Noname manuscript No.
(will be inserted by the editor)

Personalized Trajectory Matching in Spatial Networks

Shuo Shang · Ruogu Ding · Kai Zheng · Christian S. Jensen · Panos Kalnis · Xiaofang Zhou

Received: date / Accepted: date

Abstract With the increasing availability of moving-object tracking data, trajectory search and matching is increasingly important. We propose and investigate a novel problem called Personalized Trajectory Matching (PTM). In contrast to conventional trajectory similarity search by spatial distance only, PTM takes into account the significance of each sample point in a query trajectory. A PTM query takes a trajectory with user specified weights for each sample point in the trajectory as its argument. It returns the trajectory in an argument data set with the highest similarity to the query trajectory. We believe that this type of query may bring significant benefits to users in many popular applications such as route planning, carpooling, friend recommendation, traffic analysis, urban computing, and location based services in general.

PTM query processing faces two challeges: how to prune the search space during the query processing and how to schedule multiple so-called expansion centers effectively. To address these challenges, a novel two-phase search algorithm is proposed that carefully selects a set of expansion centers from the query trajectory and exploits upper and lower bounds to prune the search space in the spatial and temporal domains. An efficiency study reveals that the algorithm explores the minimum search space in both domains. Second, a heuristic search strategy based on priority ranking is developed to schedule the multiple expansion centers, which can further prune the search space and enhance the query efficiency. The performance of the PTM query is studied in extensive experiments based on real and synthetic trajectory data sets.

Keywords Personalized Trajectory Matching, Efficiency, Optimization, Spatial Networks, Spatio-Temporal Databases

Shuo Shang
Department of Computer Science, Aalborg University, Aalborg, Denmark
E-mail: sshang@cs.aau.dk

Ruogu Ding · Panos Kalnis
King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

Ruogu Ding
E-mail: ruogu.ding@kaust.edu.sa

Panos Kalnis
E-mail: panos.kalnis@kaust.edu.sa

Kai Zheng · Xiaofang Zhou
School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Australia

Kai Zheng
E-mail: kevinz@itee.uq.edu.au

Xiaofang Zhou
E-mail: zxf@itee.uq.edu.au

Christian S. Jensen
Department of Computer Science, Aarhus University, Aarhus, Denmark
E-mail: csj@cs.au.dk

1 Introduction

The continuous proliferation of mobile devices and the rapid development of positioning services such as GPS [31] enable people to log their current geographic locations and share their trajectories by means of services such as Bikely[1], GPS-Way-points[2], Share-My-

[1] http://www.bikely.com/
[2] http://www.gps-waypoints.net/

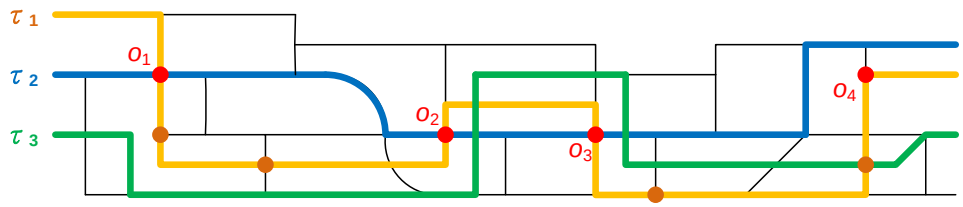

Fig. 1 An Example of Personalized Trajectory Matching in Spatial Networks

Routes[3], Microsoft GeoLife[4] [32]. Also, more and more social networking web sites, including Twitter[5], Foursquare[6], and Facebook[7] support the sharing of trajectories. The availability of massive trajectory data enables novel mobile applications. Such applications may utilize trajectory search and matching, which find trajectories that are similar in some specific sense to the query input (usually a trajectory). This type of query can benefit many services, including route planning, carpooling, friend recommendation, traffic analysis, urban computing and location based services in general. For example, tourists can check the travel histories of other tourists to improve their own travel; commuters can find carpooling partners; location based social networks can identify users with similar travel routes and recommend them as friends.

In most of existing studies (e.g., [1,6,7,11,17,18,30]), the query is specified in spatial terms only, which means that only spatial similarity is considered in trajectory matching. However, in real application scenarios, e.g., in recommender systems, spatial similarity is insufficient to evaluate the relationship between two different trajectories, due to the particular preference of users. Given a query trajectory q, existing studies treat every sample point $o \in q$ equally, no matter whether it is a petrol station, a transfer center, or a sightseeing location. As a result, a tourist service may recommend a travel route that misses intended sightseeing locations to a tourist; or a service may recommend an unsuitable carpooling partner (e.g., whose travel route does not pass by intended places) to a commuter. Although the recommended trajectories may be spatially similar to the query trajectory, it is possible that the users may not be fully satisfied with the recommendations, as their key preferences are not fulfilled.

Taking into account the weaknesses of existing trajectory matching approaches, we propose and investigate a novel problem named Personalized Trajectory Matching (PTM). On top of the conventional trajectory search based on spatial similarity only (e.g., [1, 6, 7, 11, 17, 18, 29, 30]), the PTM query also takes into account the significance of each sample point o in a query trajectory q. Initially, the system may assign a default weight to each sample point $o \in q$ to describe its significance; users are then allowed to adjust the weights according to their preference. As an example, when tourists search for travel routes or commuters seek carpooling opportunities, the weights of some places (e.g., sightseeing locations, transfer centers) should be higher than those of other sample points. To the best of our knowledge, this is the first work that investigates this more general trajectory matching problem in spatial networks while taking the significance of trajectory sample points into account. Previous studies (e.g., [25, 26]) have simply assumed trajectories of equal length (i.e., the same number of sample points) and assigned equal significance to each sample point. In contrast, we treat each trajectory as a sequence of weighted sample points with arbitrary length.

An example is demonstrated in Figure 1, where τ_1 is the query trajectory ($q = \tau_1$), while τ_2 and τ_3 are trajectories in data set T. At query time, the system assigns a default weight to each sample point $o \in \tau_1$. Further, o_1, o_2, o_3, and o_4 are the intended places specified by the user, and their weights are set to higher values than the default weight. During query processing, if only spatial similarity is considered, trajectory τ_3 will be returned, as it is closer and more similar to the query than τ_2. However, the user is unlikely to be satisfied with this result since τ_3 only contains one of the intended places and is not close to the others. Our approach takes into account the significance of each sample point $o \in \tau_1$, and τ_2 is a much better candidate for the query trajectory τ_1. Although τ_2 is not as good as τ_3 according to spatial similarity, it passes through or is close to all the intended places.

The proposed PTM query is applied in spatial networks, since in a large number of practical scenarios, objects move in such networks (e.g., roads, railways, rivers) rather than in a Euclidean space. A trajectory is a sequence of timestamped sample points of a moving object. We assume that all sample points have already been aligned to the vertices in the correspond-

[3] http://www.sharemyroutes.com/
[4] http://research.microsoft.com/en-us/projects/geolife/
[5] http://twitter.com/
[6] http://foursquare.com/
[7] http://www.facebook.com/

ing spatial network (spatial domain) according to some map-matching methods (e.g., [2, 3, 14, 20, 28]) and that between two adjacent sample points a and b, a moving object always follows the shortest path connecting a and b. The *timestamp*s of all trajectory sample points are mapped onto a time axis within the range of 24 hours (temporal domain). A straightforward approach to solving the PTM problem is called balanced search. At query time, each sample point $o \in q$ is considered as a query source (expansion center). The balanced search is conducted in the spatial and temporal domains concurrently and asks for the trajectories similar to the query trajectory q in the two domains. By combining the two domains' search results, the trajectory with the highest similarity to q can be derived. The main drawback of balanced search is search space overlap. Given two adjacent sample points (usually close to each other) that are both selected as query sources, their search spaces may overlap substantially. The trajectories in the overlapping region will be read and processed repeatedly, which yields unnecessarily poor performance. In addition, this approach lacks of an effective scheduling strategy for the multiple query sources with different weights, again leading to poor performance. To the best of our knowledge, there is no existing approach that can address PTM problem efficiently.

To improve on balanced search, a novel two-phase search algorithm is proposed. First, we carefully select a set of sample points O from the query trajectory q as query sources (expansion centers) and develop upper and lower bounds on the spatio-temporal similarity to q to constrain the search space in both domains. An efficiency study reveals that our method defines the minimum search space during the query processing. Second, a heuristic search strategy based on priority ranking is developed to schedule the multiple query sources effectively. Conceptually, we establish and maintain a dynamic priority ranking heap during the query processing. At each time, we only search the top-ranked query source until a new top-ranked query source appears. Compared to balanced search, the two-phase search algorithm has two major advantages: (i) it minimizes the search space in both the spatial and the temporal domains, and it avoids devoting unnecessary search efforts to overlapping regions; (ii) the effective heuristic search strategy focuses on trajectories more likely to be the optimal choice and further enhances the query efficiency.

To sum up, the main contributions are as follows:

- We define a novel trajectory matching query according to proposed trajectory similarity metrics. It provides new spatio-temporal features and holds the potential to benefit many popular mobile applications such as route planning, carpooling, and friend recommendation.
- We propose new metrics to evaluate trajectory similarity in the spatial and temporal domains.
- We develop an adaptive, two-phase search algorithm to compute the PTM query efficiently, with the support of upper and lower bounds and a heuristic scheduling strategy based on priority ranking. An efficiency study reveals that our solution defines the minimum search space during query processing.
- We conduct extensive experiments on real and synthetic trajectory data to investigate the performance of the proposed algorithms.

The rest of the paper is organized as follows. Section 2 presents related work and Section 3 introduces spatial networks, trajectories, and the spatio-temporal similarity metrics used in the paper and also gives the problem definition. A baseline method is introduced in Section 4, and the PTM query processing is detailed in Section 5. This paper is concluded in Section 7 after discussions on experimental results in Section 6.

2 Related Work

2.1 Trajectory Similarity Search

The problem of trajectory similarity search (e.g., [1, 4, 5, 10, 12, 30, 33]) has been studied extensively in the last two decades. The query processing involves two steps in general. First, a similarity function is defined based on the distances between the trajectory points that represent trajectories. Second, an algorithm is proposed to query a large trajectory data set. Several types of trajectory similarity functions have been proposed that target different applications, including Euclidean Distance [1], Dynamic Time Warping [30], Longest Common Subsequence [27], Edit Distance [8], Edit Distance with Real Penalty [6], and Edit Distance on Real Sequences [7]. Techniques for time series data similarity/approximation evaluation are also studied by Morse and Patel [21] and Skerkat and Rafiei [23].

