Finding the Shortest Paths in Road Networks with Minimum Pair

K. Venkatesh Guru[1]
[1] Associate Professor
Department of Computer Science and Engineering K.S.R College of Engineering Tiruchengode, India

K. Leelavathi[2], P. Manimegalai[3], S. Mounika[4], M. Nainarmohammed[5]
[2,3,4,5] U G Student
Department of Computer Science and Engineering K.S.R College of Engineering Tiruchengode, India

***Abstract*—Finding the shortest path in road networks becomes one of important issues in location based services (LBS). The problem of finding the optimal meeting point for a group of users has also been well studied in existing works. This paper investigates a new problem for two users. Each user has his / her own source and destination. However, whether to meet before going to their destinations is with some uncertainty. The paper models it as minimum path pair (MPP) query, which consists of two pairs of source and destination and a user-specified weight *α* to balance the two different needs. The result is a pair of paths connecting the two sources and destinations respectively, with minimal overall cost of the two paths and the shortest route between them. To solve MPP queries, it devises an algorithm by enumerating node pairs. An efficient algorithm based on point-to-point shortest path calculation is proposed to further improve query efficiency. It also presents a smart driving direction system IS proposed to model the dynamic traffic pattern so as to provide a user with the fastest route to a destination with edge failure situation.**

Keywords*— *MMP, MPP, Shortest Path Finding, Point-To-Point Shortest Path

I. INTRODUCTION

Finding efficient driving directions has become a daily activity and been implemented as a key feature in many map services like Google and Bing Maps. A fast driving route saves not only the time of a driver but also energy consumption (as most gas is wasted in traffic jams). Therefore, this service is important for both end users and governments aiming to ease traffic problems and protect environment.

Essentially, the time that a driver traverses a route depends on the following three aspects: 1) the physical feature of a route, such as distance, capacity (lanes), and the number of traffic lights as well as direction turns; 2) the time-dependent traffic flow on the route; and 3) a users driving behavior. Given the same route, cautious drivers will likely drive relatively slower than those preferring driving very fast and aggressively. Also, users' driving behaviors usually vary in their progressing driving experiences. For example, traveling on an unfamiliar route, a user has to pay attention to the road signs, hence drive relatively slowly. Thus, a good routing service should consider these three aspects (Usually, big cities have a large number of taxicabs traversing in urban areas.

For efficient taxi dispatching and monitoring, taxis are usually equipped with a GPS sensor, which enables them to report their locations to a server at regular intervals, e.g., 2-3 minutes. That is, a lot of GPS equipped taxis already exist in major cities, generating a huge number of GPS trajectories every day. Intuitively, taxi drivers are experienced drivers who can usually find out the fastest route to send passengers to a destination based on their knowledge (To believe most taxi drivers are honest although a few of them might give passengers a roundabout trip). When selecting driving directions, besides the distance of a route, they also consider other factors, such as the time variant traffic flows on road surfaces, traffic signals and direction changes contained in a route. These factors can be learned by experienced drivers but are too subtle and difficult to incorporate into existing routing engines.

Therefore, these historical taxi trajectories, which imply the intelligence of experienced drivers, provide us with a valuable resource to learn practically fast driving directions. In this paper, we propose a cloud-based cyber-physical system for computing practically fast routes for a particular user, using a large number of GPS-equipped taxis and the users GPS-enabled phone.

GPS equipped taxis are used as mobile sensors probing the traffic cuts, traffic, and drivers) which are far beyond the scope of the shortest/fastest path computing. GPS-equipped taxis are used as mobile sensors probing the traffic rhythm of a city in the physical world. Second, a cloud in the cyber world is built to aggregate and mine the information from these taxis as well as other sources from Internet, like web maps and weather forecast. The mined knowledge includes the intelligence of taxi drivers in choosing driving directions and traffic patterns on road surfaces. Third, the knowledge in the cloud is used in turn to serve Internet users and ordinary drivers in the physical world. Finally, a mobile client, typically running in a users GPS-phone, accepts a users query, communicates with the cloud, and presents the result to the user.

Finding the shortest path between two nodes in a graph has been well studied. In recent years, with the proliferation of mobile devices, it also becomes one of the important issues in location based services (LBS), since finding fast routes in transportation networks is a fundamental operation in navigation systems. In general, there are reported studies on shortest paths for a single user or for a group of users.

