A Predictive Location Model for Location-Based Services

Hassan A. Karimi and Xiong Liu
Dept. of Information Science and Telecommunications
University of Pittsburgh
135 N. Bellefield Ave., Pittsburgh, PA 15260
hkarimi@sis.pitt.edu, xliu@mail.sis.pitt.edu

ABSTRACT

Location-Based Services (LBSs) utilize information about users' locations through location-aware mobile devices to provide services, such as nearest features of interest, they request. This is a common strategy in LBSs and although it is needed and benefits the users, there are additional benefits when future locations (e.g., locations at later times) are predicted. One major advantage of location prediction is that it provides LBSs with extended resources, mainly time, to improve system reliability which in return increases the users' confidence and the demand for LBSs. However, much of the current location prediction research is focused on generalized location models, where the geographic extent is divided into regular-shape cells. These models are not suitable for certain LBSs whose objective is to compute and present on-road services, because a cell may contain several roads while the computation and delivery of a service may require the exact road on which the user is driving. We propose a new model, called Predictive Location Model (PLM), to predict locations in LBSs with road-level granularities. The premise of PLM is geometrical and topological techniques allowing users to receive timely and desired services.

Categories and Subject Descriptors

H.2 [**Information Systems**]: Database Management; H.2.8 [**Database Management**]: Database Applications – *spatial databases and GIS*

General Terms

Management, design, reliability.

Keywords

Location-based services, location prediction, location management, mobility, trajectory, database, probabilistic method.

1. INTRODUCTION

The recent convergence of Internet, wireless communications, mobile location-aware clients, and geoprocessing has given rise to a new generation of Location-Based Services (LBSs) [1]. The premise of this new generation LBSs is a distributed mobile computing environment where the geographic locations of the clients in space are utilized for computing and application-related optimization. LBS architectures feature location-aware devices, which are equipped with geopositioning systems, interconnected through wired and wireless networks. Following are example LBSs reported in the literature: a guidance system with caching function [2]; a context-aware tour guide system [3]; a LBS framework using Cellular Digital Packet Data (CDPD) [4]; a nearest available parking lot application [5].

Knowledge about locations of mobile devices is the basic requirement for LBSs. There are a number of approaches for determining location of a mobile client, each requiring a different infrastructure and resulting in a different accuracy level. Of these, time difference of arrival (TDOA), angle of arrival (AOA), location pattern matching (LPM), and the Global Positioning System (GPS) [6] are widely used. While GPS is only for outdoor LBSs, TDOA and AOA can be used for indoor LBSs.

Current LBSs utilize information about locations of users to determine such services as the nearest features of interest from a location. The general assumption in these systems is that the area centered at the current location of the user is where the services are needed. Although this assumption is valid and used as the basis of many computing strategies in LBSs, there are additional benefits when future locations (e.g., locations at later times) are also predicted. Location prediction provides a longer time available to prepare and present services, especially services involving complex and time consuming tasks (e.g., mobile electronic commerce), and to ensure that only desired services are delivered. For example, location-based predictive caching strategies have been proposed to deal with handoff latency in mobile IP [7]. In addition, having the priori information about locations where the user will visit at later times during a trip will extend the location management capability of LBSs and will facilitate the generation of new services that were not possible previously. For instance, a service that allows the user to plan a purchase stop for a later time.

In this paper, we present and discuss a new predictive location model in LBSs — the main contribution of the paper. The paper is organized as follows. In the next section, previous work related to location prediction in LBSs is discussed and selected existing location prediction methods and their limitations are overviewed. Then PLM is described in detail. Next, an analysis of PLM is discussed. Finally, conclusions are given.

2. RELATED WORK

In a mobile environment, information about a user's location includes two parameters, L and t, indicating that the user is at location L at time t. Location prediction dynamically estimates the mobile user's future locations using the user's current location information, the historical mobility patterns and the auxiliary information. Two necessary steps in location prediction include mobility realization and location determination. Mobility realization is to determine the trajectory of a mobile user while location determination is to estimate a location on that trajectory or the time when the user will reach a given destination. Different prediction strategies, each for a different purpose, have been reported in the literature. Current strategies may be classified into two approaches: *cell-based* approach and *map-based* approach.

In the cell-based approach [8, 9, 10, 11, 12, 13], the geographic extent is divided into regular-shape cells (e.g., hexagon). The cells are usually determined by the architecture of the cellular network, while they can also be defined for computational purposes [12, 13]. A location is usually determined by a method called paging in which a piece of search information is broadcast to every cell in the region, and the mobility is expressed as a series of cells making the trajectory of a mobile user.

The cell-based approach has played an important role in mobile networks to improve system performance such as pre-caching [12, 13]. However, this approach has the following inherent shortcomings:

1) It cannot precisely locate a mobile user. The radius of a typical cell may be 150 meters and upward (a cell could have a radius of more than 30,000 meters) [14];

2) It cannot precisely model the trajectory of a user because it does not support fine granularity (e.g., road-level granularity). It can only calculate the cell change probability based on the side through which the user leaves the cell. Therefore, it cannot precisely estimate the travel distance to a destination and consequently the time to deliver services.

