2013 IEEE 14th International Conference on Mobile Data Management

Distant-Time Location Prediction in Low-Sampling-Rate Trajectories

Meng-Fen Chiang[1,2], Wen-Yuan Zhu[1], Wen-Chih Peng[1] and Philip S. Yu[3]
[1]Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
[2]Yahoo! Communities Engineering Function, Nangang Science Park III, Taipei, Taiwan
[3]Department of Computer Science, University of Illinois at Chicago, Chicago, Illinois, USA
mfchiang@yahoo-inc.com, {wyzhu, wcpeng}@cs.nctu.edu.tw, psyu@cs.uic.edu

***Abstract*—With the growth of location-based services and social services, low-sampling-rate trajectories from check-in data or photos with geo-tag information becomes ubiquitous. In general, most detailed moving information in low-sampling-rate trajectories are lost. Prior works have elaborated on distant-time location prediction in high-sampling-rate trajectories. However, existing prediction models are pattern-based and thus not applicable due to the sparsity of data points in low-sampling-rate trajectories. For example, it becomes difficult to derive trajectory patterns, let alone utilizing trajectory patterns for distant-time location prediction. In this paper, given a query time, the current location and time, we aim to predict the location of an object at the query time. To address the sparsity in low-sampling-rate trajectories, we develop a Reachability-based prediction model on Time-constrained Mobility Graph (abbreviated as RTMG) to predict locations for distant-time queries. Specifically, we design an adaptive temporal exploration approach to extract effective supporting trajectories that are temporally close to the query time. These data points are then represented as a Time-constrained user mobility Graph (refers to as TG). In light of TG, we further derive the reachability probabilities among locations in TG. Thus, a location with maximum reachability from the current location among all possible locations in supporting trajectories is considered as the prediction result. To efficiently process queries, we proposed an index structure SOIT to organize location records for on-line query processing. We conduct extensive experiments on real low-sampling-rate datasets and demonstrate the effectiveness and efficiency of RTMG.**

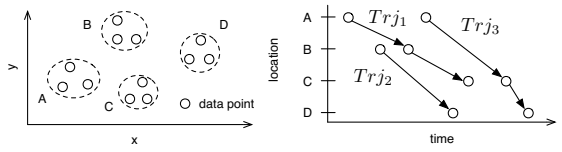
I. Introduction

With the growth of location-aware technologies and location based Internet services (e.g., Foursquare and Places on Facebook), tracking or collecting a huge amount of trajectories of users becomes feasible. Given a set of trajectories, prior works have studied the location prediction problem in which given the user's current location, the problem is to predict the next location or the location at a specific time. In particular, prior work in [5] formulates a distant-time query, where given a query time, the current location and time, one should estimate the location of objects at the query time. The distant-time query is very useful in many applications, such as pre-fetching users' future locations and delivering coupons, inferring the crowd of a region for tourism recommendations, and estimating the traffic status for transportation management [16].

Without loss of generality, a trajectory is considered as the movements of users and is expressed as a sequence of data points indicating user location information and the corresponding time. The trajectories considered in distant-time query [5] are high-sampling-rate trajectories that reveal detailed movements of users. Same as in [5], [8], [6], the next location prediction is based on the high-sampling-rate trajectories. As users could easilyl perform check-in services (e.g., Foursquare) to note their locations with a mobile phone, low-sampling-rate trajectories becomes ubiquitous. The time-ordered check-in records of a user are able to be expressed by trajectories. Moreover, on a photo sharing website (e.g., Flickr), people share geo-tagged photos whose time-stamps and geo-locations can be represented as trajectories as well. Clearly, low-sampling-rate trajectories are substantially different from high-sampling-rate trajectories, where details of movement information is generally lost. For example, time intervals between consecutive data points in low-sampling-rate trajectories may range from seconds to days [18]. In this paper, we aim to address the sparsity issue for distant-time query in low-sampling-rate trajectories.

A considerable amount of efforts has been devoted to inventing location prediction models in high-sampling-rate trajectories [10], [1], [17], [7]. Prior works assume that recent locations and velocities are available and employ a (non)linear movement model to determine near-future location mainly based on recent location and velocities. On the other hand, more complex location prediction models have also been studied in [5], [8], [6] to support distant-time queries. We mention in passing that the authors in [8] proposed a location prediction model, which infers next location of a user based on collective frequent patterns discovered from previous trajectories of all users. In [6], a path prediction model is proposed, which predicts the travel path of an object up to a specified time interval in the future. A hybrid prediction model (HPM) is proposed in [5], which is able to answer distant-time future locations. HPM relies on frequent moving patterns discovered from past trajectories as well as existing motion functions using the objects recent movements to support future location queries. While pattern-based prediction models over high-sampling-rate trajectory databases show promising query results, it may fail to effectively predict distant-time location queries over low-sampling-rate trajectories because HPM [5] may hardly mine any frequent moving patterns from a single user's low-sampling-rate trajectory. For example, suppose the

978-0-7695-4973-6/13 $26.00 © 2013 IEEE
DOI 10.1109/MDM.2013.22



(a) Dense regions (b) Low-sampling-rate trajectories

Fig. 1. Dense regions and low-sampling-rate trajectories

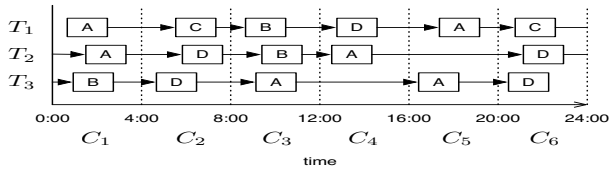


Fig. 2. Trajectory database

minimum threshold is two, only one frequent pattern, $A \rightarrow C$, that appears twice can be found from Trj_1, Trj_2 and Trj_3 as shown in Figure 1(b). In this case, HPM [5] always returns region C as the query result for any queries no matter where the user is currently located and what the query time is since only one pattern nearby the query time can be used. However, the time points of data points located at region C are far away from query time for some queries. Making prediction based on limited frequent moving patterns derived from HPM is unreliable and results in poor prediction coverage for distant-time location queries over low-sampling-rate trajectories. Therefore, the sparsity feature in low-sampling-rate trajectories calls for a new design for distant-time location prediction.