For single user, in addition to finding a single shortest path from the source to the destination, i.e. the classic shortest path problem, the issue of finding k shortest paths from the

source to the destination is studied. There are other queries being studied for a single user such as finding the alternative shortest path or the constrained shortest path] between two nodes by introducing extra conditions to the original shortest path problem.

For group users, Xu et al. study the proximity relations for a group users in road networks. The query q is a set of query points, which represent the locations where the users are. The queries are either to find a node, vc, in the road network for the query points to meet, or to find the pair wise-distance among the query points. Several distance functions are studied: the sum-distance between q and vc node is the sum of distances from all query points to the node, the max-distance for q and vc is the maximum distance among the distances from all query points to vc, the maximum-pairwise-distance is the maximum pair wise distance among all pairwise distances of q. Then, the queries for a group of users aim to find the min-sum-distance, min max-distance and max-pair wise-distance for q.

The min sum-distance query is also called the optimal meeting point (OMP) query. This paper studies a new query in road networks. It begins with a fundamental and intuitive case with two users or groups. Here, each of the two users has his/her own source and destination. The two users want to go to their destinations along the shortest paths. On the other hand, the two users may have to meet before going to their destinations. However, there could be uncertainty or weight associated with the meeting.

1. To construct a road network.

2. To find Minimum Path Pair for two source and destination.

3. To incorporate more time-dependent land mark graphs for the given time slot.

4. To find shortest path between more pairs of source and destination.

5. To take into account, edge failure scenario (with the dynamic node and edges failures), and provide smart route to the end user.

II. RELATED WORKS

Ittai Abraham And Daniel Delling studied hierarchical hub labeling for computing shortest paths. Their new theoretical insights on the structure of hierarchical labels lead to faster preprocessing algorithms, making the labeling approach practical for a wider class of graphs and enabling real-time traffic updates on road networks. They also found smaller labels, improving the query speed. They concluded that their study of hierarchical labeling makes HL practical on a wider class of problems. The work raised several natural questions. It would be interesting to study the relationship between hierarchical labeling, which are more practical, and general hub labeling, which have theoretical guarantees on the label size. In preliminary experiments, labels are produced that are about as good as those by TD for some graph classes (such as road networks), but asymptotically smaller for others (like small-world graphs). Another open question is how to efficiently compute (or approximate) optimal hierarchical labeling, or a good lower bound on their size.

Ruoming Jin And Ning Ruan the authors stated that the distance query, which asks the length of the shortest path from a vertex u to another vertex v, has applications ranging from link analysis, semantic web and other ontology processing, to social network operations. They proposed a novel labeling scheme, referred to as Highway-Centric Labeling, for answering distance queries in a large sparse graph. It empowers the distance labeling with a highway structure and leverages a novel bipartite set cover framework/algorithm. Highway-centric labeling provides better labeling size than the state-of-the-art 2-hop labeling, theoretically and empirically. It also offers both exact distance and approximate distance with bounded accuracy. A detailed experimental evaluation on both synthetic and real datasets demonstrates that highway centric labeling can outperform the state-of-the-art distance computation approaches in terms of both index size and query time.

Takuya Akiba and Yoichi Iwata proposed a new exact method for shortest-path distance queries on large-scale networks. Their method pre-computes distance labels for vertices by performing a breadth-first search from every vertex. Seemingly too obvious and too inefficient at first glance, the key ingredient introduced here is pruning during breadth-fist searches. While they could still answer the correct distance for any pair of vertices from the labels, it surprisingly reduces the search space and sizes of labels. Moreover, they showed that they could perform 32 or 64 breadth-first searches simultaneously exploiting bitwise operations. They experimentally demonstrated that the combination of these two techniques is efficient and robust on various kinds of large-scale real-world networks. In particular, their method could handle social networks and web graphs with hundreds of millions of edges, which are two orders of magnitude larger than the limits of previous exact methods, with comparable query time to those of previous methods. They concluded that they planned to investigate ways to handle even larger graphs, where indices and/or graphs might not fit in main memory. The first way is to reduce the index size by reducing graphs exploiting obvious parts and symmetry and compressing labels by making dictionaries of common sub trees for shortest path trees. Another way is disk-based or distributed implementation. Disk-based query answering is obvious and ready, and the challenges are particularly on preprocessing.