On top of the conventional trajectory search based on spatial similarity only (e.g., [1, 6, 7, 11, 17, 18, 30]), Personalized Trajectory Matching (PTM) takes into account the significance of each sample point o in the query trajectory q. To the best of our knowledge, this is the first work that investigates this more general trajectory matching problem in spatial networks. Previous studies (e.g., [25, 26]) have simply assumed trajectories of equal length (i.e., the same number of sample points) and also given equal significance to all sample points. We treat each trajectory as a sequence of weighted sample points with arbitrary length.

For query processing, most of the existing works (e.g., [1, 6–8, 11, 17, 18, 24, 30]) are conducted in Euclidean space, and a spatial index (e.g., an R-tree [15]) is adopted to accelerate the query processing. In our work, object movement is constrained by a spatial network. The optimization techniques from Euclidean space are in general insufficient for solving the problem in spatial networks since the bounds proposed in Euclidean spaces are not always valid or appropriate in spatial networks. This is the main reason that the network expansion approach [9] is adopted in our work.

2.2 Searching Trajectories by Locations

The concept of searching trajectories by locations was first proposed by Chen et al. [8], in which context the query is a set of user specified query locations (e.g., sightseeing locations), while the result is the top-k trajectories that connect or are close to the query locations according to specific metrics. This type of query targets many important applications such as route planning and recommendation. However, it considers the spatial domain only (Euclidean space). Then Shang et al. [22] observe that the spatial similarity itself is not sufficient to capture the relationship between trajectories and query locations due to the more specific preferences of users. They thus propose a novel spatial-keyword problem, called user oriented trajectory search. Given a trajectory data set, the query input contains a set of query locations and a set of textual attributes describing the user's preferences. If a trajectory connects or is close to the specified query locations (spatial domain) and the textual attributes of the trajectory are similar to the user's preferences (textual domain), it is recommended to the user for reference.

The main difference between user oriented trajectory search (UOTS) [22] and personalized trajectory matching (PTM) is six fold. (*i*) **Query type**: the UOTS query is a spatial-keyword query that is conducted in spatial and textual domains, while the PTM query is a spatio-temporal query that is conducted in spatial and temporal domains. (*ii*) **Query input**: the UOTS query input is a set of query locations with equal significance in the spatial domain and a query point in the textual domain (i.e., by using TF-IDF method, a set of keywords represented as a high-dimensional vector), while the PTM query input is a user-specified trajectory, where sample points have different weights based on the user's preferences. In the spatial domain, it is a sequence of trajectory samples with different weights; and in the temporal domain, it is a sequence of timestamps with different weights. (*iii*) **Similarity function**: the similarity function used for the UOTS query evaluates the spatial-textual relevance in the spatial and textual domains. In the spatial domain, it measures the similarity between a set of query locations and a trajectory; and in the textual domain, it computes the Jaccard distance between two high dimensional vectors. The similarity function of the PTM query evaluates the spatio-temporal relevance between two different trajectories with arbitrary lengths in the spatial and temporal domains. (The paper also studies the performance of PTM query processing based on different trajectory similarity functions.) (*iv*) **Data model and algorithm structure**: the textual domain of UOTS is a high dimensional space, and its query input is a single point; in contrast, the temporal domain of PTM is one-dimensional, and its query input is a sequence of time points. Due to these differences (high-dimensional versus one-dimensional space, and single query point versus a sequence of time points), PTM and UOTS call for different algorithms. In UOTS, the textual similarity search is single source nearest neighbor search in a high-dimensional space, which is indexed using the iDistance method [16]. In PTM, we conduct a multiple sources similarity search in the temporal domain while considering the sequence and significance of each query source. (*v*) **Optimization techniques**: due to the different query inputs and similarity functions, PTM needs specific optimization techniques, including upper and lower bounds (Equations 7–11), query source sampling method (Equations 12–19), and heuristic functions (Equations 20–22). Also, query source sampling is used in PTM query processing. In the UOTS query, the number of query locations is usually small, and thus each query location can be regarded as a query source. When computing the PTM query, we carefully select a set of query sources from all trajectory sample points to reduce the search space overlap. (*vi*) **Experimental trajectory data sets**: different trajectory data sets are used. In the UOTS query, the trajectories contain spatial and textual attributes. In the PTM query, the trajectories contain spatial and temporal attributes. Due to these six differences, the PTM query is a new problem, and the solution to UOTS does not work for the PTM problem.

3 Preliminaries

3.1 Spatial Networks

A spatial network is modeled by a connected and undirected graph $G(V, E)$, where V is a set of vertices and E is a set of edges. A vertex $v_i \in V$ indicates a road intersection or an end of a road. An edge $e \in E$ is

defined as a pair of vertices and represents a segment connecting the two constituent vertices. For example, edge $e = \{v_i, v_j\}$ represents a road segment that enables travel between vertices v_i and v_j. A weight can be assigned to each edge to represent length or application specific factors such as travel time obtained by mining the historic traffic data [13]. Given two locations a and b in a spatial network, the network distance between them is the length of the shortest network path between them (i.e., a sequence of edges linking a and b where the accumulated weight is minimal). When weights model aspects such as travel time, the lower bound of network distance is not necessarily the corresponding Euclidean distance; thus, spatial indexes such as the R-tree [15] are not effective. To enhance the efficiency of the PTM query processing, we assume that all-pair shortest path distances have been pre-computed by Dijkstra's algorithm [9]. The time complexity of this pre-computing process is $O(V^2 lg(V) + VE)$, and the corresponding requirements (time and the maximum required memory space) are covered in Section 6.

3.2 Trajectory

Raw trajectory samples obtained from GPS devices are typically of the form of $\langle longitude, latitude, timestamp \rangle$. How to map such a sample onto a given spatial network is a research problem in its own right. We assume that all trajectory sample points have already been map matched onto the vertices of the spatial network by some map-matching algorithm (e.g., [2, 3, 14, 20, 28]), and that between two adjacent sample points a and b, the object movement always follows the shortest path connecting a and b. The spatio-temporal attributes of a trajectory are defined as follows:

Definition: Data trajectory
A data trajectory is a finite, time-ordered sequence $\langle v_1, v_2, ..., v_n \rangle$, where $v_i = (p_i, t_i)$, with p_i being a sample point (at a vertex) in G and t_i being a timestamp.

Definition: Query trajectory
A query trajectory is a weighted data trajectory where each sample point has a weight: $v_i = (p_i, t_i, w_i)$.

The value of a timestamp is set to be within the range of 24-hours, and the date is not taken into consideration. This setting is justified by two observations. In many practical scenarios like urban transportation, most peoples' movements recur on a daily basis. In addition, users of trajectory matching very often plan their routes on a 24-hour basis.

3.3 Trajectory Matching Functions

Given any two spatial points (without timestamps) a and b in a spatial network, the network shortest path between them is denoted as $SP(a, b)$, and its length is denoted as $sd(a, b)$. Given a trajectory τ and a spatial point o in a spatial network, the minimum distance $d_M(o, \tau)$ between spatial point o and trajectory τ is defined as

$$d_M(o, \tau) = \min_{v_i \in \tau} \{sd(o, v_i.p)\}, \tag{1}$$

where v_i is a sample point belonging to τ.

Given two timestamped trajectory sample points $v_1 \in \tau_1$ and $v_2 \in \tau_2$, the spatial influence factor $I_s(v_1, v_2)$ and temporal influence factor $I_t(v_1, v_2)$ between v_1 and v_2 are defined as

$$I_s(v_1, v_2) = \begin{cases} 0 & \text{if } sd(v_1.p, v_2.p) > \epsilon_s \\ e^{-sd(v_1.p, v_2.p)} & \text{otherwise} \end{cases} \tag{2}$$

$$I_t(v_1, v_2) = \begin{cases} 0 & \text{if } |v_1.t - v_2.t| > \epsilon_t \\ e^{-|v_1.t - v_2.t|} & \text{otherwise} \end{cases} \tag{3}$$

Here ϵ_s and ϵ_t are spatial and temporal matching thresholds, respectively. If the distance between v_1 and v_2 exceeds the threshold, the influence factor between them is set to 0. The values of $I_s(v_1, v_2)$ and $I_t(v_1, v_2)$ are in range $[0, 1]$.

Given a query trajectory $q = \langle o_1, o_2, ..., o_m \rangle$ and a data trajectory $\tau = \langle v_1, v_2, ..., v_n \rangle$, the spatial $S_{sim}(q, \tau)$ and temporal $T_{sim}(q, \tau)$ similarity between them are defined by the following equations.

$$S_{sim}(q, \tau) = \max \begin{cases} q.head.w \cdot I_s(q.head, \tau.head) \\ +S_{sim}(q.tail, \tau) \\ \\ S_{sim}(q, \tau.tail) \end{cases} \tag{4}$$

$$T_{sim}(q, \tau) = \max \begin{cases} q.head.w \cdot I_t(q.head, \tau.head) \\ +T_{sim}(q.tail, \tau) \\ \\ T_{sim}(q, \tau.tail) \end{cases} \tag{5}$$

Here $*.head$ is the first sample point of $*$, (e.g., $q.head = o_1$ and $\tau.head = v_1$) and $*.tail$ is the trajectory obtained by removing the head from $*$. (e.g., $q.tail = \langle o_2, o_3, ..., o_m \rangle$ and $\tau.tail = \langle v_2, v_3, ..., v_n \rangle$). The weight of sample point $q.v$ is denoted as $q.v.w$. These functions extend the well known Longest Common Subsequence (LCSS) [27] by taking the sequence and significance of each sample point $o \in q$ into account.

The values of the spatial similarity $S_{sim}(q, \tau)$ and the temporal similarity $T_{sim}(q, \tau)$ are both within the range of

$$[0, \sum_{1 \le i \le m} q.o_i.w],$$

where m is the number of sample points in q. By combining Equations 4 and 5, the spatio-temporal similarity between q and τ is defined as

$$ST_{sim}(q,\tau) = \lambda \cdot S_{sim}(q,\tau) + (1-\lambda) \cdot T_{sim}(q,\tau), \quad (6)$$

where parameter $\lambda \in [0,1]$ is used to adjust the relative importance of the spatial and temporal similarity terms. Note that in our setting, we allow users to adjust the parameter λ at the query time.

3.4 Problem Definition

Given a data trajectory set T, a query trajectory q with weighted sample points, the Personalized Trajectory Matching (PTM) query finds the trajectory $\tau \in T$ with the maximum value of $ST_{sim}(q,\tau)$, such that $\forall \tau' \in T \setminus \{\tau\}(ST_{sim}(q,\tau) \geq ST_{sim}(q,\tau'))$.