On the contrary, the premise of the map-based approach [15, 16] is to determine a user location as a point on a road instead of a cell using such geopositioning systems as GPS instead of paging. To that end, the assumption of the map-based approach is that trips are made through vehicles thus constrained to roads in road networks. This assumption is made based on the fact that in most travel surveys responders identify vehicles as the main means of transport [17]. Another feature of the map-based approach is the availability of a variety of data sets (e.g., road networks, buildings, parks) for predicting mobility and providing services [18].

In map-based LBSs, if a user provides a destination, a conventional routing algorithm (e.g., the shortest-path algorithm) can be used to predict a trajectory and estimate the time when the user will reach there. However, in many LBSs the user's destination is unknown where conventional routing algorithms cannot be used. The user usually has many route choices when presented with a road network. This leads to the research issue of uncertainty management in trajectory and location prediction. However, studies [9, 19, 20] have reported that users often have some degree of regularity in their motion. For example, [19] studied the trajectories users follow and conducted an experiment over a period of six weeks and found that the users tend to follow regular trajectories (e.g., from home to a second location) more than 70% of times. Thus historical travel information plays an important role in developing location prediction models.

We propose and discuss a new map-based location prediction model. The model comprises a database module, an information retrieval module, a trajectory prediction module, a location determination module and an error control mechanism.

3. PLM

Before presenting PLM, we first give some formal definitions. User's *current location* L_k is defined as the measured geographic coordinates (latitude and longitude) by a geopositioning system at the current time t_k, where k is the ID for a particular user location. *Prediction period* T is defined as the time duration between the current time t_k and a later time t_{k+1}. Given the current location (L_k, t_k) and a prediction period T, the problem of location prediction is to estimate location L_{k+1} at time t_k+T or t_{k+1}.

3.1 Architecture

Of the different infrastructures possible for LBSs, the one with mobile clients and fixed servers is assumed. The mobiles are linked to the servers via wireless communications; standard network speeds are expected from such wireless communications (e.g., 144kb/sec-2mb/sec in 3G services [21]). Furthermore, there are only mobile-server communications and not mobile-mobile communications. Each mobile user has a mobile device which is equipped with a GPS receiver.

3.2 PLM Database (PLM-DB)

PLM requires a database, called PLM-DB, which contains two categories of data. One category, called *service area*, includes such spatial data as road network (see Section 3.2.1), landmarks, restaurants, etc. The second category includes user-specific data such as historical trajectories (see Section 3.2.2) and user profile information (e.g., age, profession, interests and home address). In addition, the second category features probabilistic information inferred from historical trajectories (see Section 3.2.3).

3.2.1 Road Network

A road network (Figure 1) consists of edges and vertices, where each edge is a road segment (a portion of a road) and each vertex is an intersection. A road network can be modeled as a graph G = <V, E> where V is the list of all intersections with a size |V| and E is the list of all road segments with a size |E|. Each road segment e may have such attributes as speed limit, length and road width. In practice, each road segment in the network has a direction, meaning either a one-way road or a two-way road. If direction is not considered, G is a symmetric (non-directed) graph. However, if direction is considered, G becomes asymmetric. In this paper, we assume symmetric road networks throughout. Figure 2 shows the graph, represented in an adjacency matrix, for the road network in Figure 1.

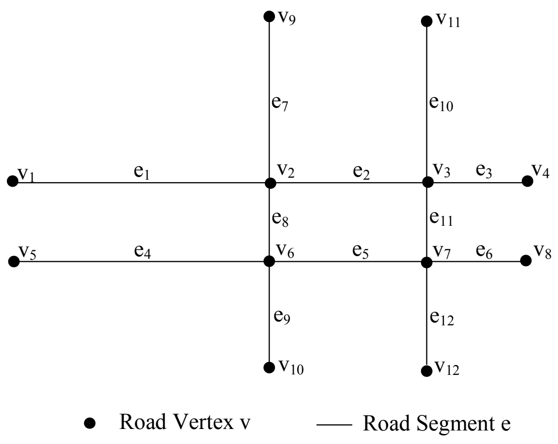

Figure 1. A simple road network G

$$
\begin{array}{c|cccccccccccc}
 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} \\
\hline
v_1 & 0 & e_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v_2 & e_1 & 0 & e_2 & 0 & 0 & e_8 & 0 & 0 & e_7 & 0 & 0 & 0 \\
v_3 & 0 & e_2 & 0 & e_3 & 0 & 0 & e_{11} & 0 & 0 & 0 & e_{10} & 0 \\
v_4 & 0 & 0 & e_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v_5 & 0 & 0 & 0 & 0 & 0 & e_4 & 0 & 0 & 0 & 0 & 0 & 0 \\
v_6 & 0 & e_8 & 0 & 0 & e_4 & 0 & e_5 & 0 & 0 & e_9 & 0 & 0 \\
v_7 & 0 & 0 & e_{11} & 0 & 0 & e_5 & 0 & e_6 & 0 & 0 & 0 & e_{12} \\
v_8 & 0 & 0 & 0 & 0 & 0 & 0 & e_6 & 0 & 0 & 0 & 0 & 0 \\
v_9 & 0 & e_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v_{10} & 0 & 0 & 0 & 0 & 0 & e_9 & 0 & 0 & 0 & 0 & 0 & 0 \\
v_{11} & 0 & 0 & e_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v_{12} & 0 & 0 & 0 & 0 & 0 & 0 & e_{12} & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Figure 2. Adjacency matrix for G