Most predictive techniques are mainly designed for high-sampling-rate trajectories based on recent movements or moving patterns of moving objects. However, future location over low-sampling-rate trajectories for a particular user is worth investigating as it can drive innovative applications for existing location-based services. For example, if we can effectively infer user's future location at a specified time point, time-sensitive information delivery (e.g., coupons) can intelligently incorporate user's future location by providing information that is valid nearby the predicted location at the specified time point. In this paper we formulate the problem of distant-time future location prediction over low-sampling-rate trajectories. Given current location at the current time point and a query time, we aim to predict the location of a user at query time. To address the sparsity issue due to low sampling rate, by leveraging various historical low-sampling-rate trajectories, we propose a Reachability-based prediction model on Time-constrained Mobility Graph (abbreviated as RTMG) that investigates user's reachability and determines the possible candidate locations. More specifically, first a spatial-temporal expansion is utilized to enlarge an investigation scope for retrieving some historical trajectories that are likely to infer locations of objects at the specified query time. For example, the data points that are spatially close are merged into regions of larger granularity as shown in Figure 1(a). Moreover, by expanding investigation time interval between current time and query time, we can infer paths from region A to other regions within the investigation time interval. These trajectories are called supporting trajectories. Based on the supporting trajectories, a Time-constrained mobility Graph (abbreviated as TG) is constructed. In light of TG, we derive reachability probabilities of vertexes (i.e., the locations) in TG and thus determine the most likely location at the query time. To improve efficiency of query processing, we also design an index structure, Sorted Interval-Tree (SOIT), to structure user mobilities according to their time locality into a data-centric balanced tree. Several operators are defined for SOIT to support data-centric spatial-temporal expansion and retrieval, i.e., automatically expanding a spatial-temporal scope according to the data distribution over time. With SOIT and its operators, we can efficiently retrieve supporting trajectories and infer time-constrained mobility network on-the-fly.

The contributions of this paper are summarized as follows:

- To address the sparsity in low-sampling-rate trajectories, we propose adaptive temporal exploration approach to retrieve supporting trajectories. These supporting trajectories are further formed into a time-constrained mobility graph TG.
- We design a Reachability-based prediction model RTMG for distant-time location prediction.
- We develop an index structure, SOIT, to facilitate data-centric reprieval and efficient query processing.
- We conduct extensive experiments on real data to evaluate our proposed framework.

The remainder of this paper is organized as follows. Section II gives the preliminary of our work. Section III presents a reachability-based prediction model for distant-time location queries. Section IV presents the index structure and operators to organize user mobilities to support dynamic construction of time-constrained mobility graph. Section V presents the performance study on RMTG and sensitivity analysis of RMTG. Related works are presented in Section VI. This paper concludes with Section VII.

II. Preliminary

In this section, we first define the data model and then formulate the problem of distant-time location prediction over low-sampling-rate trajectories.

A trajectory of a user is a sequence of location records ordered by the time stamps of location records. Note that the location records discussed in this paper could be the check-in records, GPS data points or geo-photos. Same as in [19][3], if the location information in trajectories is represented as a GPS coordinates, trajectories will be transformed into a sequence of regions, where regions are determined by existing clustering algorithms, such as OPTICS [2]. Specifically, given a set of GPS data points, we will utilize clustering algorithms to derive a set of regions. Furthermore, for some privacy concerns or the feature of social medias, the check-in records only indicate regions of users. Therefore, the location record considered in this paper is defined below:

Definition (Location Record) A location record g is a three-tuple $\langle u; l; t \rangle$, where u is the user ID, l is a location referring to a region that u stays at timestamp t.

Users may present periodic movement behavior hidden in their trajectories (e.g., daily movement behavior). Same as in HPM [5], a trajectory of a user is thus decomposed into smaller trajectories. Thus, we could decompose a trajectory of a user in daily-scale. Generally, a trajectory could be divided into sub-trajectories according to a time length determined by existing periodic methods or specified by users. These sub-trajectories are thus called decomposed trajectories. The definition of decomposed trajectories is defined as follows:

Definition (Decomposed Trajectory) Given a time length d (e.g., a day), a sequence of n location records is segmented into a set of decomposed trajectories of equal time unit $\mathbb{T}^d$=$\{T_1, ..., T_i\}$ according to their time-stamps. Each sub-trajectory T_i is a sequence of location records, where all location records are located within the same time unit d (e.g., occurred in the same day). Formally, a decomposed trajectory $T_i = \langle g_{i,1} \rightarrow \cdots \rightarrow g_{i,j} \rightarrow \cdots \rightarrow g_{i,m} \rangle$, where $1 < j \leq m$.

Figure 2 illustrates an example of decomposed trajectories in daily scale and the location information of records is represented by region identifications. As shown in Figure 2, trajectory T_1 has six location records $\langle g_{1,1}, ..., g_{1,6} \rangle$.

To better capture the spatial and temporal correlations hidden in decomposed trajectories, we present decomposed trajectories as a sequence of trajectory snapshots. To discover regions, we extract location records from decomposed trajectories and represent the location records as a spatial proximity matrix $\mathbb{M}$, which is a symmetric matrix with each entry $\mathbb{M}_{i,j}$ indicating spatial distance between location records i and j. As such, a set of decomposed trajectories is therefore transformed into a sequence of trajectory snapshots with segmentation in temporal dimension. The trajectory snapshot is defined below:

Definition (Trajectory Snapshot) Given a decomposed trajectory database $\mathbb{T}^d$ and a time cell size δ_t, a sequence of trajectory snapshots $\mathbb{C}(\delta_t, \mathcal{L})$=$\{C_1, ..., C_n\}$ is obtained by partitioning the trajectory database in temporal dimension into n time cells of equal size and transforming locations into regions $\mathcal{L}$ discovered from original location records.

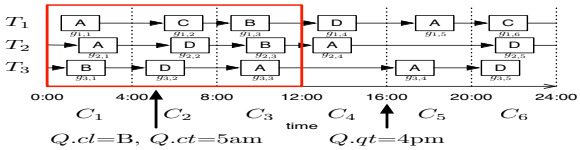
Figure 2 shows an example of trajectory snapshot with time cell size four hours. For example, we can find two regions C and D in snapshot C_2.

In this paper, we focus on location prediction over low-sampling-rate trajectories. Usually, the location records in decomposed trajectories are sparse in terms of the spatial and temporal domains. Therefore, we utilize the trajectory snapshot concept to gather information. The distant-time location prediction problem is defined as follows:

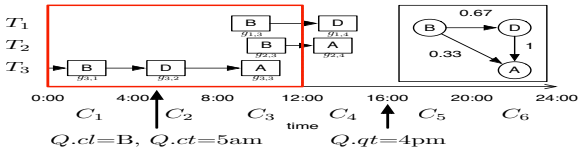
Problem (Distant-Time Location Prediction) Given a user's current location $Q.cl$, current time $Q.ct$ and query time $Q.qt$ ($Q.ct < Q.qt$), we aim at predicting the possible location that the user is likely to stay at query time $Q.qt$.