4 Baseline Method

In this section, we introduce balanced search as a straightforward baseline approach to Personalized Trajectory Matching. Given a trajectory data set T and a query trajectory q with user specified weights for each sample point $o \in q$, the balanced search method includes two steps: first, we explore the spatial and temporal domains concurrently and ask for trajectories similar to the query trajectory q in each domain; second, by integrating the results from two domains, the trajectory with the highest similarity to q is found.

4.1 Basic Idea

Consider an example in Figure 2. Trajectory τ_1 is the query trajectory, and τ_2 and τ_3 are data trajectories. Trajectory τ_1 is a sequence of 7 timestamped and weighted sample points o_i of the form $\langle p_i, t_i, w_i \rangle$. Points $\{p_1, p_2, ..., p_6\} \in \tau_2$ are the closest samples to $\{o_1.p, o_2.p, ..., o_7.p\}$, respectively. Point $p_7 \in \tau_3$ is the closest sample to $o_2.p$. To browse the spatial network and find the trajectories similar to the query trajectory (i.e., with higher values of $S_{sim}(q,\tau)$, refer to Equation 4), Dijkstra's expansion [9] is adopted. From each sample point $o_i.p \in \tau_1$, network expansion is performed using Dijkstra's algorithm, and the expansion speeds from different sample points are the same. Conceptually, the explored space is a circular region as shown in Figure 2, where the radius is the shortest network distance from the expansion center $o_i.p$ to the expansion boundary, denoted as $rs_i, i \in [1,7]$. According to Dijkstra's algorithm (Dijkstra's algorithm always selects the vertex with the minimum distance label for expansion), if $p \in \tau$ is the first sample point touched by the expansion from $o_i.p$, p is just the closest sample point to $o_i.p$ (i.e., $d_M(o_i.p,\tau) = sd(o_i.p,p)$). For instance, $p_1 \in \tau_2$ is the closest sample point to $o_1.p$, thus $d_M(o_1.p,\tau_2) = sd(o_1.p,p_1)$. Once a data trajectory has been touched by the expansions from all the sample points $o_i.p \in \tau_1$, such as τ_2 in Figure 2, we compute its spatial similarity to the query trajectory q based on Equation 4 (we have indexed the all-pairs shortest paths in Section 3.1) and this type of trajectory (e.g., τ_2) is marked as a "fully touched trajectory" in the spatial domain. Other trajectories such as τ_3 are marked as a "partly touched trajectory," and trajectories such as τ_4 are marked as an "untouched trajectory" in the spatial domain.

In the temporal domain, the timestamps of trajectory sample points $\{v.t \mid v \in \tau \wedge \tau \in T \cup \{q\}\}$ have been mapped onto a time axis. In Figure 2, $o_i.t$ is the timestamp of sample point o_i. To find the trajectories similar to the query trajectory τ_1 in the temporal domain (i.e., with higher values of $T_{sim}(q,\tau)$, refer to Equation 5), we browse the time axis by expanding the search from each $o_i.t, i \in [1,7]$ at the same speed. The browsed region of $o_i.t$ is the range $[o_i.t - rt_i,\ o_i.t + rt_i]$, where rt_i is the temporal radius of the search. The search radius rt_i is increased by one second (the minimum scale of the time axis) at each time, step by step, to form a larger scanned range until the targets are found. In the temporal domain, trajectories are also classified into three categories: fully, partly, and untouched trajectories. If a trajectory τ has been fully touched by the searches from the timestamps of all sample points, its temporal similarity $T_{sim}(q,\tau)$ can be computed based on Equation 5 and such trajectories are marked as a "fully scanned trajectory" in the temporal domain. Once a trajectory τ has been fully touched in both the spatial and the temporal domains, by combining the values of $S_{sim}(q,\tau)$ and $T_{sim}(q,\tau)$ based on Equation 6, its spatio-temporal similarity $ST_{sim}(q,\tau)$ can be derived.

4.2 Lower Bound of $ST_{sim}(q,\tau)$

To constrain the search space in both the spatial and the temporal domains, a pair of lower and upper bounds of the spatio-temporal similarity $ST_{sim}(q,\tau)$ (Equation 6) are designed[8]. If the upper bound of a trajectory τ is less than the lower bound of any other trajectories, τ cannot be the trajectory with the highest similarity to the query trajectory and can be pruned safely. Among

[8] In the following computation, we only consider the situations where for all $o_i \in q$, $rs_i < \epsilon_s$ and $rt_i < \epsilon_t$. If $rs_i \geq \epsilon_s$ (or $rt_i \geq \epsilon_t$), according to the definition of influence factor (Equations 2 and 3), for data point v outside the browsed region, we have $I_s(v,o_i) = 0$ (or $I_t(v,o_i) = 0$).

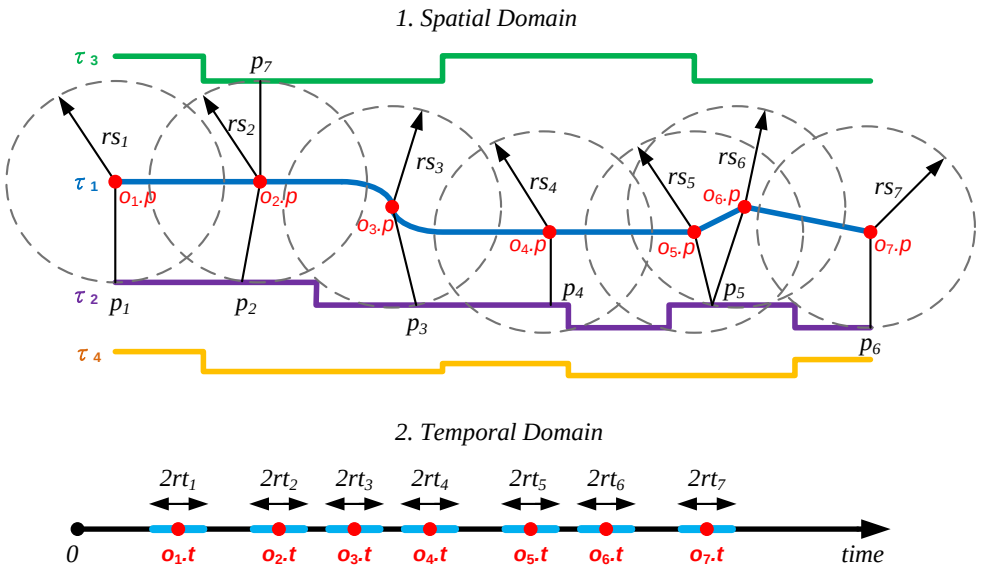

Fig. 2 An Example of Balanced Search

all trajectories fully scanned in the two domains, we define a global lower bound LB as

$$LB = \max_{\forall \tau \in T_f} \{ST_{sim}(q, \tau)\}, \tag{7}$$

where T_f is the set of trajectories that have been fully touched in both domains. Intuitively, LB is a dynamic value and is continuously updated during the search process.

4.3 Upper Bound of $ST_{sim}(q, \tau)$

We proceed to introduce our approach to estimating the upper bound of $ST_{sim}(q, \tau)$, where τ is a data trajectory that has not been fully touched in both domains. In the spatial domain, since Dijkstra's algorithm always selects the vertex with the minimum distance for expansion, if τ has not been touched by the expansion from $o_i.p$, we have $d_M(o_i.p, \tau) > rs_i$. The radius rs_i is the network distance of the current expansion from center $o_i.p$ (e.g., $d_M(o_1.p, \tau_3) > rs_1$ and $d_M(o_3.p, \tau_4) > rs_3$). Thus, for any sample point $v \in \tau$, we have

$$sd(v.p,\ o_i.p) \geq d_M(o_i.p, \tau) > rs_i$$

$$\Rightarrow e^{-sd(v.p,\ o_i.p)} < e^{-rs_i}$$

$$\Rightarrow I_s(v, o_i) < e^{-rs_i}$$

Refer to Equation 4 and let $q.head = o_i$ and $\tau.head = v$. If the value of $sd(v.p,\ o_i.p)$ cannot be obtained (i.e., $v.p$ has not been touched by the expansion from $o_i.p$), it is replaced by the value of rs_i, and the upper bound on the spatial similarity $S_{sim}(q, \tau).ub$ is estimated as

$$S_{sim}(q, \tau) = \max \begin{cases} q.head.w \cdot I_s(q.head, \tau.head) \\ +S_{sim}(q.tail, \tau) \\ \\ S_{sim}(q, \tau.tail) \end{cases}$$

$$\leq \max\{\alpha,\ S_{sim}(q, \tau.tail)\} = S_{sim}(q, \tau).ub \tag{8}$$

$$\alpha = \begin{cases} q.head.w \cdot e^{-sd(q.head.p,\ \tau.head.p)} \\ +S_{sim}(q.tail, \tau) & \text{if } C_1 \\ \\ q.head.w \cdot e^{-rs_i} + S_{sim}(q.tail, \tau) & \text{if } C_2 \end{cases}$$

C_1: the value of $sd(q.head.p, \tau.head.p)$ is available (can be obtained from the current network expansion).
C_2: the value of $sd(q.head.p, \tau.head.p)$ is not available and is replaced by the value of rs_i.

Similarly, in the temporal domain, if a trajectory τ has not been fully touched by the search from each $o_i.t$, for any timestamped sample point $v \in \tau$, we have

$$|v.t - o_i.t| > rt_i$$

$$\Rightarrow e^{-|v.t-o_i.t|} < e^{-rt_i}$$

$$\Rightarrow I_t(v, o_i) < e^{-rt_i}$$

Refer to Equation 5 and assume that $q.head = o_i$ and $\tau.head = v$. If the value of $|v.t - o_i.t|$ cannot be obtained (i.e., $v.t$ has not been touched by the search from $o_i.t$),

we use the value of rt_i to replace it and estimate the upper bound of temporal similarity $T_{sim}(q,\tau).ub$ as

$$T_{sim}(q,\tau) = \max \begin{cases} q.head.w \cdot I_t(q.head, \tau.head) \\ +T_{sim}(q.tail, \tau) \\ \\ T_{sim}(q, \tau.tail) \end{cases}$$

$$\leq \max\{\beta,\ T_{sim}(q,\tau.tail)\} = T_{sim}(q,\tau).ub \quad (9)$$

$$\beta = \begin{cases} q.head.w \cdot e^{-|q.head.t - \tau.head.t|} \\ +T_{sim}(q.tail,\tau) & \text{if } C_1 \\ \\ q.head.w \cdot e^{-rt_i} + T_{sim}(q.tail,\tau) & \text{if } C_2 \end{cases}$$

C_1: the value of $|q.head.t - \tau.head.t|$ is available.
C_2: the value of $|q.head.t - \tau.head.t|$ is not available and is replaced by the value of rt_i.