Christian Sommer considered the point-to-point (approximate) shortest-path query problem, which is the following generalization of the classical single-source (SSSP) and all-pairs shortest paths (APSP) problems: The author first presented with a network (graph). A so-called preprocessing algorithm may compute certain information (a data structure or index) to prepare for the next phase. After this preprocessing step, applications may ask shortest-path or distance queries, which should be answered as fast as possible. Due to its many applications in areas such as transportation, networking, and social science, this problem has been considered by researchers from various communities (sometimes under different names): algorithm engineers construct fast route planning methods, database and information systems researchers investigate materialization tradeoffs, query processing on spatial

networks, and reachability queries, and theoretical computer scientists analyze distance oracles and sparse spanners.

Andy Diwen Zhua and Hui Ma stated that given two locations s and t in a road network, a distance query returns the minimum network distance from s to t, while a shortest path query computes the actual route that achieves the minimum distance. These two types of queries find important applications in practice and a plethora of solutions have been proposed in past few decades. The existing solutions, however, are optimized for either practical or asymptotic performance, but not both. In particular, the techniques with enhanced practical efficiency are mostly heuristic based, and they offer unattractive worst-case guarantees in terms of space and time. On the other hand, the methods that are worst-case efficient often entail prohibitive preprocessing or space overheads, which render them inapplicable for the large road networks (with millions of nodes) commonly used in modern map applications.

Lijun Chang , Xuemin Lin and Lu Qin stated that Driven by many applications, they studied the problem of computing the top-k shortest paths from one set of target nodes to another set of target nodes in a graph, namely the top-k shortest path join (KPJ) between two sets of target nodes. While KPJ is an extension of the problem of computing the top-k shortest paths (KSP) between two target nodes, the existing technique by converting KPJ to KSP has several deficiencies in conducting the computation. To resolve these, they proposed to use the best-first paradigm to recursively divide search subspaces into smaller subspaces, and to compute the shortest path in each of the sub spaces in a prioritized order based on their lower bounds.

III. METHODOLOGY PREPARE YOUR PAPER BEFORE STYLING

The models a road network as an undirected weighted graph G = (V, E) where V and E represent the set of nodes and the set of edges of G, respectively. The existing system contains a new query named minimum path pair (MPP) query, denoted as q = {(v1s, v1t), (v2s,v2t)}, which is composed of two source/destination pairs, with an α to balance the needs of going to the destinations fast and the needs to meet. The MPP query addresses the issue of finding the shortest paths with meeting uncertainty i.e., both paths need not meet each other. All the existing system aspects are worked out in proposed system. In addition, the proposed system finds shortest path between multiple pairs of source and destination. It proposes a time-dependent landmark graph to provide a user with the practically fastest route to a given destination at a given departure time.

A. Minimum Path Pair (MPP) Query for Shortest Path Finding

This section contains a new query named minimum path pair (MPP) query, denoted as q = {(v1s, v1t), (v2s, v2t)}, which is composed of two source/destination pairs, with an α to balance the needs of going to the destinations fast and the needs to meet. The MPP query addresses the issue of finding the shortest paths with meeting uncertainty i.e., both paths need not meet each other.

B. Add Terminal

In this section, Terminal details are added and saved to ‘ Terminal’ table. X Position and Y Position in the graph details are added. The details are viewed using data grid view control.

C. Add Neighbor Terminal

In this section, Terminal id, Neighbor Terminal id and Distance details are added and saved to ‘ Neighbor Terminals table. This distance information will assist the route calculation.

D. Graph Construction

In this section, a graph is being constructed. To view the graph, click the Create Graph From table button. It displays a graph with nodes in round shape with arrows as edges. You can click on the node and drag it. It will move to new location. All the lines will be redrawn. The ‘ nodes’ table and ‘ links’ table records are used to create the graph. [Note: During Add Terminal ‘ Nodes’ record is created. During Add Neighbor Terminal ‘ Links’ record is created.]. It denotes the terminals and road segment details for the entire connected path.

E. Graph Propagation With Out Edge Failure

In this section, a text box is provided to find the path between two terminals. In textbox, enter from terminal, to terminal. For example (1,5). Click ‘ Path’ button. The entire path from terminal 1 to 5 will be displayed. First all the terminals connected to ‘ from terminal’ are found out. Then from all the terminals, the next links up to the ‘ to terminal’ is found out.