III. Algorithm of Location Prediction

In this section, we propose a Reachability-based prediction model on Time-constrained Mobility Graph, RTMG,



(a) Static temporal exploration



(b) Supporting trajectories

Fig. 3. Static temporal exploration with $k = 1$ and time-constrained mobility graph.

for distant-time location prediction. First, we will extract trajectories to capture user movement behaviors with regard to a given query. These trajectories are sub-trajectories of decomposed trajectories, naming *supporting trajectories* to facilitate the presentation throughout this paper. Then, in light of supporting trajectories, we model user behaviors as a Time-constrained mobility Graph (TG) and propose a reachability-based prediction algorithm to predict user locations.

A. Extracting Supporting Trajectories

Given a query at location $Q.cl$ at current time $Q.ct$, we first determine the set of supporting trajectories that may indicate user location at the given query time $Q.qt$. Given a set of decomposed trajectories $\mathbb{T}^d$, we extract supporting trajectories that satisfy the following two constraints:

- *spatial constraint*: the trajectory should initiate at location $Q.cl$,
- *temporal constraint*: the time interval between the start time and the end time of the trajectory should be no greater than $h = Q.qt - Q.ct$.

The set of supporting trajectories is denoted as $S = \{S_{p,q,r} \mid 0 < p \leq |\mathbb{T}^d|\}$, where each supporting trajectory $S_{p,q,r}$ is a sub-trajectory of a decomposed trajectory T_p by only extracting location records from the q^{th} to r^{th} location records in T_p.

In the beginning, we check whether there is any trajectory containing location record at $Q.cl$ in the snapshot containing $Q.ct$. If there is no any trajectory available, we extend the time information to include more location records. For example, in Figure 3(a), no trajectory satisfy the spatial ($Q.cl = B$) and temporal ($Q.ct$=5am) constraint in the second snapshot. Thus, we enlarge the time interval to include potential location records. We propose two temporal exploration approaches: (1) static temporal exploration and (2) adaptive temporal exploration. The static temporal exploration refers to collecting location records in a static time interval for all queries, whereas the adaptive temporal exploration dynamically adjusts time intervals according to different $Q.ct$.

1) Static Temporal Exploration: Let $\alpha(Q.ct)$ is the snapshot that contains the time point $Q.ct$. Given current location $Q.cl$, current time $Q.ct$ and a static exploration time interval k, we first extract supporting trajectories with their starting

location record at $Q.cl$ during the time interval $C_{\alpha(Q.ct)-k} \leq t \leq C_{\alpha(Q.ct)+k}$. Note that the range of static exploration time interval k is $0 \leq k \leq \frac{n}{2}$ and n is the number of snapshots. Essentially, $k = 0$ refers to no exploration and $k = \frac{n}{2}$ refers to full exploration centered from $\alpha(Q.ct)$ up to the snapshot that is $\frac{n}{2}$ away on both sides (i.e., the entire timeline).

Figure 3(a) illustrates the idea of static temporal exploration with $k = 1$. The static time interval exploration become the three trajectory snapshots. Then, we identify the ending snapshot according to $Q.qt$. In this example, the fourth trajectory snapshot is thus the ending snapshot. Therefore, from the decomposed trajectories, we extract sub-trajectories with their starting location record at B in the static time interval and the ending location records falling into the ending snapshot. As a result, we derive three supporting trajectories as shown in Figure 3(b). An example of supporting trajectories $S_{3,1,3}$=$B \rightarrow D \rightarrow A$ is the sub-trajectory starting from the first location record to the third location record in trajectory T_3.

Note that the setting of time intervals for static temporal exploration is very critical. With a large time interval, we can collect more supporting trajectories with more location records. However, these location records may not be relevant to the query (i.e., far away from the current time and the query time). On the other hand, if the time interval is too small, the amount of supporting trajectories may not be sufficient for prediction. Furthermore, depending the query time $Q.ct$, the exploration should result in different time intervals owing to the intrinsic distribution of location records. Thus, the desired time interval should be adjusted adaptively for different queries according to the distribution of location records over the timeline. As a result, we further propose an adaptive temporal exploration to dynamically determine a time interval.

2) Adaptive Temporal Exploration: Adaptive temporal exploration aims to dynamically determine the time interval for a query based on the temporal correlation between the query and current set of supporting trajectories. We invoke adaptive temporal exploration if we do no have sufficient and high quality supporting trajectories to develops the prediction model for a given query. Specifically, we broaden the time interval with the guidance of temporal correlation between the query and current set of supporting trajectories. Otherwise, we accomplish the extraction of supporting trajectories in the desired time interval. If the entire timeline is investigated, essentially the whole set of trajectories is used to provide more information, and thus may be more useful.

To measure the temporal correlation between the query and the current set of supporting trajectories, we first define *Local Dependency* for a region L_i w.r.t. a snapshot I_j as follows:

$$Dep_j^{L_i} = p_j^{L_i}(1 - H^{L_i}) \tag{1}$$

where p^{L_i} represents the probability that a region L_i appears in snapshot I_j and H^{L_i} is the Shannon's Entropy

$$H^{L_i}(p_1^{L_i}, p_2^{L_i}, ..., p_n^{L_i}) = -\sum_{j=1}^{n} p^{L_i} \log p_j^{L_i} \tag{2}$$

indicating the temporal dependency of a region L_i in snapshot I_j. The value of $Dep_j^{L_i}$ ranges between 0 and 1. Intuitively, a region L_i that appears exclusively in snapshot I_j (i.e., high temporal dependency in snapshot I_j) results in $Dep_j^{L_i} = 1$.

As a result, the local dependency for all regions in snapshot I_j is as follows:

$$Dep_j^{\mathcal{L}} = \sum_{i=1}^{|\mathcal{L}|} Dep_j^{L_i}. \tag{3}$$

We claim that a time interval is sufficient for exploring supporting trajectories if the local dependency of the time interval is higher than the expected value of local dependency under the assumption that regions are uniformly distributed over the entire timeline. Let $E[H]$ be the expected value of Shannon's Entropy. The expected value of location dependency in any snapshot should be as follows:

$$E[Dep_j^{\mathcal{L}}] = \frac{|\mathcal{L}|(1 - E[H])}{n} \tag{4}$$

where n is the number of snapshots and $|\mathcal{L}|$ is the number of regions.

As a result, given a time interval $C_{\alpha(Q.ct)-k} \leq t \leq C_{\alpha(Q.ct)+k}$ (i.e., k time cells away from the snapshot that contains the current time point $Q.ct$), we can determine the quality of the time interval based on the distribution of location records over the timeline as follows:

$$\min_{0 \leq k \leq \frac{n}{2}} \frac{Dep_t^{\mathcal{L}}}{(2k+1)E[Dep_j^{\mathcal{L}}]} \geq 1. \tag{5}$$

That is, we initiate temporal exploration from $k = 0$ (no exploration) up to $k = \frac{n}{2}$ (full exploration) and terminate the exploration process if equation (5) holds. When the entire timeline is investigated (i.e., full exploration with $k = \frac{n}{2}$), essentially the entire set of trajectories will be considered as the set of supporting trajectories.

