation of the taxi (ID 1277) from 9 a.m. to 14 p.m. is shown in Figure 15. The deviation is within the range 1.45 min~17.38 min. During rush hours, such as 9 a.m. and 13 p.m., there is a large deviation in rush hours. The deviation is larger than 15 minutes. In the non-traffic peak periods, the deviation of traffic time is less than 2 minutes. Similarly, the travel time estimation of the taxi (ID 8662) also has large deviation in rush hours (11 a.m. to 13 p.m. at noon and 17 p.m. in the afternoon), shown in Figure 16. The main reason for the larger deviation is that the variance of traffic jam time is larger in rush hours than at other times. The variance of traffic congestion leads to the uncertainty of travel time in the cells. And the sparsity of trajectory can also cause larger estimated deviations.

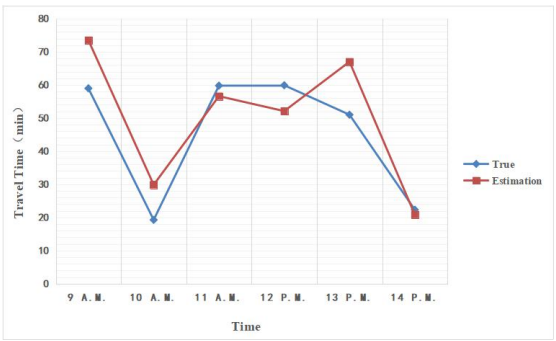

Figure 15. Travel time estimation of taxi ID 1277 on Friday

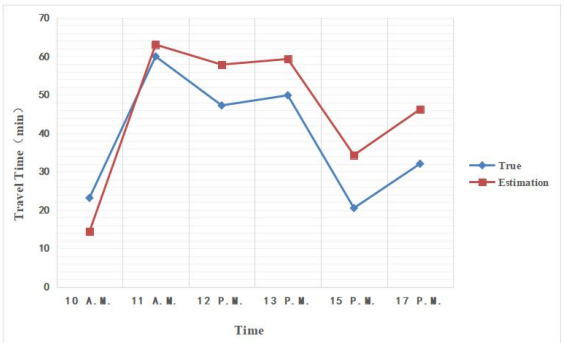

Figure 16. Travel time estimation of taxi ID 8662 on Friday

We chose 5, 10, 15, 20 and 25 paths at different time slots on Friday respectively, and analyze the deviation between the estimated time and the real value. As shown in Figure 17 and Figure 18, the MAE is in the range of 8.54min~10.43min and MRE is around 0.2. When the number of path increases, the MAE drops from 10.43 minutes to 8.54 minutes, the change MAE is not obvious, which means that our method has good overall accuracy, especially for large scale estimations, and the MRE floats up and down around 20%, while the estimation accuracy is relatively stable, although having more taxis increases the variability in the historical travel time of a path.

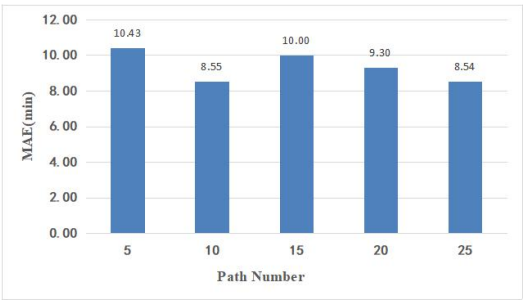

Figure 17. MAE with a varying quantity of paths

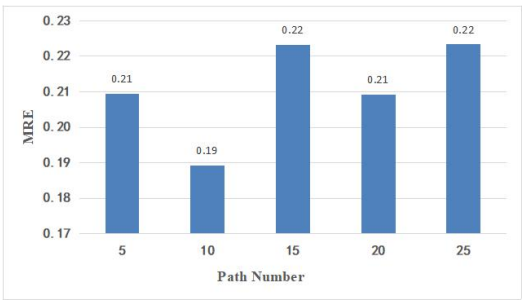

Figure 18. MRE with a varying quantity of paths

We also test the estimation accuracy of our model at different time slots on Friday. Figure 19 shows a larger volatility on travel time estimation in rush hours (9 a.m. in the morning, 13 p.m. at noon and 17 p.m. in the afternoon) and at other time slots(15 p.m, which is the time the students from school). Because in rush hours, greater uncertainty is introduced by the traffic congestion, which further leads to a large variance in cell stay time statistics, and finally produces a large deviation in the estimation of the travel time. For example at 13 p.m., the MAE has reached 15.41 minutes and the MRE is about 0.47 at 17 p.m.. But on the average, the MAE is only 8.50 minutes and the MRE is 0.23, which also means that the estimation model proposed in this paper is effective.