3.2.2 *Historical Trajectories*

A user moving between two locations in a road network generates a trajectory. We define a user's *trajectory* as a sequence of connected road segments or a sequence of connected vertices between two locations, namely start point and end point. For example, if the user moves from v_1 to v_{10} through v_2 and v_6 in Figure 1, then the trajectory is represented as Traj(e_1, e_8, e_9). The GPS can record the user's locations at fixed (distance or time) intervals and the speed at each location. These recorded locations together with speed information can be map-matched with the road network to generate a trajectory. The computed trajectories are transmitted to the server and stored in PLM-DB. The trajectory data contains valuable information about a specific user who travels regularly in the service area. When a user is new to the service area, he/she does not have historical trajectories. In this case, the server uses a trajectory derived from the trajectories of all users. For PLM, we assume that the user travels regularly in the service area.

3.2.3 *Probability Matrix*

Probabilistic methods are a promising approach in modeling uncertainties associated with trajectory prediction. We observe that intersections are usually where the uncertainty happens which leads to the different trajectory choices. To capture the probabilistic information at an intersection v with n road segments, we define a probability matrix as follows:

$$
M(v, t_k) = \begin{bmatrix} P(R_1 \mid R_1) & P(R_2 \mid R_1) & \dots & P(R_n \mid R_1) \\ P(R_1 \mid R_2) & P(R_2 \mid R_2) & \dots & P(R_n \mid R_2) \\ \dots & \dots & \dots & \dots \\ P(R_1 \mid R_n) & P(R_2 \mid R_n) & \dots & P(R_n \mid R_n) \end{bmatrix} \quad (1)
$$

where t_k is the current time, and $R_1, R_2, \dots R_n$ are road segments that share v. $P(R_1 \mid R_1)$ is the probability of taking R_1 when the user in currently on R_1, and $P(R_2 \mid R_1)$ is the probability of taking R_2 when the user in currently on R_1, and so on.

Historical trajectory information stored at the server can be used to infer the number of times the user has traveled on each road segment and the trajectory choice at each intersection. The data is then used to calculate the parameters of the probability matrix. Therefore, $M(v, t_k)$ can be expressed as:

$$
M(v, t_k) = \begin{bmatrix} N_{1,1}/N_1 & N_{2,1}/N_1 & \dots & N_{n,1}/N_1 \\ N_{1,2}/N_2 & N_{2,2}/N_2 & \dots & N_{n,2}/N_2 \\ \dots & \dots & \dots & \dots \\ N_{1,n}/N_n & N_{2,n}/N_n & \dots & N_{n,n}/N_n \end{bmatrix} \quad (2)
$$

where N_i is the number of times the user has traveled on road segment R_i. $N_{j,i}$ is the number of times the user has taken road segment R_j when he/she is on road segment R_i. In other words, the following should hold:

$$
\sum_{j=1}^{n} N_{j,i} = N_i \quad (3)
$$

With the assumption that no U-turn (at intersections) is allowed, $N_{i,i} = 0$. If there is no historical data about the user's travel at v, the model may assume that the probabilities of turning to R_j ($j \neq i$) from R_i are equal. For example, when the user in on e_1 (e_1 is an instance of R_i) and moves toward v_2 in Figure 1, $N_{2,1} = N_{8,1} = N_{7,1}$. After the user takes one of the roads at v_2, $N_1 = 1$. Since $N_{2,1} + N_{8,1} + N_{7,1} = N_1$, we get $N_{2,1} = N_{8,1} = N_{7,1} = 1/3$. It should be noted that N_i and $N_{j,i}$ are all integer numbers. For consistency, the model initializes N_1 to 3 instead of 1. Therefore, $N_{2,1} = N_{8,1} = N_{7,1} = 1$. With these numbers, the initial matrix M of v_2 at t_0 is as follows:

$$
M(v_2, t_0) = \begin{bmatrix} P(e_1|e_1) & P(e_2|e_1) & P(e_8|e_1) & P(e_7|e_1) \\ P(e_1|e_2) & P(e_2|e_2) & P(e_8|e_2) & P(e_7|e_2) \\ P(e_1|e_8) & P(e_2|e_8) & P(e_8|e_8) & P(e_7|e_8) \\ P(e_1|e_7) & P(e_2|e_7) & P(e_8|e_7) & P(e_7|e_7) \end{bmatrix}
$$

$$
= \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix} \quad (4)
$$

Figure 3 illustrates the generalized probability matrix initialization algorithm called *ProbInitial*. The algorithm takes a road network G and creates a probability matrix for all vertices in the network.