B. Time-Constrained Mobility Graph

Inspired from the previous work [6], we model user mobility behavior as a Time-constrained mobility Graph, TG. TG is represented as a directed weighted graph $\mathbb{TG}^Q$=(V,E,W), where each node $v \in V$ represents a location and each edge e(u,v) represents a transition from location u to location v weighted by transition frequencies denoted w(u,v). TG is built from the set of supporting trajectories. Explicitly, for each supporting trajectory $S_{i,j,k}$, the set of unique locations in $S_{i,j,k}$ forms V. A path from the location associated with location record j to the location associated with location record k is created and the transition probability associated with an edge $e(u, v)$ is updated accordingly. Consequently, from the set of supporting trajectories, TG is able to capture movement behaviors during the time interval [$Q.ct$, $Q.qt$].

Figure 3(b) illustrates the time-constrained mobility graph that is constructed based on the supporting trajectories after static temporal exploration with $k = 1$. In total, three nodes are created for location A, B and D. For the supporting trajectory $S_{2,3,4}$=$B \rightarrow A$, the edge e(B,A) is created with transition frequency from B to A once and similarly an edge is created e(B,D) for $S_{1,3,4}$ with transition frequency once. Therefore, we could derive transition probabilities from B to D and from B to A as 0.67 and 0.33, respectively.

C. Location Prediction

Most previous studies evaluated the possibility of a candidate location merely according to mobility statistics such as immediate transition probabilities of moving patterns [5] or the traveling probability of a path [6]. For example, given a candidate path P: $v_1 \rightarrow ... \rightarrow v_k$ up to a prediction length, MaxLike in [6] returned the path v_k as the predicted answer if the travel probability of the path P is maximized among all possible paths between v_1 and v_k.

The mobility statistics collected from low-sampling-rate trajectories are very sparse and making prediction merely based on sparse mobility statistics of a single transition or a single path derived from a mobility graph may bias the prediction results. In addition, rather than probabilities of single immediate transition or single path, the probability of connectivity between node pairs is an important indicator of closeness of the node pair. Some node pairs that are located structurally close to each other in a time-constrained mobility graph and can be easily identified based on simple mobility statistics.However, some node pairs that are located far apart but they may be closely related based on connectivity if there are multiple paths connecting these two nodes on a mobility graph.

Consider one example of TG in Figure 3(b), where location B is structurally close to D with higher immediate transition probability than $w(B,A)$. However, location A has higher connectivity than D because there are multiple paths connecting B and A on the mobility graph. To incorporate both immediate transition frequency and connectivity in distant-time location prediction, we use the metric, reachability RCH, to estimate the probability that a user is located at each candidate region on a TG.

Definition (Reachability) Let A be the $|V| \times |V|$ transition probability matrix of a time-constrained mobility graph TG $\mathbb{G}^Q$. Given the restart probability $c \in (0,1)$, the reachability from $Q.cl$ to any node $v \in V$ is denoted as a vector $RCH_{Q.cl}$. $RCH_{Q.cl}$ can be derived by

$$RCH^k_{Q.cl} = cE_{Q.cl} + (1-c)RCH^{k-1}_{Q.cl}A \qquad (6)$$

when $RCH_{Q.cl}$ is converged, where $E_{Q.cl}$ is a vector, the entry representing $Q.cl$ is one and the rest entries are set to be zero.

Given a query Q and its TG G^Q, we propose to compute the reachability between $Q.cl$ and $v \in V$ in G^Q as a metric to predict the user's location at query time $Q.qt$.

IV. Index Structure

In this section, we present our indexing structure Sorted Interval-Tree (SOIT) in Section IV-A, including index construction and maintenance. Following this, we introduce a set of operators to support on-line predictive query processing with SOIT.

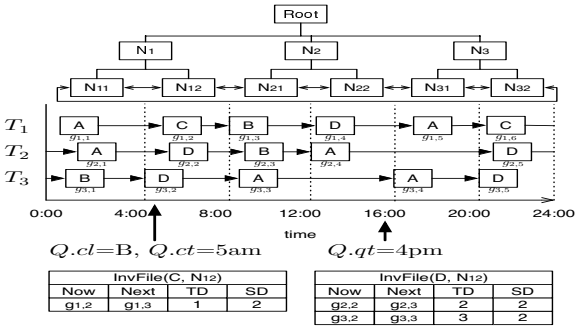


Fig. 4. Indexing scheme and query processing

A. Sorted Interval-Tree

1) Sorted interval-tree construction: The sorted interval-tree indexes a set of location records into a balanced tree such that each leaf time cell contains similar amount of data by partitioning a timeline into a sequence of time cells and maintaining a set of location records in each time cell no more than the size of b, where b is the branching factor. Figure 4 illustrates a set of location records indexed by SOIT with a branching factor of three. Centered at $Q.ct$=5am, the partitions that overlap with the time point is N_{12}, which consists of three location records, one locates at location C and the other two locate at location D.

Other time cells (varying time interval) are presented in shaded-color according to their temporal distance to N_{12} containing the current time $Q.ct$. For example, N_{11} and N_{21} are temporally closer to current time than N_{31} and thus colored in lighter grey.

Each leaf time cell of SOIT is associated with a group of inverted files, where each inverted file stores a group of location records with their time-stamps covered by the leaf time cell. Each record in an inverted file that is covered by time cell N contains four entries:

- Now: location record covered by N.
- Next: location record immediately after Now.
- TD: travel time between the end time of Now and the start time of Next.
- SD: total time that a user stayed during Next.

A leaf time cell is associated with a minimum bounding time interval of covered location records. All leaf time cells are sorted according their start time in ascending order and are connected into a list with sibling link for efficient query processing. Suppose $\{g^N_1, ..., g^N_m\}$ is the set of location records covered by a leaf time cell N. Then the minimum bounding time interval of N is a time interval I^N=$[t(g^N_1), t(g^{N+1}_1)]$, where $t(g^N_1)$ is the start time, $t(g^{N+1}_1)$ is the end time and $N+1$ is the next adjacent time cell in ascending order. In this way, the interval-tree is guaranteed to cover the entire timeline to ensures efficient query processing. For instance, SOIT in Figure 4 decomposes the timeline into a sequence of time cells (N_{11} to N_{32}) that completely cover the entire timeline in the order of the start time of minimum bounding time intervals. The scope of an intermediate time cell is recorded by the minimum bounding time interval, which is an

Algorithm 1: Sorted Insertion

Input: the root of current interval-tree, *Root*, and a location record to be inserted g;

Output: N: a time cell where g is placed;

```
1  N ← Root;
2  if N is a leaf then return N;
3  else
4      for e in N do
5          N' ← the time cell pointed by e;
6          if Overlap(N',g) then
7              N ← N';
8      return SortedInsertion(N, g);
```

extension of its children time cells. In this example, the time cell N_{12} is associated with two inverted files, $InvFile(C, N_{12})$ is specific to location C visited by a user during $I^{N_{12}}$ and $InvFile(D, N_{12})$ is specific to location D visited during $I^{N_{12}}$.