For example, {1,3,5}, {1,4,5}, {1,3,4,5}, {1,3,6,5} are the path for 1,5. When you click the first listbox (left side), the total distance will be displayed in the bottom label. The distance for all the paths is calculated inside the coding. Of which the shortest distance is also calculated. For example 30 is the shortest distance and (Both the path 1, 4 ,5 and 1, 3, 6 and 5 are having distance as 30, then the two paths are listed in second list box. The hop count is displayed in the third list box control. For example Hop count is 3 for [1,4,5] and 4 for [1,3,6,5]. One of the shortest path distances can be chosen to travel.

F. Graph Propagation with Edge Failure

In this section, all options are same as above form. In addition, failed path is displayed in ‘ failed path’ listbox (left side bottom). For example, in created network menu options form If you click create graph from table button, the graph is displayed. If you right click the node label in the graph, then the node is set as Inactive. ‘ Node’ tables record is having a field with name Status. It changes to Inactive. If the paths (listed in first list box) contains that failed node (In real time, the road segment is failed/block due to accident or meeting like that) then the path will not be displayed (in fourth list box – left side bottom list box). Fifth and sixth list boxes show the path values and hop counts. So the entire path as well as failed path can be viewed, so that one can omit the failed path during traveling.

G. Time-Dependent Landmark Graph

This section first describes the construction of the time dependent landmark graph, and then details the travel time estimation of landmark edges.

In practice, to save energy and communication loads, taxis usually report on their locations in a very low frequency, like 2-5 minutes per point. This increases the uncertainty of the routes traversed by a taxi [3], [4].

Meanwhile, we cannot guarantee there are sufficient taxis traversing on each road segment anytime even if we have a large number of taxis. That is, we cannot directly estimate the speed pattern of each road segment based on taxi trajectories. In our method, we first partition the GPS log of a taxi into some taxi trajectories representing individual trips according to the taximeters transaction records. There is a tag associated with a taxis reporting when the taximeter is turn on or off, i.e., a passenger get on or off the taxi. Then, we employ our IVMM algorithm [4], which has better performance than existing map-matching algorithms when dealing with the low-sampling rate trajectories.

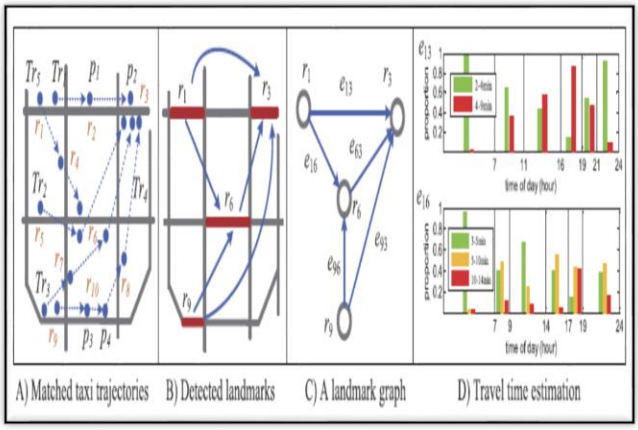

Fig 1 Landmark graph Construction

Therefore, they are build two different landmark graphs for weekdays and weekends, respectively. That is, we paper all the weekday trajectories (from different weeks and months) into one weekday landmark graph, and put all the weekend trajectories into the weekend landmark graph. We also find that the traffic pattern varies in weather conditions. Therefore, we, respectively, build different landmark graphs for weekday and weekend, and for normal and severe weather conditions, like storm, heavy rain, and snow. In total, 2 _ 2 ¼ 4 landmark graphs are built. The weather condition records are crawled from the weather forecast website.

If we set k = 4, the top-4 road segments (r1, r3, r6, r9) with more paperions are detected as landmarks. Note that the consecutive points (likep4) from a single trajectory (Tr4) can only be counted once for a road segment (r10). This aims to handle the situation that a taxi was stuck in a traffic jam or waiting at a traffic light where multiple points may be recorded on the same road segment (although the taxi driver only traversed the segment once), as shown in Fig. 3C. After the detection of landmarks, we convert each taxi trajectory from a sequence of road segments to a landmark sequence, and then connect two landmarks with an edge if the transitions between these two landmarks conform to Definition 3.4 (supposing _ ¼ 1 in this example).

H. Rough Route Smoothing

In Rough Route Smoothing, Given a rough route Rough : qs □ l1 □ l2 □ l3 □ ,,, □ ln-1 □ ln □ qd, we say Rough satisfies

- Source-Farther Principle if □□i = 1, 2, ... n-1, dist(li+1;qs) > dist(li;qs)
- Destination-Closer Principle if □□i = 1, 2, ... n-1, dist(li+1;qd) > dist(li;qd)
- Dist(l¬;qd) < dist(li,qd), and
- Next-Nearest Principle if □□i = 1, 2, ... n-1, dist(li;li+1) = minj>i{dist(li;lj)},
- Where dist(li,lj) is the road network distance from li to lj.