The travel information stored in the trajectory data is retrieved to update the probability matrix. If the past trajectory information indicates that the user has crossed Intersection v from R_i, both N_i and $N_{j,i}$ in Equation (2) will change. The N_i will increment by one, and one of the $N_{j,i}$ will also increment by one. For example, if the user takes e_8 from e_1 at v_2, the matrix M at time t_k becomes:

$$M(v_2,t_k) = \begin{bmatrix} 0 & 1/4 & 2/4 & 1/4 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix} \quad (5)$$

Therefore, matrix M for each intersection is dynamic and is updated periodically. We assume that each time the user logs into the server, the mobile device will transmit the newly colleted trajectories, and the server will update the probability matrix using the new trajectories.

```
ProbInitial(G)                    //G = <V, E>
1.  create list ProbML            //ProbML stores all probability matrices
2.  for each v in V do
3.      Edg = edges linked to v
4.      N = size of Edg
5.      create a N by N matrix M for v
6.      set all elements of M to 1/(N-1)
7.      set all diagonal elements to zero
8.  return ProbML
```

Figure 3. ProbInitial algorithm

```
ProbUpdate(G, TrajSet)            //G = <V, E>
                                  //TrajSet is the set of trajectories
1.  for each trajectory t in TrajSet do
2.      for each road segment $e_i$ of t do
3.          find the vertex v between $e_i$ and $e_{i+1}$
4.          find the probability matrix M of v and read the row of $e_i$
5.          Numer = read the numerator of the element ($e_i$, $e_{i+1}$)
6.          Denom = read the denominator of the element ($e_i$, $e_{i+1}$)
7.          set the element ($e_i$, $e_{i+1}$) to (Numer+1)/(Denom+1)
8.          set all other elements in the row of $e_i$ to (Numer)/(Denom+1)
9.  return
```

Figure 4. ProbUpdate algorithm

Figure 4 illustrates the generalized probability matrix update algorithm called *ProbUpdate*. There are two loops in the algorithm. The first one loops through all trajectories, and the second one loops through the sequence of road segments within each trajectory. The output is the updated probability matrix.

3.3 Information Retrieval for Prediction

Considering the geographic range of a service area (e.g., the Pittsburgh metropolitan), it is impractical to retrieve the entire PLM-DB because the user's activity range may only contain a small portion of the service area. We propose a concept called dynamic computational window (DCW) to deal with this problem. A DCW is defined as a circular clipping window that centers at the user's current location to retrieve information from PLM-DB for location prediction. The radius of DCW is determined by both the user's travel speed and the prediction period T. While T is predetermined by the system, the user's travel speed is an uncertain factor. When the user drives on a highway, the speed can be estimated as the speed limit of the highway, however when the user drives in local roads, the user's speed is affected by the traffic lights and the traffic. For simplicity, we assume that the user's speed is the same as the speed limit of the road on which the user is driving, SpeedLimit(t_k). Therefore, the size of a DCW, R, can be determined by:

$$R = \text{SpeedLimit}(t_k) \cdot T \quad (6)$$

For example, if the user is driving on a 25-mile/hr road and the prediction period time is 30 minutes, then the size of a DCW is 12.5 miles.

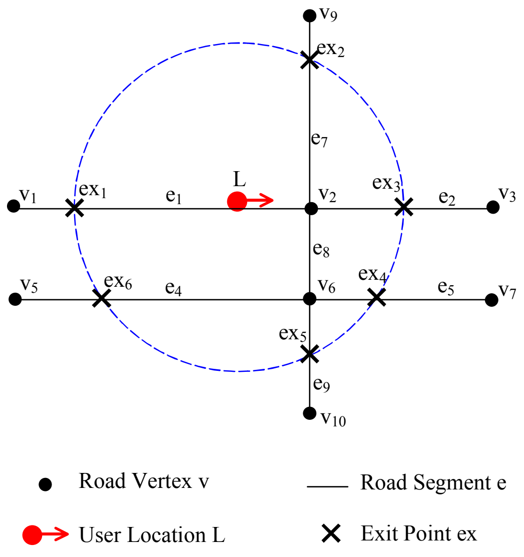

Figure 5. DCW and sub-network G'

$$\begin{array}{c|cccccccc} & v_1 & v_2 & v_3 & v_5 & v_6 & v_7 & v_9 & v_{10} \\ \hline v_1 & 0 & e_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_2 & e_1 & 0 & e_2 & 0 & e_8 & 0 & e_7 & 0 \\ v_3 & 0 & e_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_5 & 0 & 0 & 0 & 0 & e_4 & 0 & 0 & 0 \\ v_6 & 0 & e_8 & 0 & e_4 & 0 & e_5 & 0 & e_9 \\ v_7 & 0 & 0 & 0 & 0 & e_5 & 0 & 0 & 0 \\ v_9 & 0 & e_7 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_{10} & 0 & 0 & 0 & 0 & e_9 & 0 & 0 & 0 \end{array}$$