By combining the upper bounds of the spatial similarity (Equation 8) and the temporal similarity (Equation 9), the upper bound of the spatio-temporal similarity $ST_{sim}(q,\tau)$ is defined as follows.

$$ST_{sim}(q,\tau).ub = \begin{cases} \lambda \cdot S_{sim}(q,\tau) \\ +(1-\lambda) \cdot T_{sim}(q,\tau).ub & \text{if } C_1 \\ \\ \lambda \cdot S_{sim}(q,\tau).ub \\ +(1-\lambda) \cdot T_{sim}(q,\tau) & \text{if } C_2 \\ \\ \lambda \cdot S_{sim}(q,\tau).ub \\ +(1-\lambda) \cdot T_{sim}(q,\tau).ub & \text{if } C_3 \end{cases} \quad (10)$$

C_1: data trajectory τ is fully touched in the spatial domain and partly touched in the temporal domain.
C_2: data trajectory τ is partly touched in the spatial domain and fully touched in the temporal domain.
C_3: data trajectory τ is partly touched in both the spatial and the temporal domains.

Among all data trajectories that have not been fully touched in the spatial and temporal domains, we define the global upper bound UB as

$$UB = \max_{\tau \in T_n} \{ST_{sim}(q,\tau).ub\}, \quad (11)$$

where T_n is a set of trajectories that have not been fully touched in the two domains (i.e., $T_n = T \setminus T_f$). Similar to LB, UB is a quantity whose value changes dynamically during query processing.

To reduce the computation and storage loads, we only compute and maintain the lower bounds of partly touched trajectories. According to our definition, untouched trajectories must not have spatio-temporal upper bounds that are greater than those of partly touched trajectories. For instance, in Figure 2, τ_3 is a partly touched trajectory, and τ_4 is an untouched trajectory in the spatial domain. According to Equation 8, for expansion center (trajectory sample point) $o_2.p$, we have:

$$I_s(o_2,v).ub = \begin{cases} e^{-sd(o_2.p,\ p_7)} & \text{if } v.p = p_7 \in \tau_3 \\ e^{-rs_2} & \text{if } v.p \in \tau_3 \setminus \{p_7\} \end{cases}$$

$$I_s(o_2,v').ub = e^{-rs_2} \qquad \forall v' \in \tau_4$$

$$sd(o_2.p, p_7) \leq rs_2 \Rightarrow I_s(o_2,v).ub \geq I_s(o_2,v').ub$$

For the other expansion centers $o_i.p \in \tau_1 \setminus \{o_2.p\}$, it is easy to find that $\forall v \in \tau_3, \forall v' \in \tau_4, I_s(o_i,v).ub = I_s(o_i,v').ub$. By combining the results stated above, we obtain $S_{sim}(\tau_1,\tau_3) \geq S_{sim}(\tau_1,\tau_4)$. The temporal domain is handled similarly.

4.4 Algorithm

In the spatial and temporal domains, the network expansion and time axis search stop once $LB > UB$. All trajectories that have not been fully touched, including partly touched trajectories and trajectories completely outside the explored regions, can be pruned safely. The instant value of LB is the maximum spatio-temporal similarity to the query trajectory q, and the corresponding data trajectory τ is returned. The process of balanced search is detailed in Algorithm 1.

In Algorithm 1, network expansion is performed from each center (trajectory sample point) $o_i.p$ in turn. The expansion follows Dijkstra's algorithm [9], which always selects the vertex with the minimum distance label for expansion (lines 5–6). For each newly scanned trajectory τ in the spatial domain, if it has not been touched by the expansion from $o_i.p$ before, it is labeled as being touched by $o_i.p$, and its spatial similarity upper bound $S_{sim}(q,\tau).ub$ and global upper bound UB are updated correspondingly (lines 7–10). If we find a trajectory τ that has been fully touched in both the spatial and the temporal domains, we can compute the value of $ST_{sim}(q,\tau)$ based on Equation 6 (lines 11–12). If the value of $ST_{sim}(q,\tau)$ is greater than the global lower bound LB, the value of LB is replaced by the value of $ST_{sim}(q,\tau)$ (lines 13–14). If the value of LB is greater than that of UB or $rs_i \geq \epsilon_s$, the query processing is terminated (lines 15–16). In the temporal domain, the same search process is conducted and the search is expanded by incrementation by 1 ($rt = rt + 1$) (lines 17–18).

5 PTM Query Processing

The main drawback of balanced search as introduced in Section 4 is search space overlap. In the query trajectory, two adjacent sample points (usually close to each

Algorithm 1 Balanced Search

Data: trajectory set T, query trajectory q
Result: $\max_{\tau \in T}\{ST_{sim}(q,\tau)\}$ and the corresponding trajectory

```
1  LB ← 0;  UB ← 0;
2  S_sim(q,τ).ub ← 0;  T_sim(q,τ).ub ← 0;  ST_sim(q,τ).ub ← 0;
3  while true do
4      // in the spatial domain
5      for each o_i ∈ q do
6          Search(o_i.p);
7          for each newly scanned trajectory τ do
8              if τ.touch(o_i.p) = false then
9                  τ.touch(o_i.p) ← true;
10                 update S_sim(q,τ).ub, UB;
11             if ∀o_i ∈ q (τ.touch(o_i.p) ∧ τ.touch(o_i.t)) are true then
12                 compute ST_sim(q,τ);
13                 if ST_sim(q,τ) > LB then
14                     LB ← ST_dist(q,τ);
15             if (LB > UB) ∨ (rs_i ≥ ε_s) then
16                 return LB and the corresponding τ;
17     // in the temporal domain
18     The same search process as in the spatial domain, with (o_i.p, S_sim(q,τ), S_sim(q,τ).ub, rs_i, ε_s) replaced by (o_i.t,
       T_sim(q,τ), T_sim(q,τ).ub, rt_i, ε_t).
```

other in the spatial and temporal domains) that are both selected as expansion centers can have search spaces that overlap substantially. The trajectories in the overlapping regions will be accessed and processed repeatedly, which yields unnecessarily poor performance. In addition, balanced search lacks an effective strategy for scheduling which query sources to consider, again leading to poor performance.

To overcome these weaknesses and answer the PTM query efficiently, we propose an adaptive, two-phase search algorithm. First, we carefully select a set of sample points from the query trajectory q as query sources. Then we develop upper and lower bounds on the spatio-temporal similarity to q to constrain the search space in both domains (Section 5.1). An efficiency study reveals that our method defines the minimum search space during query processing (Section 5.2). Second, we propose a heuristic strategy based on priority ranking for the scheduling of the multiple query sources, thus avoiding to devote unnecessary search efforts to the trajectories that are unlikely to be promising (Section 5.3). Finally, the complete procedure of the proposed two-phase PTM search algorithm is detailed in Section 5.4.

5.1 Identifying Query Sources

To reduce search space overlap and improve the PTM query efficiency, a set of sample points are carefully selected as expansion centers from the query trajectory q. The set of query sources is denoted as $O_q.s$ in the spatial domain and as $O_q.t$ in the temporal domain. Other sample points can be attached to their nearest expansion center, and the spatial/temporal distance to trajectories can be estimated correspondingly. In the example in Figure 3, $\tau_1 = \langle o_1, o_2, ..., o_8\rangle$ is the query trajectory ($q = \tau_1$), and τ_2 and τ_3 are data trajectories in T. A set of sample points $O_q.s = \{o_2.p, o_4.p, o_7.p\}$ are selected as expansion centers in the spatial domain (how to select expansion centers is covered in Section 5.2), and other sample points are attached to their closest expansion centers (i.e., $\{o_1.p, o_3.p\}$ is attached to $o_2.p$; $o_5.p$ is attached to $o_4.p$; $\{o_6.p, o_8.p\}$ is attached to $o_7.p$). For a non-expansion-center sample point $o_b.p \in q$, its network distances to a data trajectory $\tau \in T$ is estimated based on Equation 12.

$$d_M(o_b.p,\tau) \geq rs_b = \begin{cases} rs_a - sd(o_a.p, o_b.p) & \text{if } C_1 \\ d_M(o_a.p,\tau) - sd(o_a.p, o_b.p) & \text{if } C_2 \end{cases} \quad (12)$$

C_1: τ has not been touched by the expansion from expansion center a.
C_2: τ has been touched by the expansion from expansion center a.

Here, $o_a.p \in O_q.s$ is $o_b.p$'s nearest expansion center. Consider the example shown in Figure 3. Here, $o_4.p$ is an expansion center, and $o_5.p$ is a non-expansion-center sample point attached to $o_4.p$. Trajectory τ_3 has not

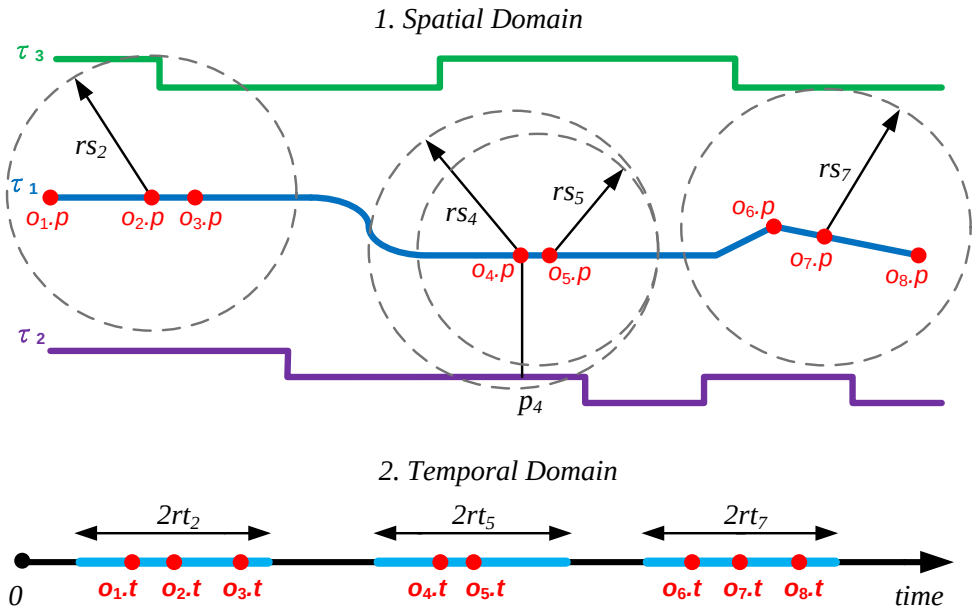

Fig. 3 Identifying Query Sources

been touched by the expansion from $o_4.p$, which means that τ_3 is outside the circular region $(o_4.p, rs_4)$. If the value of rs_5 is equal to that of $(rs_4 - sd(o_4.p, o_5.p))$, we can guarantee that the circular region $(o_5.p, rs_5)$ is enclosed by circular region $(o_4.p, rs_4)$. Then we are sure that τ_3 is outside the circular region $(o_5.p, rs_5)$, and the value of $d_M(o_5.p, \tau_3)$ is greater than that of rs_5. On the other hand, τ_2 has been touched by the expansion from $o_4.p$ and $p_4 \in \tau_4$ is the closest sample point to $o_4.p$ (i.e., $d_M(o_4.p, \tau_2) = sd(o_4.p, p_4)$). This means that τ_2 is tangent to the circular region $(o_4.p, d_M(o_4.p, \tau_2))$, and p_4 is the tangent point. If $rs_5 = d_M(o_4.p, \tau_2) - sd(o_4.p, o_5.p)$, we can guarantee that the circular region $(o_5.p, rs_5)$ is enclosed by the circular region $(o_4.p, rs_4)$, and thus we can get $d_M(o_4.p, \tau_2) \geq rs_5$.