Retrieving the set of requested data points in the worst case only takes $O(b \log_b n)$ when a complete linear search is performed in the target leaf time cell, where b is the branching factor and n is the number of time cells. Complete coverage means that for any query point at least one time cell stored in the subtree is always guaranteed to contain the query point. This property guarantees that at most one traversal from the root to a leaf is enough to find requested location records of the time cell overlapped with a query point. The settings of minimum bounding time interval for each time cell ensure our SOIT without holes in the time cell coverage and thus the property holds in our index structure.

2) Proximity-aware maintenance: The set of location records can be flexibly grouped based on different criteria. In this study, in order to retrieve groups of time cells that are temporally close to a query point, we need to organize the groups of time cells by their start times.

SOIT facilitates efficient retrieval of time cells based on their temporal locality. To achieve this, we modify the general insertion procedure of building a balanced R-tree by grouping and ordering time intervals according to their start points. Algorithm 1 presents the idea of finding a time cell that an incoming location record should be inserted. If the time cell has enough space, then the location record is inserted to the time cell. Otherwise the time cell is split into two. Let g denote a location record to be inserted. If the minimum bounding time interval of an entry e in a time cell contains the start time of g, we can place g in entry e in ascending order of their start times. Then we follow the pointer of the current entry. Recursively, we continue this procedure until a leaf time cell is reached.

B. Query Processing

First, we define a set of operators for query processing using SOIT. Following this, we present a speed up version for query processing with the help of SOIT.

Overlap(N,g): Given a location record g and the time cell N at any level, if the time-stamp of g is contained in the minimum bounding time interval of the time cell N, then return true, otherwise return false.

Retrieve(t): Given a query point t, time cells overlapped with the query point are retrieved by performing a depth-first search through SOIT.

GetNxtTmCell(N,L): Given a query location L and a leaf time cell N, we follow the sibling link to obtain adjacent time cell $N+1$ ($N-1$) at its right(left) hand side, return $N+1$ ($N-1$) if it contains location records at the query place. Otherwise, cascade this operator until a valid time cell is found or return null if all time cells are scanned regarding query place L.

1) Static query processing: We first discuss how overlapped time cells retrieval facilitates query processing. Without the index structure, an intuitive approach to retrieve sub-trajectories for a query Q=(cl, ct, qt) is to scan the entire trajectory to select location records satisfying location $Q.cl$ and time constraints $Q.ct$ and extract subsequent location records up to the prediction length h, which takes $O(n)$ steps where n is total number of location records of the entire trajectory.

2) Dynamic query processing: With the index structure, two key techniques are designed to speed up query processing with the help of sorted interval-tree: (1)*overlapped time cells retrieval* and (2) *prediction length filtering*.

First, *Retrieve*($Q.ct$) operator is applied to select a leaf time cell overlapped with the query point $Q.ct$, which takes $O(b \log_b n)$. Second, among the groups of inverted files associated with the time cell, we scan these groups and select the inverted file that is associated with $Q.cl$, which in worse case takes $O(m)$ steps where m is the number of all possible places. Third, the location records stored in the qualified inverted file are the set of qualified location records and likewise subsequent location records for each qualified within the prediction length h are extracted. Because the third step is the same as the intuitive approach, we compare the time cost of these two approach simply based on the first two steps.

Theoretically, static query processing takes $O(n)$ steps to retrieve the set of location records that satisfy location and time constants, whereas dynamic query processing takes at most $O(b \log_b n)$ + $O(m)$ steps. Empirically, since the number of possible locations in a leaf time cell is generally much smaller than m and the branching factor b is simply a constant number, dynamic query processing can achieve significant pruning power with SOIT.

To apply the third step with our index structure, we present prediction length filtering to prune unnecessary examinations. Considering current location record g_i, *temporal distance* is defined to estimate the temporal distance between g_i and other candidate location record. Based on the temporal distance, prediction length filter is adopted to prune candidate location records that cannot be reached from g_i within a prediction length.

Definition (Temporal Distance) Given an observed transition from location record g_i to g_j, its temporal distance $D(g_i,g_j)$ is defined as:

$$D(g_i, g_j) = \begin{cases} g_i.SD + g_j.TD + g_j.SD & \text{if } g_i \to g_j \text{ exists} \\ \infty & \text{otherwise} \end{cases} \quad (7)$$

Algorithm 2: Speedup Query Processing

Input: A query Q=($Q.cl$,$Q.ct$,$Q.qt$) and the sorted interval-tree $SOIT$;

Output: $L(1:k)$: a ranked list of predicted locations;

1 $h \leftarrow Q.qt - Q.ct$;
2 $N \leftarrow$Retrieve($Q.ct$);
3 $Inv \leftarrow$Select Invfile($Q.cl$,$Q.ct$) of N;
4 **if** Inv *is empty* **then**
5 $N \leftarrow$ GetNxtTmCell(N,P);
6 $Inv \leftarrow$ Select $Invfile$($Q.cl$,$Q.ct$) of N;
7 $G \leftarrow$ DynamicMobilityGraph($Inv, h, SOIT$);
8 $L \leftarrow$ RWR(G);
9 **return** top-k ranked locations in L;

where $g_i.SD(g_j.SD)$ represents the estimated amount of time that a user spend at regions of $g_i(g_j)$ and $g_j.TD$ stands for the travel time that a user moved from the region of g_i to the region of g_j.

To apply temporal distance to prune candidate location records, the location record stored in inverted file is sorted by the sum of SD and TD. Then we define prediction length filter as follows:

Lemma (Prediction Length Filter) Given an observed transition from location record g_i to g_j, where g_i is covered by the leaf time cell N_p and g_j is covered by the leaf time cell N_q, if $D(g_i, g_j) > h$ then $D(g_i, g_k) > h$ holds for all g_k covered by N_{q+l}, where $1 \leq l$.