Figure 6. Retrieved adjacency matrix for G'

Figure 5 depicts a DCW (dashed circle) for a user driving on road e_1 in the road network shown in Figure 1. Because a DCW is a clipping window, it cuts the road network into a smaller portion. Therefore, only the road network contained in the DCW is retrieved for prediction purposes. We consider all the vertices inside the DCW and their immediate (neighbor) vertices as the vertices of the sub-network. The sub-network is a new graph G' = <V', E'> where V' is the list of all retrieved vertices with a size |V'| and E' is the list of all retrieved edges with a size |E'|. Figure 5 shows the retrieved sub-network from the network in Figure 1, and the adjacency matrix for G' is shown in Figure 6.

DCW represents the largest geographic extent within which the user's activities are expected during the prediction period T. We define the intersections between the DCW and the sub-network as *exit points* because they represent all the possible exit locations where the user may leave the DCW (see Figure 5). Exit points may be considered as destinations of predicted trajectories.

The number of exit points can be inferred from the topological information in G'. Suppose the number of vertices inside a DCW is N ($N \leq |G'|$ because G' may include vertices outside the DCW), we define a new graph G" = <V", E"> where V" is the list of the N vertices inside the DCW and E" is the list of edges between those N vertices. For each vertex u inside the DCW, we know the number of edges linked to it or the number of neighbor vertices (num_neighbors(u)) by looking up the row of u in the adjacency matrix of G'. If we sum up the number of edges linked to each vertex in G" as $\sum_{u(G'')} \text{num\_neighbors}(u)$, we get a duplicated count of those edges in G", which is exactly twice the number of edges ($2|E''|$). Therefore, if we subtract the duplicated count of edges in G" from $\sum_{u(G'')} \text{num\_neighbors}(u)$, we get the number of edges that intersect with the DCW, which is also the number of exit points (num_Exp):

$$\text{num\_Exp} = \sum_{u(G'')} \text{num\_neighbors}(u) - 2|E''| \quad (7)$$

For the example shown in Figure 5, v_2 has four edges (e_1, e_2, e_8, e_7) linked to it and four neighbor vertices (v_1, v_3, v_6, v_9), similarly v_6 also has four edges (e_8, e_4, e_5, e_9) and four neighbor vertices (v_2, v_5, v_7, v_{10}). G" is a graph including only v_2 and v_6, and there is only one edge e_8 in G". Therefore, the number of exit points is $(4 + 4) – 2 = 6$, as shown.

3.4 Trajectory Prediction

PLM utilizes the information in G' and the exit points to predict trajectories. The start point is the user's current location while the end point could be any of the exit points. A shortest-path algorithm can be used to compute a trajectory between the current location and each of the exist points. However, each computed trajectory has the same probability. This is a problem that needs to be resolved, especially when the number of computed trajectories is large. In addition, the shortest-path algorithm can only compute one trajectory for each given pair of start and end points. This will again decreases the system's ability to handle uncertainty because a user may have alternative trajectories to reach a destination from a start point. We propose a method to detect all candidate trajectories and rank them based on the probabilistic information stored. By doing so, the system is provided with a mechanism that identifies all options and select appropriate ones for location prediction.

3.4.1 Candidate Trajectories

We propose a graph search tree to detect candidate trajectories in a sub-network G', retrieved by a DCW, where the root is the user's current location L, the intermediate nodes are the vertices in G', the leaf nodes are the exit points, and the linkages between nodes are the road segments. To decrease the search space, we assume the following rules:

1) The user does not turn around at an intersection;
2) The user does not visit a vertex that has already been visited during a trip.

Figure 7 depicts one search tree constructed for the DCW in Figure 5. Clearly, certain braches are pruned by the rules of the search tree.

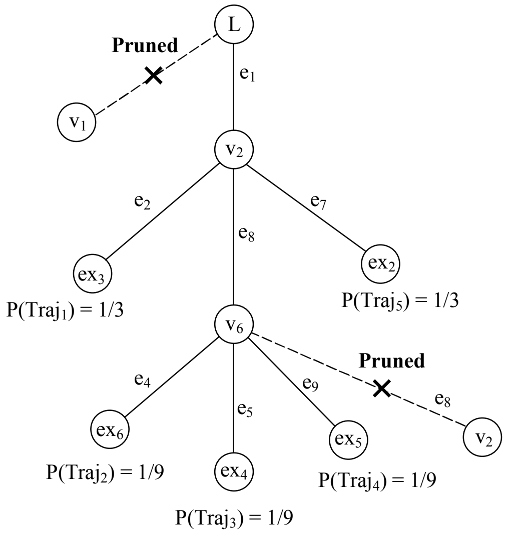

Figure 7. Trajectory search tree

```
TrajSearch(G', L, ExpSet)         //G' = <V', E'>
                                  //L is current location
                                  //ExpSet is the set of exit points
1. for each u in V' do
2.    status[u] = "unprocessed"
3. create list Vetx               //Vetx stores vertices of a trajectory
4. create list Traj               //Traj stores all trajectories
5. i = 0                          //i stores the number of trajectories
6. u0 = the nearest vertex of L
7. DFS(u0)                        //begin depth-first search from L
8. return Traj

DFS(u)
1. status[u]= "processed"
2. add u to Vetx
3. for each neighbor vertex v of u do
4.    e = the edge between u and v
5.    if status [v] != "processed" and
      e does not intersect with any exit point in ExpSet then
6.       DFS(v)
7. add Vetx to Traj[i] and increment i by one
8. status[u] = "stopped"
9. remove u from Vetx
10. return
```