For any sample point $v \in \tau$, we have

$$sd(v.p,\ o_b.p) \geq d_M(o_b.p,\ \tau) \geq rs_b$$

$$\Rightarrow I_s(v, o_b) = e^{-sd(v.p,\ o_b.p)} \leq e^{-rs_b}.$$

Then, we can substitute the values of rs_b (sample point $o_b.p$ is a non-expansion-center in the spatial domain; for expansion center $o_a.p$, we use its original radius rs_a.) in Equation 8 and the spatial similarity upper bound $S_{sim}(q, \tau).ub$ can be computed.

A similar procedure is conducted in the temporal domain to reduce the search space overlap. A set of sample points $O_q.t = \{o_2.t, o_5.t, o_7.t\}$ is carefully selected from the query trajectory q as expansion centers, and other non-expansion-center sample points in q are attached to their closest expansion centers. Note that the expansion centers in the spatial domain need not be the same as those in the temporal domain. For example, in Figure 3, $o_4.p$ is an expansion center in the spatial domain, but $o_4.t$ is a non-expansion-center in the temporal domain. For a non-expansion-center $o_b.t$, its temporal distance to $v.t,\ v \in \tau$, can be estimated as

$$|v.t - o_b.t| \geq rt_b = \begin{cases} rt_a - |o_a.t - o_b.t| & \text{if } C_1 \\ |v.t - o_a.t| - |o_a.t - o_b.t| & \text{if } C_2 \end{cases} \quad (13)$$

C_1: $v.t$ has not been touched by the search from the expansion center $o_a.t$.
C_2: $v.t$ has been touched by the search from the expansion center $o_a.t$.

Here, $o_a.t \in O_q.t$ is the closest expansion center to $o_b.t$. If $v.t$ has not been touched by the search from $o_a.t$, $v.t$ is outside the search range defined by $(o_a.t, rt_a)$, where $o_a.t$ is the expansion center and rt_a is the search radius. If the value of rt_b is equal to that of $(rt_a - |o_a.t - o_b.t|)$, we can guarantee that the search region $(o_b.t, rt_b)$ is enclosed by the search region $(o_a.t, rt_a)$. Thus $v.t$ is outside the search region $(o_b.t, rt_b)$ and $|v.t - o_b.t| > rt_b$. Once $v.t$ has been touched by the search from $o_a.t$, $rt_b = |v.t - o_a.t| - |o_a.t - o_b.t|$ guarantees that $v.t$ is outside or tangent with the search region $(o_b.t, rt_b)$, and $|v.t - o_b.t| \geq rt_b$. By integrating the two cases introduced above, we get $\forall v \in \tau, |v.t - o_b.t| \geq rt_b$. Next, the values of rt_b for non-expansion-center sample points can be substituted into Equation 9 to compute the temporal similarity upper bound $T_{sim}(q, \tau).ub$. By combining the spatial and temporal similarity upper bounds based on Equation 10, the spatio-temporal similarity upper bound $T_{sim}(q, \tau).ub$ is obtained.

5.2 Efficiency Study

In this section, we study the efficiency of the proposed algorithm for computing the PTM query by estimating the total search space in the spatial and temporal domains during query processing. In general, a small search space implies low computation cost, including data access and processing costs, and high query processing efficiency.

Assume that trajectories are uniformly distributed in both the spatial and the temporal domains and that sample points in the query trajectory q are uniformly distributed as well. In an extreme case where each sample point in τ is an expansion center (as in balanced search in Section 4), the search space for each individual expansion center is minimized while the number of expansion centers is maximized. Note that when any two adjacent sample points (usually close to each other) are selected as expansion centers, the search space overlap is substantial. The trajectories in the overlapping region will be read and processed repeatedly, which yields unnecessarily poor performance. In another extreme case, only the two ends of q (i.e., the source and destination) are selected as expansion centers. While the number of expansion centers is minimized, the search space for individual expansion center may be very large. The optimal selection of expansion centers can be estimated using linear programming. Let $\langle o_1 = s, o_2, .., o_{n-1}, o_n = t\rangle$ be the sample points in q ordered from source s to destination t. Let A be an $n \times n$ matrix where $a_{ij} = 1$ if the i^{th} and the j^{th} sample points are adjacent expansion centers (i.e., no sample points in-between them are expansion centers) and $a_{ij} = 0$, otherwise. Our goal is to minimize the objective functions in both domains:

$$\omega_s = \sum a_{ij} \frac{1}{4}\pi(d_{ij} + 2\epsilon_s)^2 \tag{14}$$

$$\omega_t = \sum a_{ij}(d_{ij} + 2\epsilon_t) \tag{15}$$

subject to $i < j$, $\sum_{j=1}^{n} a_{0j} = 1$, $\sum_{i=1}^{n} a_{i0} = 1$, $\sum_{j=1}^{n} a_{ij} \leq 1$, $\sum_{i=1}^{n} a_{ij} \leq 1$, and $\sum_{j=1}^{n} a_{ij} = \sum_{k=1}^{n} a_{ki}$. In the spatial domain, d_{ij} is the network distance between the i^{th} and the j^{th} sample points along the trajectory q, while in the temporal domain, d_{ij} is the temporal distance $|i.t - j.t|$ along the time axis. We use $\sum a_{ij}\frac{1}{4}\pi(d_{ij}+2\epsilon_s)^2$ and $\sum a_{ij}(d_{ij}+2\epsilon_t)$ to estimate areas of the total search space in the spatial and temporal domains, respectively, and we use $\sum a_{ij}$ to estimate the number of expansion centers.

Considering the online processing scenario, the time cost of finding the optimal selection of expansion centers by solving the above objective function may not be practical. Thus, we simplify the objective functions (Equations 14 and 15) by assuming that the gap between any two adjacent expansion centers is identical and the sample points in q are uniformly distributed. Our goal is now changed to finding the optimal number of expansion centers. Then, we have:

$$\omega_s(\theta) = \frac{\theta}{4}\pi(\frac{q.l_s}{\theta - 1} + 2\epsilon_s)^2 \tag{16}$$

$$\omega_t(\mu) = \mu(\frac{q.l_t}{\mu - 1} + 2\epsilon_t), \tag{17}$$

where $q.l_s$ and $q.l_t$ are the lengths of query trajectory q in the spatial and temporal domains, respectively, and θ and μ are the number of expansion centers in the two domains. The values of $\theta = (x+1)$ and μ resulting in the minimum ω_s and ω_t are obtained using the derivatives of the Functions 16 and 17.

$$\omega_s(\theta)' = \frac{\partial \omega}{\partial \theta} = 0$$

$$\Rightarrow 2q.l_s^2 x^{-3} + (q.l_s^2 + 4\epsilon_s q.l_s)x^{-2} + 4\epsilon_s = 0 \tag{18}$$

$$\omega_t(\mu)' = \frac{\partial \omega}{\partial \mu} = 0 \Rightarrow \mu = \sqrt{\frac{q.l_t}{2\epsilon_t}} + 1 \tag{19}$$

The cubic equation in Equation 18 can be solved by applying the general formula of roots. Then $\theta = (x + 1)$ uniformly distributed sample points (including the source and destination) are selected in q as the expansion centers. In Equation 19, the value of μ is computed directly. The necessity of the query source sampling step in the spatial and temporal domains depends on the values computed for θ and μ in the two domains, respectively. If the number of vertices in the query trajectory is equal to the value of θ (i.e., $|q| = \theta$), it is not necessary to conduct query source sampling in the spatial domain. Similarly, if $|q| = \mu$, it is not necessary to conduct query source sampling in the temporal domain. By following the aforementioned procedure, the expansion centers that define the minimum search spaces in the two domains are found.

5.3 Heuristic Trajectory Search Strategy

We proceed to introduce a heuristic scheduling strategy based on priority ranking of multiple query sources (i.e., the selected expansion centers in the spatial and temporal domains) adopted in the PTM query processing. A carefully crafted scheduling strategy can help avoid devoting unnecessary search effort to trajectories that are unlikely to be the optimal choice and can thus further enhance query efficiency.