Proof: Because the time cells are resulted from partitioning a timeline and are arranged in ascending order of their start times, the start time of the time cell N_q always appears earlier than that of N_{q+l} in timeline. If $D(g_i, g_j)$ is greater than the prediction length h, any location record that appears after those covered by time cell N_q cannot be reached within h from g_i. ■

Algorithm 2 shows the pseudo-code of the the speedup version of query processing. The sorted interval-tree is pre-constructed. The Retrieve operator is first applied to select time cells that satisfy time constraint $Q.ct$ (Line 2). After that, the inverted file that satisfies location constraint $Q.cl$ is retrieved (Line 3). Then, we apply GetNxtTmCell operator to derive the adaptive time interval for query Q (Line 4-7). After that, a time-constrained mobility graph TG is built by algorithm 3 using SOIT (Line 8). Then, we apply Random Walk with Restart to estimate the reachability of each candidate location in TG. Note that in this study we utilize the basic power iteration approach to compute reachability, several existing techniques can be applied to further speed up reachability computation [12][4].

Algorithm 3 summarizes the idea of time-constrained mobility graph construction by dynamically probing the index structure to collect supporting trajectories. At the beginning, a collection of unchecked transitions are stored into $Queue$. For each unchecked transition r and remaining prediction length h, we first check whether its temporal distance $D(r.Src, r.Dst)$ is less than h (Line 4). If transition r is less than remaining prediction length h, we record r by updating the current time-constrained mobility graph $\mathbb{G}^Q$ (Line 5-9). We also update

Algorithm 3: Dynamic Mobility Graph

Input: an inverted file Inv, a prediction length h and the sorted interval-tree $SOIT$;

Output: $\mathbb{G}^Q = (V, E, W)$: a time-constrained mobility graph;

1 $Queue \leftarrow$ Push(Inv,h);
2 **while** $Queue$ *is not empty* **do**
3 (r,h)$\leftarrow$ Pop($Queue$);
4 **if** $D(r.Now, r.Next) \leq h$ **then**
// update graph
5 $P_{now} \leftarrow$ the location of r.Now;
6 $P_{next} \leftarrow$ the location of r.Next;
7 $V \leftarrow V \cup P_{now} \cup P_{next}$;
8 $E \leftarrow E \cup e(P_{now}, P_{next})$;
9 $w(P_{now}, P_{next})$++;
// check subsequent records
10 $t \leftarrow$ the time-stamp of r.Dst;
11 $h \leftarrow h - D(r.now, R.next) - r.SD$;
12 $N \leftarrow$ Retrieve(t);
13 $Inv \leftarrow$ Select $Invfile$(P_{next},t) of N;
14 $Queue \leftarrow$ Push((Inv, h));
15 **return** G;

check time point t and remaining prediction length h (Line 10-11). After that, we search next time cell satisfying time constraint $Q.ct$ and retrieve location records from the inverted file satisfying location constraint P_{dst} (Line 12-13). We insert the retrieved location records and updated prediction length into $Queue$ and repeat the entire process from Line 2 until no location records is available in $Queue$. Eventually, a time-constrained mobility graph $\mathbb{G}^Q$ for query Q based on supporting trajectories is established.

V. Performance Evaluation

In this section, we conducted experiments to evaluate the performance of our algorithm on real datasets.

A. Datasets and Settings

1) Datasets: In this paper, we use two real datasets to conduct extensive experimental results.

Nokia: The dataset contains 80 user trajectories, collected over several months by Nokia, and consists of 16,920 trajectories after decomposition into daily-scale (i.e., d set to a day). The GPS coordinates was first transformed into region symbol by Nokia to exclude GPS coordinates for privacy concern. Regions are user-specific and they are ordered by the time of visit. Each region corresponds to a circle with radius 100 meters. Eventually, we are provided with sequences of regions, where each visit lasts longer than 20 minutes and the average time interval between consecutive location records over 80 users is approximately 11 hours.

Gowalla: The dataset contains 50 users, 113 decomposed trajectories in daily-scale, and 30 distinct user-specific regions on average. The average time interval between consecutive location records is approximately 17 days. The regions were discovered by OPTICS [2] with ε set to be 100 meters and $MinPts$ set to be three.

We divided the dataset into two partitions: a training set containing earlier 75% consecutive location records in each trajectory and a testing set containing the rest 25% location

records in each trajectory. The training set is used for building prediction model and the testing set is used for performance evaluation. A query Q is formed by a pair of consecutive location records in the testing set, where the current location $Q.cl$ and current time $Q.ct$ are the region, the time-stamp of the first location record; $Q.qt$ is the time-stamp of the second record; and the ground-truth of this query is the region of the second record. In total there are 12,897 distinct queries.

2) Metrics: To evaluate the accuracy of distant-time location queries, the prediction accuracy is measured by *Hit Rate*. For each query Q, our model returned a ranked list of top-k locations. We define an outcome to be true positive if the ground-truth is contained in top-k locations. Given a collection of queries Q^u of user u, we define the hit rate *HR* of the query set Q^u as $HR(Q^u) = \frac{|TP^u|}{|Q^u|}$, where $|TP^u|$ is the number of true positive outcomes over Q^u for user u. Given a set of users U, we measure the prediction accuracy by average hit rate over the set of users U as follows:

$$HR(Q^U) = \frac{1}{|U|} \sum_{u \in U} \frac{|TP^u|}{|Q^u|} \tag{8}$$

We also evaluate the reliability of a prediction model by measuring the ratio of queries that a prediction model can return query results among all queries as follows:

$$Coverage(Q^U) = \frac{1}{|U|} \sum_{u \in U} \frac{|RTN^u|}{|Q^u|} \tag{9}$$

where $|RTN^u|$ is the amount of queries in Q^u that a model can response to. Intuitively, the more queries that a model can return query results, the higher coverage of a model.

3) Baselines: To evaluate the performance, we compare the proposed prediction model with the following two baselines.

Baseline(HPM): To evaluate the effectiveness of a distant-time prediction model, we compare proposed reachability-based prediction model (RTMG) with the existing distant-time prediction model (HPM [5]). Given a user's recent movements and a query time, HPM discovered frequent patterns from past trajectories and then combined with user's recent movements to support distant-time queries. The parameter, minimum confidence, is set to 0.3 and the maximum number of time-stamps for storing recent movements is set to 20.

Baseline(Frequency): As mentioned in [5], user's recent movement is less important compared to locations nearby the query time. Therefore, we collect and rank the set of locations nearby the query time by the number of occurrences for distant-time queries. In the experiment, we set the default time range to be three hours.

B. Performance Study

In this section, we evaluate the performance of proposed prediction model.

1) Evaluation of prediction quality: We compare the proposed prediction model with two baselines.