Figure 8. TrajSearch algorithm

Figure 8 illustrates the general algorithm called *TrajSearch* for detecting candidate trajectories. The algorithm takes the sub-network G', the exit point set ExpSet and the current location L. It first initialize the vertices in G' to an unprocessed status. Then it calls a Depth First Search (DFS) algorithm to detect the candidate trajectories. DFS first takes the nearest vertex u_0 of L based on the moving direction of the user. It then changes the status of u_0 to "processed" and finds all the children (neighbor vertices) of it. Then the children become the parents by calling DFS (see Line 6 in DFS), the recursive process continues until one exit point is detected (see Line 5 in DFS). DFS then changes the status of the input vertex to "stopped", meaning one trajectory has been detected.

There is one loop in the algorithm which goes through all neighbor vertices of a particular vertex u. All other operations performed within the loop, such as changing status, have O(1) time complexity. The loop through all neighbors of all vertices in G' takes $O\left(\sum_{u(G')} \text{num\_neighbors}(u)\right) = O(2|E'|)$. Therefore, the overall time complexity is O(|E'|), where |E'| is the number of edges in G'.

3.4.2 *Trajectory Probability*

Given a trajectory Traj(e_1, e_2, e_3, …, e_n), we denote its probability by P(Traj) as a joint probability of its edges P(e_1, e_2, e_3, …, e_n). The probabilities of the candidate trajectories detected by the search tree can be calculated using the probability matrices (see Section 3.2.3) at the vertices and the Bayesian theorem:

$$P(b, a) = P(b|a) \cdot P(a) \quad (8)$$

where P(b,a) is the probability that b and a occur, P(b|a) is the probability that b occurs given that a has already occurred, and P(a) is the probability that a occurs.

For the search tree in Figure 7, assume vertex v_2 has an initial probability matrix as defined in Equation (4) and vertex v_6 also has an initial probability matrix:

$$M(v_6, t_0) = \begin{bmatrix} P(e_8|e_8) & P(e_4|e_8) & P(e_5|e_8) & P(e_9|e_8) \\ P(e_8|e_4) & P(e_4|e_4) & P(e_5|e_4) & P(e_9|e_4) \\ P(e_8|e_5) & P(e_4|e_5) & P(e_5|e_5) & P(e_9|e_5) \\ P(e_8|e_9) & P(e_4|e_9) & P(e_5|e_9) & P(e_9|e_9) \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix} \quad (9)$$

Because the user is currently on edge e_1, the probability P(e_1)=1. By reading the probability parameter P(e_2|e_1) for v_2, we get P(e_2, e_1) = P(e_2|e_1) · P(e_1) = 1/3. Similarly, P(e_1,e_8) = P(e_1,e_7) = 1/3. By reading the probability parameter P(e_4|e_8) for v_6, we get P(e_1,e_8,e_4) = P(e_1) · P(e_8|e_1) · P(e_4|e_1,e_8) = 1·(1/3)·(1/3) = 1/9 (note that P(e_4|e_1,e_8) = P(e_4|e_8)). Similarly, P(e_1,e_8,e_5) = P(e_1,e_8,e_9) = 1/9. The probabilities for the trajectories are labeled at the leaf nodes (exit points) in the search tree (see Figure 7) and the probability calculation can be carried out as the tree searches candidate trajectories.

3.5 Location Determination

The trajectories detected can be ranked based on their probabilities. We define those trajectories with probabilities over a predetermined threshold value as *regular trajectories*. From a probability point of view, regular trajectories are the most possible trajectories that the user will take. The server may use regular trajectories for location determination. Once candidate trajectories are ranked, locations can be estimated by using extrapolation with a distance determined by Equation 6.

3.6 The Generalized Model

Figure 9 illustrates the major steps of our proposed PLM, where the client transmits GPS data to the server and provides the server with current location information for trajectory and location prediction. The DCW module is responsible for retrieving information for prediction. The trajectory prediction module uses the topological information of road network and the probabilistic information from historical trajectories to predict a trajectory and the location determination module uses the geometrical information (extrapolated distance) to estimate a location. After the locations of services are predicted, the system will prepare services and deliver them. In case of incorrect prediction, an error control mechanism is used. The mechanism specifies an agreement between the server and the client such that the client will update the server if and only if the deviation or error (e.g., the nearest distance) between the predicated trajectory and the actual trajectory is greater than a predetermined value.