Consider the scenario demonstrated in Figure 4. In the spatial domain, trajectory τ_2 has been fully touched

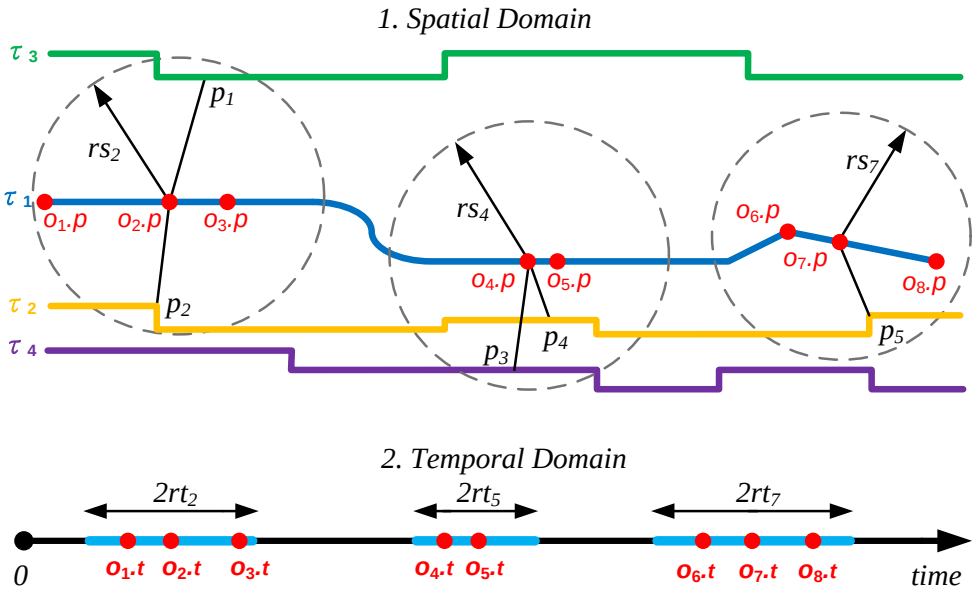

Fig. 4 Heuristic Trajectory Search Strategy

(i.e., touched by the expansion from all selected expansion centers $O_q.s = \{o_2.p, o_4.p, o_7.p\}$), while trajectories τ_3 and τ_4 are both partly touched (τ_3 has been touched by the expansion from $o_2.p$, and τ_4 has been touched by the expansions from $o_4.p$ and $o_7.p$). Each expansion center $o_a.p \in O_q.s$ is given a label $o_a.p.priority$ to describe its priority. We carefully maintain a dynamic priority heap ordered on $o_a.p.priority$ that contains these query sources. In each step, we consider the query source on the top of the heap and expand the corresponding search until its place is replaced. Then we expand from the new top-ranked query source. The priority of each query source $o_a.p \in O_q.s$ is defined as follows.

$$o_a.p.priority = \frac{\sum_{p_i \in C(o_a.p)} o_i.w}{\sum_{o_j \in q} o_j.w} \sum_{\tau_i \in T_p \setminus T_t(o_a.p)} e^{S_{sim}(q,\tau_i).ub} \quad (20)$$

Here, $C(o_a.p)$ is a cluster of sample points that contains expansion center $o_a.p$ and all non-expansion-center sample points considering $o_a.p$ as their nearest expansion center. T_p is the set of partly touched trajectories in the spatial domain (e.g., $T_p = \{\tau_3, \tau_4\}$) and $T_t(o_a.p)$ is the set of trajectories touched by the expansion from $o_a.p$ (e.g., $T_t(o_2.p) = \{\tau_2, \tau_3\}$, $T_t(o_4.p) = \{\tau_2, \tau_4\}$, $T_t(o_7.p) = \{\tau_2\}$, etc.). The fully touched trajectories (e.g., τ_2) and all untouched trajectories (i.e., the trajectories that have not been touched by any expansion in the spatial domain) are not taken into account in this ranking model.

Fundamentally, the priority of $o_a.p$ should reflect the significance of cluster $C(o_a.p)$ (sample point $o_a.p$ is a selected expansion center in the spatial domain, and $\{p_i \mid p_i \in C(p_a)\} \setminus \{o_a.p\}$ are attached to p_a). The higher the significance, the higher the priority. The significance of $C(o_a.p)$ can be expressed as the ratio of $\sum_{p_i \in C(o_a.p)} o_i.w$ to $\sum_{o_j \in q} o_j.w$. Then, for partly touched trajectories, our target is to transform them into fully touched trajectories as soon as possible (as mentioned before, only if a trajectory has been full touched, we compute its spatial similarity to q according to Equation 4). To be a fully touched trajectory, a trajectory should be touched by the expansions from all expansion centers $o_a.p \in O_q.s$. The priority of $o_a.p$ is proportional to its "margin" (i.e., the size of $T_p \setminus T_t(p_a)$). For example, in Figure 4, $T_p = \{\tau_3, \tau_4\}$, $T_t(o_2.p) = \{\tau_2, \tau_3\}$, and $T_p \setminus T_t(o_2.p) = \{\tau_4\}$. As a result, the margin of p_1 is 1. Further, $T_t(o_7.p) = \{\tau_2\}$ and $T_p \setminus T_t(p_7) = \{\tau_3, \tau_4\}$. Hence the margin of p_7 is 2. Next, $S_{sim}(q,\tau).ub$ is used to estimate the spatial similarity between q and τ. Intuitively, a trajectory τ with larger $S_{sim}(q,\tau).ub$ should have a higher probability to be the trajectory with the highest spatial similarity to q. If $\tau \in T_p \setminus T_t(o_a.p)$, the value of $S_{sim}(q,\tau).ub$ should be proportional to the priority of $o_a.p$.

Similarly, in the temporal domain, the priority of each query source $o_a.t \in O_q.t$ is given as follows.

$$o_a.t.priority = \frac{\sum_{o_i.t \in C(o_a.t)} o_i.w}{\sum_{o_j \in q} o_j.w} \sum_{\tau_i \in T_p \setminus T_t(o_a.t)} e^{T_{sim}(q,\tau_i).ub} \quad (21)$$

In the above, we only consider the situations where no trajectory has been touched in both the spatial and

the temporal domains. Once a trajectory has been touched in both domains, a new ranking method is necessary to evaluate the priority of each query source $o_a \in \{O_q.s \cup O_q.t\}$, and the priority labels is calculated as follows.

$$o_a.priority = \frac{\sum_{o_i \in C(o_a)} o_i.w}{\sum_{o_j \in q} o_j.w} \sum_{\tau_i \in T_p \setminus T_t(o_a)} e^{ST_{sim}(q,\tau_i).ub} \quad (22)$$

Here, T_p is the set of partly touched trajectories in both the spatial and temporal domains, and $T_t(o_a)$ is the set of trajectories touched by the search from o_a. The trajectories that have been fully touched in the both domains and the trajectories that have not been touched by search in any of the two domains are disregarded.

5.4 Algorithm

The two-phase PTM query processing algorithm is shown in Algorithm 2. Initially, the default values of the global lower bound LB (Equation 7), the global upper bound UB (Equation 11), and the upper bounds of the spatial (Equation 8), the temporal (Equation 9), the spatio-temporal similarity (Equation 10) are set to 0 (lines 1–2). Then, expansion centers in the both spatial and temporal domains are selected according to the method introduced in Section 5.1 (line 3). The values of the priority labels of the query sources (i.e., $\forall o_i.p \in O_q.s$ ($o_i.p.priority$), $\forall o_j.t \in O_q.t$ ($o_j.t.priority$) and $\forall o \in \{O_q.s \cup O_q.t\}$($o.priority$)) are set to 0 (lines 4–5). While no trajectory has been touched in both the spatial and temporal domains, the query processing is conducted in the two domains independently.

In the spatial domain, the query source with the maximum value of $o_i.p.priority$ is selected as the current search point $Csp.s$ (line 6), and the search is from $Csp.s$ according to Dijkstra's algorithm (line 10). For each newly scanned trajectory τ, if τ has not been touched by the expansion from $o_i.p$ before, its label is updated as being touched by $o_i.p$ (lines 11–13). During this process, $S_{sim}(q,\tau).ub$ and other related parameters are updated correspondingly (line 14). If there exists a query source $o'.p \in O_q.s \setminus \{o_i.p\}$ with a higher value of $o'.p.priority$ than that of $o_i.p.priority$, $o'.p$ replaces $o_i.p$ as the new search point, and the expansion from $o_i.p$ terminates (lines 15–16). If a trajectory has been touched in both the spatial and the temporal domains, the current search terminates (lines 17–18), and a new priority ranking method (Equation 22) is adopted to schedule different query sources in the two domains. In the temporal domain, the same search process is conducted and the search is expanded as $rt = rt + 1$ (lines 19–20).

Then, we select the query source with the maximum value of $o.priority$ as the current search point Csp (line 21). If $o \in O_q.s$, the search region is expanded according to Dijkstra's algorithm. If $o \in O_q.t$, the search is expanded by incrementing rt by 1. Each newly scanned trajectory τ is labeled as being touched by the search from o, and the values of $ST_{sim}(q,\tau).ub$ and all related parameters (e.g., $o.priority$, UB) are updated correspondingly (lines 24–28). If there exists a query source $o' \in O_q.s \cup O_q.t \setminus \{o\}$ with $o'.priority > o.priority$, o' replaces o as the new expansion center, and the search from o terminates (lines 29–30). If a trajectory has been touched by all query sources in the two domains, the value of $ST_{sim}(q,\tau)$ is computed based on Equation 6. If $ST_{dist}(q,\tau) > LB$, LB is updated to be the value of $ST_{sim}(q,\tau)$ (lines 31–34). Once the value of LB exceeds that of UB or the search radiuses of all query sources exceed the matching thresholds, the trajectory τ with the maximum value of $ST_{sim}(q,\tau)$ (LB) is returned, and the search terminates (lines 35–36).

Extension: it is straightforward to extend the proposed techniques to support a top-k PTM query. Among all data trajectories fully scanned in the spatial and temporal domains, we define a global lower bound LB_k for a top-k PTM query as

$$LB_k = \min_{\tau \in T_k} \{ST_{sim}(q,\tau)\}, \quad (23)$$

where T_f is the set of trajectories that have been fully touched in the both domains, and $T_k = \{\tau_1, \tau_2, ..., \tau_k\}$ ($T_k \subseteq T_f$), such that

$$\forall \tau \in T_k (\forall \tau' \in T_f \setminus T_k (ST_{sim}(q,\tau) \geq ST_{sim}(q,\tau'))).$$

The value of LB_k is dynamic and is updated continuously during the search process.

The search process for the top-k PTM query is conducted by substituting Equation 23 into Algorithms 1 and 2. If the value of LB_k exceeds UB, the query processing terminates and trajectory data set T_k is returned as the top-k result.