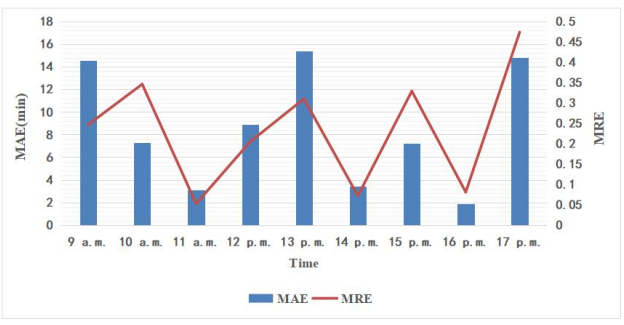

Figure 19. MAE and MRE at different time on Friday

7. CONCLUSION

The real-time GPS trajectories of taxis are very import for intelligent traffic management in modern cities. A practical solution for trajectory-based travel time estimation needs to be suitable for sparse trajectories. In this paper, we propose a framework to estimate the travel time of a path in current time slot based on cells by using spatio-temporal index. We define a cell-based estimation model to describe the travel time in cells, and give out the implementation of the framework. In the preprocessing stage, the outliers and stay points detection, filtering and map-matching have been used to process the trajectories and generate two data sources based on Cassandra: reconstructed road network data (cell-based route patterns) and map-matched trajectories. Based on Google S2 index, we proposed a spatio-temporal index strategy for range queries and KNN queries, which are used in the travel time estimation stage. The estimation algorithm utilize the cell stay time in current slot and history observations, the route patterns in cells and road network properties to predict the travel time. We experiment on T-Drive data set and demonstrate the good accuracy and the robustness for both historical trajectories and real-time trajectories. As shown in the experiments, the framework presented in this paper is a practical solution for travel time estimation based on sparse trajectories.
In the future, we plan to analyze the spatial and temporal distribution characteristics of trajectories based on the index, and study the impact of other factors, such as points of interest (POI), divisions of urban functional regions and weather conditions with the estimated travel time.

ACKNOWLEDGEMENTS

This study was funded by the Basic Research Funding of Chinese Academy of Surveying and Mapping (7771804).

REFERENCES

Cassandra Development Team, 2017. Apache Cassandra database, Version 3.10. The Apache Software Foundation https://cassandra.apache.org/ (3 February 2017).

Chawla, S., Zheng, Y., and Hu, J., 2013. Inferring the Root Cause in Road Traffic Anomalies. *IEEE, International Conference on Data Mining*, pp.141-150.

Chen, M., and Chien, S., 2001. Dynamic freeway travel-time prediction with probe vehicle data: link based versus path based. *Transportation Research Record Journal of the Transportation Research Board*, 1768(1), 157-161.

Eric Veach, Jesse Rosenstock, Eric Engle, Robert Snedegar, and Julien Basch, 2017. S2 Geometry Library. S2Geometry http://s2geometry.io/.

Goh, C. Y., Dauwels, J., Mitrovic, N., Asif, M. T., Oran, A., and Jaillet, P., 2013. Online map-matching based on Hidden Markov model for real-time traffic sensing applications. *International IEEE Conference on Intelligent Transportation Systems*, Vol. 24, pp. 776-781.

Hofleitner, A., and Bayen, A., 2011. Optimal decomposition of travel times measured by probe vehicles using a statistical traffic flow model. *International IEEE Conference on Intelligent Transportation Systems*, Vol. 263, pp. 815-821.

J. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., and Sun, G., et al., 2010. T-drive: driving directions based on taxi trajectories. *Sigspatial International Conference on Advances in Geographic Information Systems*, pp.99-108.

Newson, P., and Krumm, J., 2009. Hidden Markov map matching through noise and sparseness. *ACM Sigspatial International Conference on Advances in Geographic Information Systems*, pp. 336-343.

OpenStreetMap Data, 2018. OpenStreetMap Project. The OpenStreetMap Foundation http://www.openstreetmap.org. (1 March 2018).

Rahmani, M., Jenelius, E., and Koutsopoulos, H. N., 2013. Route travel time estimation using low-frequency floating car data. *International IEEE Conference on Intelligent Transportation Systems*, pp. 2292-2297.

Vivek, Mishra, 2015. Beginning Apache Cassandra Development. Springer, Berlin, pp. 53-54.

Wang, Y., Zheng, Y., and Xue, Y., 2014. Travel time estimation of a path using sparse trajectories. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.25-34.

Wolfson, O., Zheng, Y., and Ma, S., 2013. T-share: A large-scale dynamic taxi ridesharing service. *IEEE, International Conference on Data Engineering*, pp.410-421.

Wu, C. H., Ho, J. M., and Lee, D. T., 2004. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4), pp. 276-281.

Yuan, N. J., Zheng, Y., Xie, X., and Sun, G., 2011. Driving with knowledge from the physical world. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 316-324.

Yuan, N. J., Zheng, Y., Zhang, L., and Xie, X., 2013. T-finder: a recommender system for finding passengers and vacant taxis. *IEEE Transactions on Knowledge & Data Engineering*, 25(10), pp. 2390-2403.

Zhan, X., Hasan, S., Ukkusuri, S. V., and Kamga, C., 2013. Urban link travel time estimation using large-scale taxi data with partial information. *Transportation Research Part C*, 33(4), pp. 37-49.

Zhang, F., Zhu, X., Hu, T., Guo, W., Chen, C., and Liu, L., 2016. Urban link travel time prediction based on a gradient boosting method considering spatiotemporal correlations. *ISPRS International Journal of Geo-Information*, 5(11), pp. 201.

Zheng, Yu, 2015a. Introduction to urban computing. Geomatics & Information Science of Wuhan University, 40(1), pp.1-13.

Zheng, Yu , 2015b. Methodologies for cross-domain data fusion: an overview. IEEE Transactions on Big Data, 1(1), pp.16-34.

Zheng, Yu, 2015c. Trajectory Data Mining: An Overview. *ACM*, 6(3), pp. 1-41.