Effect of data sparsity: To study the prediction quality for low-sampling-rate trajectories, we derive trajectories of different data sparsity by re-sampling data points of trajectories with varying probabilities $p\%$. A re-sampled trajectory contains $p\%$ location records randomly chosen from an original trajectory. The number of location records of a re-sampled trajectory decreases as $p\%$ decreases. The same as the baseline Frequency, we set the time cell size δ_t to three hours. The top-k is set to be three for all approaches. Figure 5(a) shows the prediction accuracies on Nokia dataset, where the performances of each prediction model decrease while the data sparsity increases (i.e., smaller $p\%$). For example, adaptive RTMG achieves 54.4% hit rate on original trajectories ($p = 100\%$) and slightly decreases to 39.2% on re-sampled trajectories ($p = 20\%$). HPM only achieves 12% hit rate on original trajectories ($p = 100\%$) and decreases to less than 1% on re-sampled trajectories ($p = 20\%$) due to the limitation of moving patterns. Figure 5(b) shows that adaptive RTMG can handle more queries than HPM because of the adaptive temporal exploration in trajectory extraction while HPM fails to predict locations due to the lack of moving patterns. Clearly, not only does adaptive RTMG achieve five times higher accuracy than HPM, the introduced exploration approach also promises higher coverage of query processing.

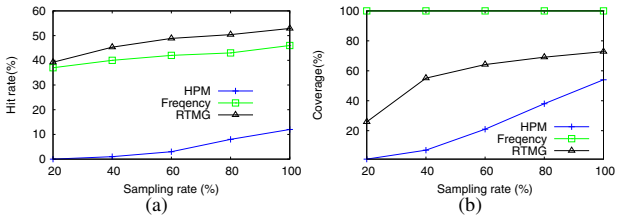
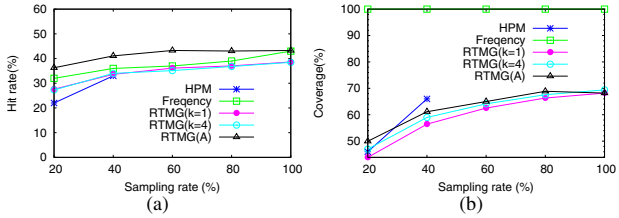


Fig. 5. Effect of different sampling rate

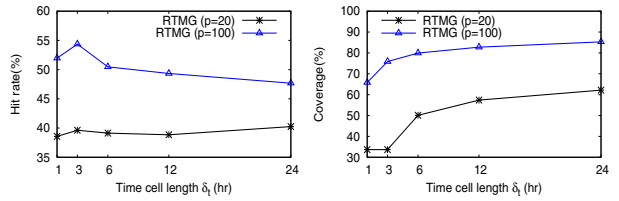
Effect of temporal exploration: To better understand the effectiveness of adaptive temporal exploration, we also study the prediction quality for both static and adaptive temporal exploration to baseline prediction models. Figure 6(a) shows that the prediction accuracies on Gowalla dataset for all prediction models, where RTMG (k=1) indicates no temporal exploration (time-sensitive) and RTMG (k=4) indicants full temporal exploration (time-irrelevant) for supporting trajectory extraction. At $p = 20\%$, RTMG(A) (i.e., adaptive RTMG) outperforms others with 36.3% hit rate on re-sampled trajectories. Particularly, RTMG (k=4) essentially extracts supporting trajectories regardless of time constraint and performs worse than RTMG (k=1) and adaptive RTMG (A). For example, RTMG (k=1) achieves 27.6% hit rage and RTMG (k=4) 27.2% respectively at $p = 20\%$. HPM only achieves 22% hit rate at $p = 20\%$ and failed to make further predictions within one day while the sampling rate decreases after $p = 40\%$ due to the intensive computation on association rule discovery.

Figure 6(b) shows that RTMG (A) can handle more queries at each re-sampling rate than others because the adaptive temporal exploration encourage RTMG to explore sufficient amount of supporting trajectories for different queries while others such as HPM fail to prediction locations due to the lack of moving patterns. In summary, adaptive temporal exploration can extract sufficient and effective supporting trajectories with



(a) (b)

Fig. 6. Effect of temporal exploration



(a) (b)

Fig. 7. Effect of different time cell size

the guidance of temporal dependency, leading to higher accuracy and higher coverage especially for distant-time location queries at extremely low-sampling-rate.

Effect of time cell: To investigate the effect of time cell sizes, we compare the hit rate and coverage of our proposal with varying size of time cells. Figure 7(a) shows that our proposed scheme achieves the best prediction accuracy (HR) when the size of time cell is three hours long. Intuitively, time cells of greater size can accumulate more moving behaviors and should be useful for distant-time location prediction as shown in Figure 7(a) with cell size no greater than three hours. Also, greater sizes of time cells facilitate effective retrieval for more queries and thus achieve higher coverage as shown in Figure 7(b). However, time cells of coarse granularity can only improve limited prediction accuracy because distant-time location queries are highly time-dependent and thus moving behaviors accumulated far away from an investigation time point become less useful.

2) Evaluation of prediction efficiency: We compare the query processing cost between static and dynamic approaches. Static approach refers to construct time-constrained mobility graph from scratch which requires a complete scan of trajectories to extract supporting trajectories for each query. Dynamic approach refers to processing queries using our index structure, which is constructed off-line.

The impact of index structure: Dynamic query processions costs less response time to build a time-constrained mobility graph compared to static approach because only a limited number of lookups are required. Figure 8 shows the amount of lookups required to build a time-constrained mobility graph. Clearly, the number of lookups are significantly reduced with the help of SOIT. For example, the number of lookups in static and dynamic approach is 506 and 18 respectively at prediction length being three (hr).

Evaluation of prediction scalability: To study scalability, we generate five trajectory datasets with different data sizes based on the original dataset. Each original trajectory trj is re-generated by slightly adjusting time-stamps of location records in trj. Under these settings, we obtain the largest

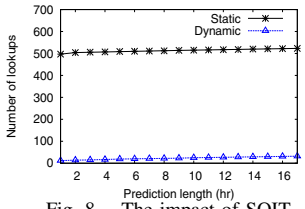


Fig. 8. The impact of SOIT

TABLE I
TRAJECTORY DATASETS DESCRIPTION

Dataset name	D1	D2	D3	D4	D5
Trajectories	12,680	18,467	27,519	38,508	50,546
Total points	38,694	46,367	61,758	84,953	115,924
Time interval(hr)	10.6	17.6	19.9	19.3	17.7

dataset $D5$ that consists of 50,546 trajectories with average time interval 17.7 hours between consecutive location records. Table I summarizes the details.

Figure 9(a) shows the the response time to construct TG graph with (Dynamic) and without (Static) SOIT over different sizes of dataset. As shown in 9(a), dynamic approach promises linear complexity on different dataset compared to static approach. For instance, the response time to construct TG by static approach is 4,247 (ms) and that by dynamic approach with index structure only takes 749 (ms) on dataset D5, which suggests that dynamic approach is several orders of magnitude faster on large dataset. Figure 9(b) plots the query response time in constructing SOIT and building TG graph with SOIT respectively. Clearly, the overhead to construct SOIT for a given trajectory is extremely small compared to total cost to construct TG graph for query processing. Therefore, it is reasonable to say that our index structure SOIT can efficiently speedup RTMG by reducing the time required in TG construction.