The source-farther principles states that the distance from each landmark to the source should be farther than its previous one. The destination-closer principles state that each landmark should be closer than its previous landmark to the destination. The next-nearest principle (also termed as " non turn-back" principle) states that the next landmark li+1 should be the nearest landmark of li among all the landmarks after li. Local smoothing and Global smoothing is carried out in this module.

Local smoothing. This step aims to find the longest subsequence from the resulting sequence of the global smoothing so as to satisfy the next-nearest principle. It's clear that the brute-force algorithm which checks all the subsequences takes exponential time.

I. Refined Routing

In refined routing, Suppose after the smoothing, we get a rough route Rough: qs □ l'1 □□1'2 □□□' □□□□□' n-1□□□' n□□qd. This stage finds in the real road network a detailed fastest route that sequentially passes the landmarks of a rough route by dynamic programming. Assume r1,r2, ... ,rn are the corresponding road segments of l'1, l'2, … , l'n, i.e., ri = l'i. Using these notations, we have the initial states fs(1) and fe(1) as follows:

fs(1)= T(qs,r1.e, r1.s)+ tes(1)

fe(1)= T(qs,r1.s, r1.e)+ tse(1).

Let Tise = T(ri.s, ri+1.e, ri+1.s) denote the time of the fastest route (using speed constraint in real road network) which starts from point ri.s and ends at point ri+1.e without crossing ri+1.s in road network Gr. Then, Tiee, Tiss, and Ties can be similarly defined. Now, we have the state transition equations.

After fs(n) and fe(n) are computed, the total travel time for the optimal route in the real road network is

For the proposed refined routing, the following processes are carried out.In addition with the above mentioned rough routing algorithm, we enhance the methodology. It helps for finding peak hour affected path (A) and finding path with road failure occurred in the intermediate hops (B). For that, during the road segment data collection, whether it is affected by heavy traffic during peak hour is also gathered to

find (A). In addition, the terminal failed data is also collected during finding (B). The above said rough route smoothing algorithm is applicable to both the algorithms discussed below. The algorithm for (A) to list the peak hour affected paths details is listed below.

Input:

Nodes List NL, Edges List EL, Peak Hour Affected Edges E' , Starting Node SN, Ending Node EN

Output:

List of Path available L and Peak Hour Affected Path PHL

1. N' <- Starting Node SN

2. Find All Nodes N that is neighbors to the N'

3. P <- N'

4. For each n □□□

5. Propagate the next hop nodes such that 'n' as "from node in Edge" and next hop node EN' as "to node in Edge" until the EN is found out. Add the Node EN' to P

6. Add the Path P to L

7. If P contains E' Add the Path P to PHL

8. Next

9. Return L and PHL

For (A), all the road terminals (nodes list), road segments (edges list), road segments affected in peak hours (peak hour affected edges) are given. Source and destination terminal is given to find all the available paths.

The algorithm in addition with finding all the paths, path disturbed during heavy traffic is also found out. So, using the above said algorithm, for the given source and destination road terminal, all the available routes as well as road segments which are affected by heavy traffic in peak hour schedule is listed out.

The drive is able to know the path to be avoided in peak hour so that fastest driving to destination can be achieved.

The algorithm for (B) to list the road failure paths details is listed below.

Input:

Nodes List NL, Edges List EL, Failure Nodes FN, Starting Node SN, Ending Node EN

Output:

List of Path available L and Road Failure Path RFP

- N' <- Starting Node SN
- Find All Nodes N that is neighbors to the N'
- P <- N'
- For each n □□□
- Propagate the next hop nodes such that 'n' as "from node in Edge" and next hop node EN' as "to node in Edge" until the EN is found out. Add the Node EN' to P
- Add the Path P to L
- If P contains E' Add the Path P to RFP
- Next
- Return L and RFP

For (B), all the road terminals (nodes list), road segments (edges list), failure terminals are given. Source and destination terminal is given to find all the available paths. The algorithm in addition with finding all the paths, path containing failure terminal is also found out.