6 Experiments

Next, we report on an extensive experimental study with real and synthetic spatial data sets to understand the performance properties of the two PTM query processing algorithms presented in the paper. We use graphs extracted from two spatial networks in the studies, namely the Beijing Road Network (BRN)[9] and the North America Road Network (NRN)[10], which contain 28,342

[9] http://www.iscas.ac.cn/

[10] http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm

Algorithm 2 Two-Phase PTM Query Processing

Data: trajectory set T, query trajectory q
Result: $\max_{\tau \in T}\{ST_{sim}(q,\tau)\}$ and the corresponding trajectory

$LB \leftarrow 0$; $UB \leftarrow 0$;
$S_{sim}(q,\tau).ub \leftarrow 0; T_{sim}(q,\tau).ub \leftarrow 0; ST_{sim}(q,\tau).ub \leftarrow 0$;
select query sources in the spatial and temporal domains;
$\forall o_i.p \in O_q.s$ $(o_i.p.priority \leftarrow 0)$; $\forall o_j.t \in O_q.t$ $(o_j.t.priority \leftarrow 0)$;
$\forall o \in \{O_q.s \cup O_q.t\}$ $(o.priority \leftarrow 0)$;
$Csp.s \leftarrow o_i.s \in O_q.s$ with the maximum value of $o_i.p.priority$;
$Csp.t \leftarrow o_j.t \in O_q.s$ with the maximum value of $o_j.t.priority$;
while *true* **do**
  //in the spatial domain
  $Search(Csp.s)$;
  for *each newly scanned trajectory* τ **do**
    if $\tau.touch(o_i.p) = false$ **then**
      $\tau.touch(o_i.p) \leftarrow true$;
      Update $S_{sim}(q,\tau).ub$ and all related parameters;
  if $\exists o'.p \in O_q.s \setminus \{o_i.p\}$ $(o'.p.priority > o_i.p.priority)$ **then**
    $Csp.s \leftarrow o'.p$;
  if $\exists o_j.t \in O_q.t$ $(\tau.touch(o_j.t) = true)$ **then**
    Break;
  //in the temporal domain
  The same search process as in the spatial domain, with ($Csp.s$, $O_q.s$, $S_{sim}(q,\tau).ub$, $Priority_S(s_i)$) replaced by ($Csp.t$, $O_q.t$, $T_{sim}(q,\tau).ub$, $Priority_T(s_j)$).
$Csp \leftarrow o \in \{O_q.s \cup O_q.t\}$ with the maximum $o.priority$;
while *true* **do**
  $Search(Csp)$;
  for *each newly scanned trajectory* τ **do**
    if $\tau.touch(o) = false$ **then**
      $\tau.touch(o) \leftarrow true$;
      Update $ST_{sim}(q,\tau).ub$ and all related parameters;
      $UB = \max\{ST_{sim}(q,\tau).ub\}$;
  if $\exists o' \in O_q.s \cup O_q.t \setminus \{o\}$ $(o'.priority > o.priority)$ **then**
    $Csp \leftarrow o'$;
  if $\forall o_i \in O_q.s \cup O_q.t$ $(\tau.touch(o_i) = true)$ **then**
    Compute $ST_{dist}(q,\tau)$;
    if $ST_{dist}(q,\tau) > LB$ **then**
      $LB \leftarrow ST_{sim}(q,\tau)$;
  if $LB > UB \vee (\forall o_i.p \in O_q.s\ (rs_i \geq \epsilon_s) \wedge \forall o_j.t \in O_q.t\ (rt_j \geq \epsilon_t))$ **then**
    return *LB and the corresponding trajectory* τ;

vertices and 175,812 vertices, respectively. The graphs are stored as adjacency lists. We pre-compute shortest path distances between all-pair vertices for each graph by using Dijkstra's algorithm [9] to enhance the efficiency of the PTM query processing. The preprocessing requirements (time and the maximum required memory space) are listed in Table I. In addition to BRN and NRN, we also test the pre-computing algorithm on a much larger road network, the Pennsylvania Road Network (PRN)[11] with 1,088,092 vertices, to observe its performance. The pre-computing algorithm was implemented in C++ and run on a cluster with 16 servers. Each server has a 16-core CPU (2.40GHz) and 300GB memory. Note that the maximum required memory in a server is the memory shared by 16 parallel threads.

For the studies with BRN, we used a real trajectory data set collected by the MOIR project [19]. Raw trajectory samples obtained from GPS devices are of the form $\langle longitude, latitude, timestamp\rangle$. They were map matched to the vertices in the corresponding graph according to the *Passby* map-matching method [20]. Between any two adjacent sample points a and b, we assume that the object follows the shortest path connecting a and b. For the studies with NRN, synthetic trajectory data was used. The generated synthetic trajectory samples are of the form of $\langle vertex,\ timestamp\rangle$. Syn-

[11] http://snap.stanford.edu/data/index.html

Table I: Pre-Processing Requirements

	Time (second)	**Memory Space (byte)**
All-pair shortest path BRN (28,342 vertices)	8.5	100,868,096 (≈ 96.2 MB)
All-pair shortest path NRN (175,812 vertices)	380.8	661,221,376 (≈ 630.6 MB)
All-pair shortest path PRN (1,088,092 vertices)	23,404.5 (≈ 6.5 hours)	4,817,809,408 (≈ 4.5 GB)
Map-matching BRN (10,000 trajectories)	211.6	65,011,743 (≈ 62.0 MB)

Table II: Parameter Settings

	BRN	**NRN**
Number of trajectories, $\|T\|$	6,000–10,000 (default 8,000)	10,000–30,000 (default 20,000)
Query trajectory length, $q.length$	20–100 (default 60)	20–100 (default 60)
The relative importance parameter, λ	0–1 (default 0.5)	0–1 (default 0.5)

thetic trajectories are uniformly distributed in NRN, and their lengths are from 20 to 100. Map-matching algorithms was implemented in C++ and run on a Windows 7 platform with an Intel Core i5-2410M Processor (2.67GHz, 3MB L3) and 8GB memory. The requirements (time and maximum required memory space) of the map-matching process in BRN are also listed in Table I.

In the experiments, the graphs were memory resident when running Dijkstra's algorithm, while the trajectory data was stored on disk due to its large size. To achieve efficient data access, a trajectory hashing approach was adopted in both the spatial and temporal domains. A data trajectory was represented as a sequence of timestamped sample points $o_i = (p_i, t_i)$, where p_i is a vertex in $G(V, E)$ and t_i is a timestamp. For each vertex $p_i \in V$, we maintain a pointerlist $p_{i.traj}$ to identify the trajectories that contain p_i (i.e., pointing to the positions of the trajectories on disk). Similarly, in the temporal domain, all timestamps were mapped onto a 24-hour time axis, and the smallest unit is 1 second. For each time point t on the time axis, we maintain a pointerlist $t._{traj}$ to identify the trajectories that contain t. If a vertex v was scanned by a network expansion in the spatial domain, or a time point t was scanned by a search in the temporal domain, the related trajectories (i.e., the trajectories in the corresponding pointerlist) can be accessed efficiently.

All algorithms of PTM were implemented in C++ and run on a Windows 7 platform with an Intel Core i5-2410M Processor (2.67GHz, 3MB L3) and 8GB memory. The maximum memory spaces required by PTM query processing are listed in table III.

Table III: Maximum Memory Space Required by PTM Query Processing

	BRN (byte)	**NRN (byte)**
balanced algorithm	123,887,616 (≈ 118.2 MB)	929,390,592 (≈ 886.3 MB)
two-phase algorithm	43,319,296 (≈ 41.3 MB)	383,474,668 (≈ 365.7 MB)

All experimental results are averaged over 50 independent trails with different query inputs. The main performance metrics are CPU time and the number of visited trajectories. The number of visited trajectories is selected as a metric for two reasons: (*i*) it describes the exact amount of data accesses; (*ii*) it reflects the real disk I/O cost to a certain degree. The parameter settings are listed in Table II. By default, the number of trajectories were 8,000 and 20,000 in BRN and NRN, respectively. The length of a query trajectory was set to 60, and λ was set to 0.5 for both BRN and NRN. Each query trajectory was randomly selected from the trajectory data set T. The weight of each sample point in a query trajectory was also randomly generated and the weights are integers in the range [1, 5]. The two-phase algorithm (Section 5) is denoted as "Two-phase PTM" in Figures 5, 6, and 7. For the purpose of comparison, two naive algorithms were also implemented: balanced search (Section 4) denoted as "Balanced" and the two-phase algorithm without the heuristic search strategy, denoted as "PTM without heuristic" in Figures 5, 6, and 7.

6.1 Effect of the Number of Trajectories $|T|$

First of all, we investigated the effect of the number of trajectories $|T|$ on the performance of the three search algorithms with the default settings. Intuitively, the denser the trajectory data is, the smaller the required search space is, and thus the queries can be faster. In Figure 5, the CPU time and the number of visited trajectories for balanced search and both two-phase algorithms (with and without heuristic search strategy) decrease with the increasing number of trajectories. From Figure 5, it is clear that the CPU time and the number of visited trajectories required by balanced search are 3–4 times higher than those of the two-phase algorithm. In addition, with the help of the query source sampling step (Section 5.1), the performance of the balanced algorithm is improved by approximately a factor of 3 in terms of both CPU time and the number of visited tra-

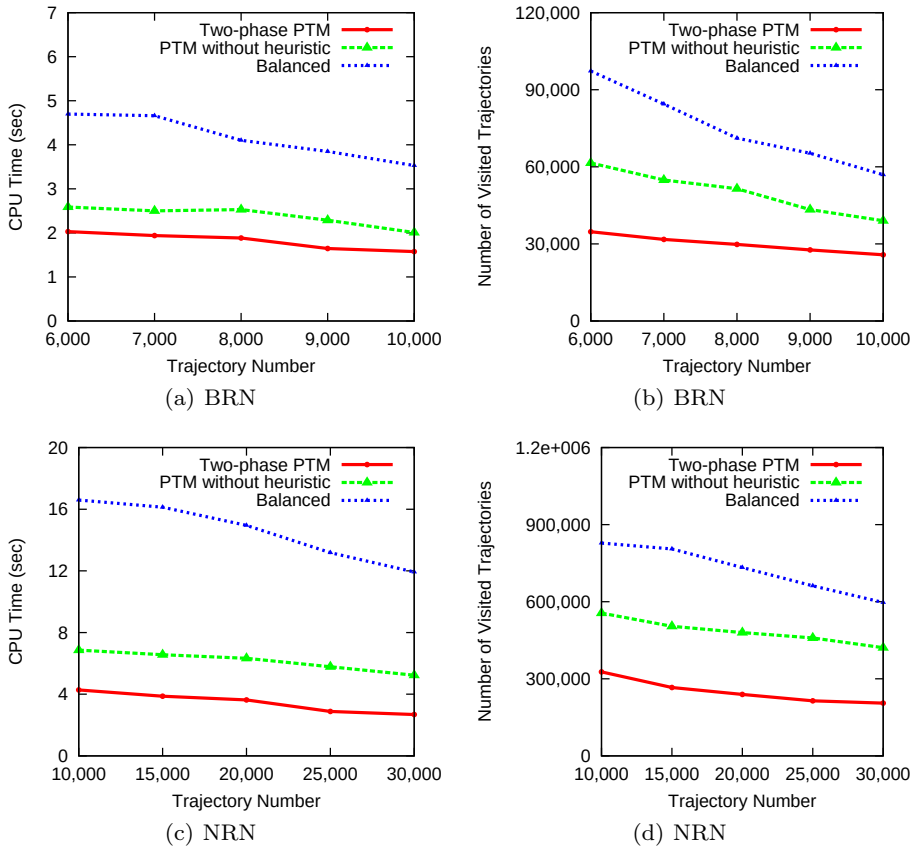

Fig. 5 Effect of the Number of Trajectories

jectories (by comparing "PTM without heuristic" to "Balanced"). Furthermore, with the help of the heuristic search strategy (Section 5.3), the performance of the two-phase algorithm is improved by approximately a factor of 2 in terms of both CPU time and the number of visited trajectories (by comparing "PTM without heuristic to "Two-Phase PTM). These results demonstrate the importance of carefully selecting expansion centers and developing the corresponding lower and upper bounds (to prune the search space), and the benefit of the heuristic search strategy.