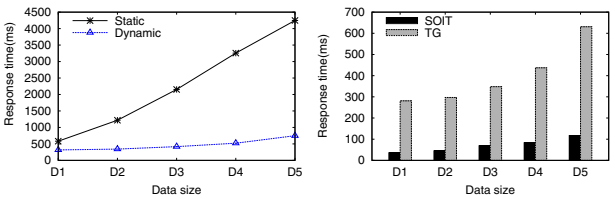
VI. Related Work

A. Index Structure for Moving Objects

Most existing solutions for moving object indexing assume a linear movement model given details of past moving information (e.g., near-past position and velocity). They can be further divided into object partitioning [9][11] and space partitioning solutions [17][7]. The indexing techniques share the same goal of maintaining spatial proximity for moving objects such that they can support efficient queries and updates. For example, the authors in [17] proposed a B^x-tree-based index structure to organize uncertain moving objects to support k-nearest neighbor query or range queries. The authors in [7] proposed the PEB-tree to integrates location proximity and policy compatibility by encoding both the location privacy compatibility and the spatial proximity among users in a one-dimensional value that is amenable to B+-tree indexing.

B. Location Prediction

A considerable effort has been devoted to designing location prediction models in high-sampling-rate trajectory databases. Most of them mainly focused on predicting locations in near future [5][8][6]. For example, Monreale et al. [8] proposed a location prediction model, which answers next location of



(a) (b)

Fig. 9. Scalability and index construction cost

a user relying on frequent patterns discovered from previous trajectories of a group of people. Jeung et al. [6] proposed a path prediction model to predict the travel path of an object up to a time duration in the future. They proposed the maximum likelihood algorithm to return paths that maximize the travel probability among all possible paths. Jeung et al. [5] proposed a hybrid prediction model (HPM) to manage relatively distant-time future locations. HPM relies on frequent patterns discovered from previous trajectories as well as existing motion functions using the object's recent movements to support future location queries. As low-sampling-rate trajectories can be easily accumulated from location-based services and social services, studying low-sampling-rate trajectories becomes important [14][15][13][18]. To the best of our knowledge, no attention has been paid to answer distant-time location queries in low-sampling-rate trajectory databases.

VII. Conclusion

To address the sparsity in low-sampling-rate trajectories, we develop a Reachability-based prediction model on Time-constrained Mobility Graph (RTMG) to predict locations for distant-time queries. Specifically, we design an adaptive temporal exploration approach to extract effective supporting trajectories that are temporally close to the query time. Based on the supporting trajectories, a Time-constrained mobility Graph (TG) is constructed to capture mobility information at the given query time. In light of TG, we further derive the reachability probabilities among locations in TG. Thus, a location with maximum reachability from the current location among all possible locations in supporting trajectories is considered as the prediction result. To efficiently process queries, we proposed the index structure SOIT to organize location records. Extensive experiments with real data demonstrated the effectiveness and efficiency of RTMG. First, RTMG with adaptive temporal exploration significantly outperforms the existing pattern-based prediction model HPM [5] over varying data sparsity in terms of higher accuracy and higher coverage. Also, the proposed index structure SOIT can efficiently speedup RTMG in large-scale trajectory dataset. In the future, we could extend RTMG by considering more factors (e.g., staying durations in locations, application usages in smart phones) to further improve the prediction accuracy.

Acknowledgment

Wen-Chih Peng was supported in part by the National Science Council, Project No. 100-2218-E-009-016-MY3 and 100-2218-E-009-013-MY3, by Taiwan MoE ATU Program, by Academic Sinica, Project No. AS-102-TP-A06, by D-Link and by HTC.

References

[1] C. Aggarwal and D. Agrawal, "On nearest neighbor indexing of nonlinear trajectories," in *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2003, pp. 252–259.

[2] M. Ankerst, M. Breunig, H. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," *ACM SIGMOD Record*, vol. 28, no. 2, pp. 49–60, 1999.

[3] X. Cao, G. Cong, and C. Jensen, "Mining significant semantic locations from gps data," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1009–1020, 2010.

[4] X. Cao, G. Cong, and C. S. Jensen, "Retrieving top-k prestige-based relevant spatial web objects," *Proceedings of VLDB Endowment*, vol. 3, pp. 373–384, September 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1920841.1920891

[5] H. Jeung, Q. Liu, H. Shen, and X. Zhou, "A hybrid prediction model for moving objects," in *Proceedings of the 24th ICDE International Conference on Data Engineering*. Ieee, 2008, pp. 70–79.

[6] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen, "Path prediction and predictive range querying in road network databases," *The VLDB Journal*, vol. 19, pp. 585–602, August 2010. [Online]. Available: http://dx.doi.org/10.1007/s00778-010-0181-y

[7] D. Lin, C. Jensen, R. Zhang, L. Xiao, and J. Lu, "A moving-object index for efficient query processing with peer-wise location privacy," *Proceedings of the VLDB Endowment*, vol. 5, no. 1, pp. 37–48, 2011.

[8] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "Wherenext: a location predictor on trajectory pattern mining," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2009, pp. 637–646.

[9] S. Šaltenis, C. Jensen, S. Leutenegger, and M. Lopez, *Indexing the positions of continuously moving objects*. ACM, 2000, vol. 29, no. 2.

[10] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu, "Prediction and indexing of moving objects with unknown motion patterns," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 2004, pp. 611–622.

[11] Y. Tao, D. Papadias, and J. Sun, "The tpr*-tree: an optimized spatio-temporal access method for predictive queries," in *Proceedings of the 29th International Conference on Very Large Data Bases*. VLDB Endowment, 2003, pp. 790–801.

[12] H. Tong, C. Faloutsos, and J. Pan, "Random walk with restart: fast solutions and applications," *Knowledge and Information Systems*, vol. 14, no. 3, pp. 327–346, 2008.

[13] M. Ye, P. Yin, W. Lee, and D. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *Proceedings of ACM SIGIR International Conference on Research and Development in Information Retrieval*, 2011.

[14] J. Ying, W. Lee, T. Weng, and V. Tseng, "Semantic trajectory mining for location prediction," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2011, pp. 34–43.

[15] J. Yuan, Y. Zheng, and X. Xie, "Discovering regions of different functions in a city using human mobility and pois," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012.

[16] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2010, pp. 99–108.

[17] M. Zhang, S. Chen, C. Jensen, B. Ooi, and Z. Zhang, "Effectively indexing uncertain moving objects for predictive queries," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 1198–1209, 2009.

[18] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *Proceedings of the 28th ICDE International Conference on Data Engineering*. ICDE, 2012, pp. 1144–1155.

[19] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 312–321.