IV. EXPERIMENTAL RESULTS

The following Figure 2 describes experimental result for number of node participate routing process in existing and proposed accuracy analysis. The table contains number of travel node with in time interval (60 km/hour), existing routing travel node path accuracy and proposed routing node path accuracy details are shown.

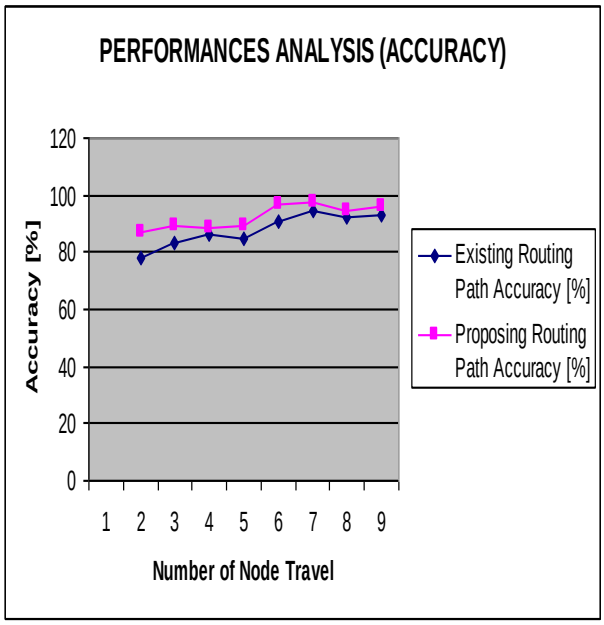

Figure 2 Existing and Proposed Accuracy Analysis

The following Table 2 describes experimental result for number of node participate routing process in existing and proposed failure rate analysis. The table contains number of travel node with in time interval (60 km/hour), existing routing travel node failure rate and proposed routing node failure rate details are shown.

V. CONCLUSION

This paper proposed a novel problem as minimum path pair (MPP) in road networks. For two users, it finds the shortest paths with meeting uncertainty. Firstly, it devises two simple algorithms by enumerating path pairs and node pairs, respectively. We prove that candidate pairs should be inside a limited region, and the others can be pruned. It further proposes a fast algorithm based on point-to-point shortest path query to improve query efficiency. Furthermore, the experiment results illustrate that MPP well balance the needs of taking the shortest routes to destinations

and meeting before going to destinations. The paper provides a best assistance in finding shortest path. The application become useful if the below enhancements are made in future.

If the application is designed as web service, it can be integrated in many web sites. Peak hours and non-peak hours' road traffic situation can be taken into account for future study. The application is developed such that above said enhancements can be integrated with current modules.

REFERENCES

[1] G. Eason, B. [1] [1] I. Abraham, D. Delling, A. V. Goldberg, and R. F. F. Werneck, " Hierarchical hub labelings for shortest paths," in ESA, 2012.

[2] [2] [2] R. Jin, N. Ruan, Y. Xiang, and V. E. Lee, " A highway-centric labeling approach for answering distance queries on large sparse graphs," in SIGMOD, 2012.

[3] [3] [3] T. Akiba, Y. Iwata, and Y. Yoshida, " Fast exact shortest-path distance queries on large networks by pruned and mark labeling," in SIGMOD, 2013.

[4] [4] C. Sommer, " Shortest-path queries in static networks," ACM Comput. Surv., vol. 46, no. 4, 2014.

[5] [5] [5] A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, and S. Zhou, " Shortest path and distance queries on road networks: towards bridging theory and practice," in SIGMOD, 2013.

[6] [6] .Chang,X.Lin,L.Qin,J.X.Yu,andJ.Pei, "Efficiently computing top-k shortest path join," in EDBT, 2015.

[7] [7] T. Hunter, R. Herring, P. Abbeel, and A. Bayen, "Path and Travel Time Inference from GPS Probe Vehicle Data," Proc. Neural Information Processing Systems (NIPS), 2009.

[8] [8] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, " Map-Matching for Low-Sampling-Rate GPS Trajectories," Proc. Int' l Conf. Advances in Geographic Information Systems (GIS), 2009.

[9] [9] J. Yuan, Y. Zheng, C. Zhang, and X. Xie, " An Interactive-Voting Based Map Matching Algorithm," Proc. Int' l Conf. Mobile Data Management (MDM), 2010.

[10] [10] E. Kanoulas, Y. Du, T. Xia, and D. Zhang, " Finding Fastest Paths on a Road Network with Speed Patterns," Proc. Int' l Conf. Data Eng. (ICDE), 2006.