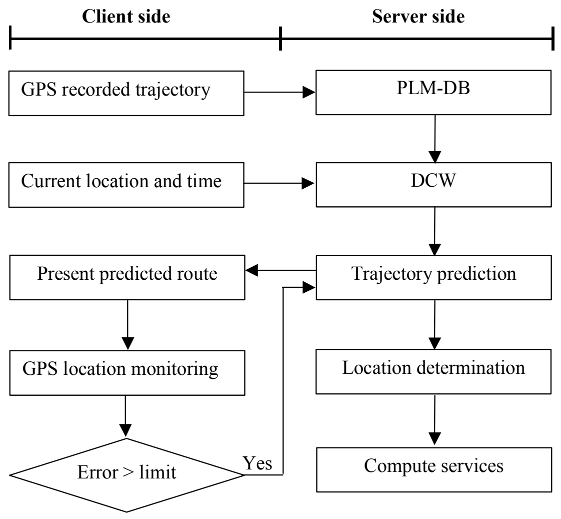

Figure 9. Steps of PLM

4. MODEL ANALYSIS

The probability matrix at each vetex (intersection) is updated periodically based on either user-specific or group trajectories. The quality of the probability matrix direclty determines the PLM's ability to detect regular trajectories. When a user follows a regular trajectory, certain conditional probability parameters in the probability matrix of certain intersections and the probabilities of trajectories will change.

Let us revisit the DCW and the network given in Figure 5. Assume the user follows a regular travel trajectory $e_1 \to e_8 \to e_9$ for twenty times after the matrix initialization, then both intersections v_2 and v_6 will have an updated probability matrix.

For v_2, the updated matrix becomes:

$$M(v_2, t_k) = \begin{bmatrix} 0 & 1/23 & 21/23 & 1/23 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix} \quad (10)$$

For v_6, the updated matrix becomes:

$$M(v_6, t_k) = \begin{bmatrix} 0 & 1/23 & 1/23 & 21/23 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix} \quad (11)$$

By reading the probability parameter $P(e_2|e_1)$ for v_2, we get $P(e_1, e_2) = P(e_2|e_1) \cdot P(e_1) = 1/23 = 0.0435$. Similarly, $P(e_1,e_8) = 21/23 = 0.9130$, $P(e_1,e_7) = 1/23 = 0.0435$. By reading the probability parameter $P(e_4|e_8)$ for v_6, we get $P(e_1,e_8,e_4) = P(e_1) \cdot P(e_8|e_1) \cdot P(e_4|e_1,e_8) = 1\cdot (21/23) \cdot (1/23) = 0.0397$ (note that $P(e_4|e_1,e_8) = P(e_4|e_8)$). Similarly, $P(e_1,e_8,e_5) = 1\cdot (21/23) \cdot (1/23) = 0.0397$ and $P(e_1,e_8,e_9) = 1 \cdot (21/23) \cdot (21/23) = 0.8336$. With these probabilities, the regular trajectory $e_1 \rightarrow e_8 \rightarrow e_9$ (shown by the arrows in Figure 10) is detected which has a much higher probability than any other trajectory.

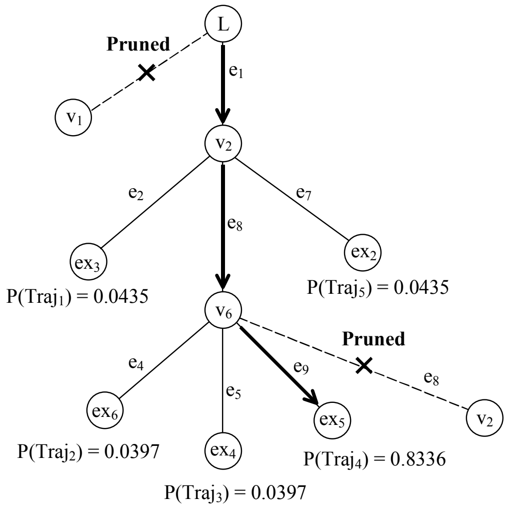

Figure 10. Regular trajectory detection

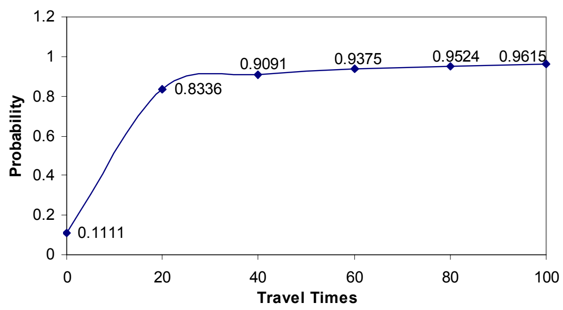

Figure 11. Probability growth of a regular trajectory

Using the same example, Figure 11 shows that the more the regular trajectory is repeated, the more stable is its probability. When the regular trajectory is traveled twenty times, its probability will be over 80%, and when the regular trajectory is traveled forty times, its probability will be above 90%.

5. CONCLUSIONS

Location prediction plays an important role in LBSs, in particular it can be used to improve performance and provide desired services. In this paper, we discussed two basic approaches towards location prediction and introduced a new prediction model for LBSs. We argue that map-based location prediction approach, such as the model presented in this work, has several benefits for those LBSs whose objective is to compute and present on-road services:

1) Reliability. The map-based approach supports road-level granularity and therefore improves the precision of location prediction in LBSs;

2) Computing resources. If future locations are predicted well in advance, there will be ample time for planning computing resources, especially for tasks demanding high-processing speed and/or large storage capacities;

3) Desired services. Location prediction facilitates the possibility of providing desired services by preparing and confirming them well in advance.

The proposed model has been analyzed only with synthetic data. We have yet to analyze PLM with real road networks and trajectories to evaluate its performance.

6. REFERENCES

[1] D.H. Stojanovic and S.J. Djordjevic-Kajan. Developing location-based services from a GIS perspective. In *5th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service (TELSIKS 2001),* vol. 2, pp. 459 – 462, 2001.

[2] N. Davis, K.Cheverst, K,Mitchell, and A.Friday. Caches in the air: Disseminating information in the guide system. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, New Orleans, USA, February 1999.

[3] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of developing and deploying a context-aware tourist guide. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, August 2000.

[4] R. Jana, T. Johnson, S. Muthukrishnan, and A. Vitaletti. Location based services in a wireless WAN using cellular digital packet data (CDPD). In *Second ACM international workshop on Data engineering for wireless and mobile access*, May 2001.

[5] H.D. Chon, D. Agrawal, and A.E. Abbadi. NAPA: Nearest Available Parking Lot Application. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, 2002.

[6] P. Bahl, A. Balachandran, A. Miu, G. Voelker, W. Russell, and Y. Wang. PAWNs: Satisfying the Need for Ubiquitous

Connectivity and Location Services. *IEEE Personal Communications Magazine (PCS)*, Vol. 6, October 2001.

[7] A. T. Campbell, J. Gomez, S. Kim, C.Wan. Comparison of IP Micro-Mobility Protocols. *IEEE Wireless Communications Magazine*, Vol. 9, No. 1, February 2002.

[8] S. K. Das and S. K. Sen. Adaptive Location Prediction Strategies Based on a Hierarchical Network Model in a Cellular Mobile Environment. *The Computer Journal*, Vol. 42, No.6, 1999.

[9] B.P. Vijay Kumar and P. Venkataram. Prediction-based location management using multilayer neural networks. *Journal of Indian institute of science*, pp.7-21, 2002.

[10] J. Biesterfeld, E. Ennigrou, and K. Jobmann. Location Prediction in Mobile Networks with Neural Networks. In *Proc. of the International Workshop on Applications of Neural Networks to Telecommunications '97*, S. 207-214, Melbourne, June 1997.

[11] S. H. Shah, and K. Nahrstedt. Predictive Location-Based QoS Routing in Mobile Ad Hoc Networks. In *Proceedings of IEEE International Conference on Communications (ICC 2002)*, New York, NY, April 2002.

[12] U. Kubach and K. Rothermel. An Adaptive, Location-Aware Hoarding Mechanism. In *Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, pp. 615-620, Antibes, France, July 2000.

[13] U. Kubach. A Map-Based, Context-Aware Hoarding Mechanism. Berichtskolloquium des Graduiertenkollegs Parallele und Verteilte Systeme, University of Stuttgart, Germany, July 2000.

[14] Y. Zhao. Mobile Phone Location Determination and Its Impact on Intelligent Transportation Systems. *IEEE Transaction on Intelligent Transportation Systems*, Vol. 1, No.1, March 2000.

[15] H. Gowrisankar, and S. Nittel. Reducing Uncertainty in Location Prediction of Moving Objects in Road Networks. In *2nd Int. Conference on Geographic Information Science (GIScience 2002)*, Boulder, Colorado, September 2002.

[16] O. Wolfson. The Opportunities and Challenges of Location Information Management. In *Intersections of Geospatial Information and Information Technology Workshop*, 2001.

[17] Tranplan Associates. Waterloo Region Travel Survey 1987: An Overview of the Survey Findings. Regional Municipality of Waterloo, Department of Planning and Development, October 1989.

[18] T. Tugcu, and C. Ersoy. Application of a Realistic Mobility Model to Call Admissions in DS-CDMA Cellular Systems. In *Vehicular Technology Conference (VTC'2001),* spring, Rhodes, Greece, May 2001.

[19] S. Schönfelder. Some notes on space, location and travel behaviour. In *Swiss Transport Research Conference*, Monte Verita, Ascona, 2001.

[20] J. Scourias and T. Kunz. An Activity-based Mobility Model and Location Management Simulation Framework. In *Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM'99)*, Seattle, USA, August 1999.

[21] White Paper: What is 3G? http://www.genericsgroup.com/what/consultancy/whatis3G.pdf