Note that (*i*) the number of visited trajectories may be greater than the size of trajectory data set T since a trajectory may be visited several times from different query sources; (*ii*) the CPU time is not fully aligned with the number of visited trajectories. To prune the search space, the two-phase algorithm needs more computational effort to maintain the bounds and the priority heap. In some cases, the increment of the computation cost may offset the benefits of the reduction in the number of visited trajectories.

6.2 Effect of Query Trajectory Length *q.length*

Figure 6 presents the performance of the three algorithms with varying number of sample points in a query trajectory. Since more sample points cause more query sources to be processed, the CPU time and the number of visited trajectories are expected to be higher for all three search algorithms. However, the CPU time and the number of visited trajectories of balanced search increase much faster than for the two-phase algorithms for two reasons. First, all the sample points in the query trajectory are treated as query sources, which causes even more visits of trajectories. Second, balanced search treats all query sources equally since no heuristic search strategy is adopted. For instance, with *q.length* = 100, the two-phase algorithm outperforms balanced search by almost a factor of 5 (for both CPU time and visited trajectories).

6.3 Effect of λ

The value of λ is adopted to adjust the relative importance of the spatial and temporal similarity factors. In

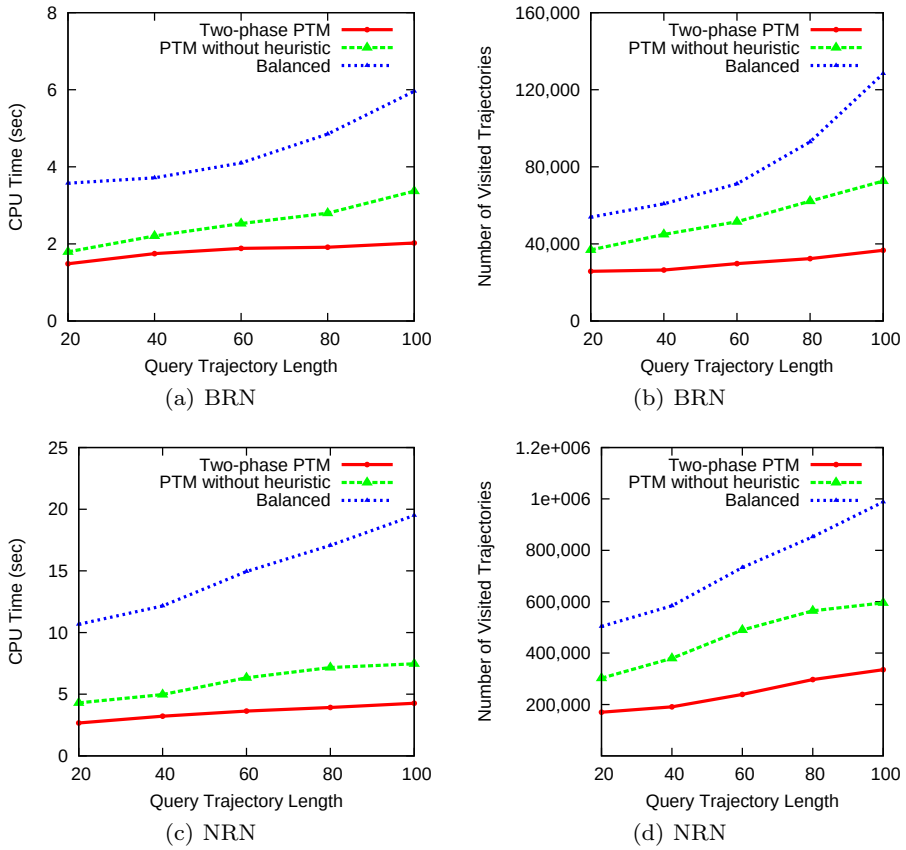

Fig. 6 Effect of Query Trajectory Length

the extreme case where $\lambda = 1$, the PTM query is conducted in the spatial domain only. On the other hand, when $\lambda = 0$, the temporal similarity is the sole factor considered. Figure 7 demonstrates the performance of the three algorithms for different values of λ. It is clear that the search effort (CPU time and visited trajectories) required in the spatial domain is higher than that required in the temporal domain.

6.4 Effect of Different Similarity Functions

We also study the performance of PTM query processing with different trajectory similarity functions. Two other typical trajectory similarity functions, NNT [24] and BCT [8], are selected and they are suitable for spatial trajectories (rather than for data streams) with arbitrary lengths.

By integrating NNT with spatial and temporal matching thresholds ϵ_s and ϵ_t and the significance of trajectory samples, the NNT similarity between a query trajectory q and a data trajectory τ is defined as follows.

$$S_{dist}(v,\tau) = \min_{\forall v' \in \tau} \{sd(v.p, v'.p)\}$$

$$I_s(v,\tau) = \begin{cases} 0 & \text{if } S_{dist}(v,\tau) > \epsilon_s \\ S_{dist}(v,\tau)^{-1} & \text{otherwise} \end{cases} \quad (24)$$

Here, v and v' are trajectory samples in q and τ, respectively, and $S_{dist}(v,\tau)$ is the minimum spatial distance between v and τ.

$$T_{dist}(v,\tau) = \min_{\forall v' \in \tau} \{|v.t - v'.t|\}$$

$$I_t(v,\tau) = \begin{cases} 0 & \text{if } T_{dist}(v,\tau) > \epsilon_t \\ T_{dist}(v,\tau)^{-1} & \text{otherwise} \end{cases} \quad (25)$$

Here $T_{dist}(v,\tau)$ is the minimum temporal distance between v and τ. The spatial similarity, temporal similarity, and spatio-temporal similarity of NNT are given in Equations 26, 27, and 28.

$$S_{sim}(q,\tau) = \sum_{i=1}^{|q|} v_i.w \cdot I_s(v_i,\tau) \quad (26)$$

$$T_{sim}(q,\tau) = \sum_{i=1}^{|q|} v_i.w \cdot I_t(v_i,\tau) \quad (27)$$

$$ST_{sim}(q,\tau) = \lambda \cdot S_{sim}(q,\tau) + (1-\lambda) \cdot T_{sim}(q,\tau) \quad (28)$$

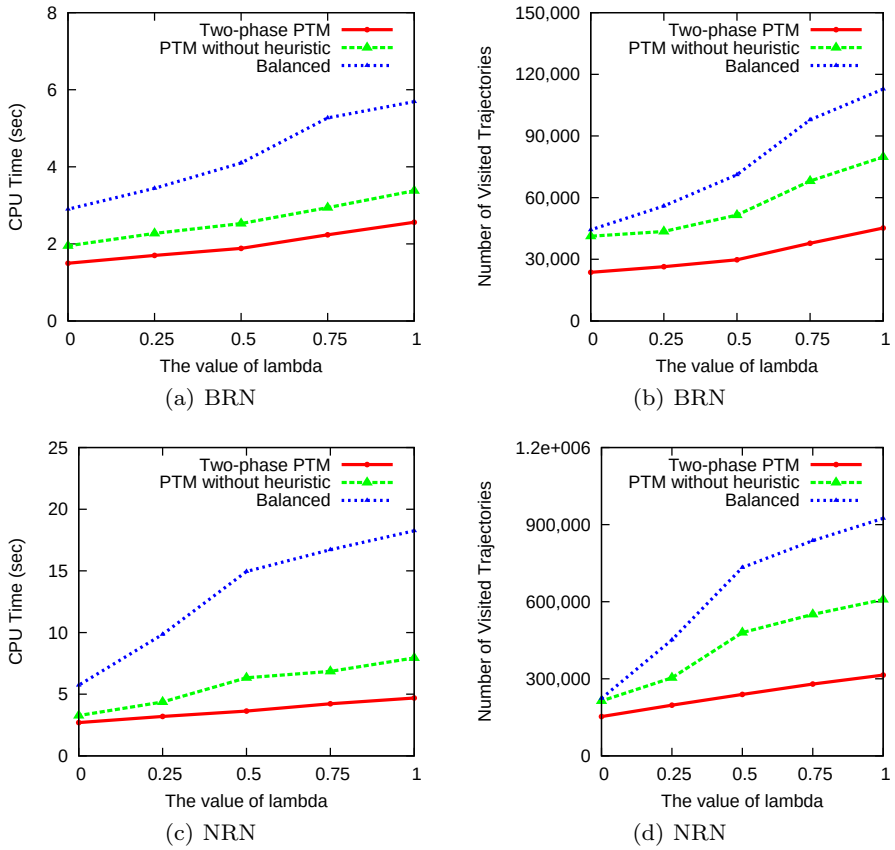

Fig. 7 Effect of λ

The upper bound on spatial similarity $S_{sim}(q,\tau).ub$ and the upper bound on temporal similarity $T_{sim}(q,\tau).ub$ are defined as follows.

$$S_{sim}(q,\tau).ub = \sum_{\{v_i|\tau\in S_t(i)\}} v_i.w \cdot S_{dist}(v_i,\tau) + \sum_{\{v_j|\tau\notin S_t(j)\}} v_j.w \cdot rs_j^{-1} \quad (29)$$

Here $S_t(i)$ is a set of trajectories that are touched by the expansion from $v_i.p$ in the spatial domain, and rs_j is the radius of the expansion region of $v_j.p$.

$$T_{sim}(q,\tau).ub = \sum_{\{v_i|\tau\in T_t(i)\}} v_i.w \cdot T_{dist}(v_i,\tau) + \sum_{\{v_j|\tau\notin T_t(j)\}} v_j.w \cdot rt_j^{-1} \quad (30)$$

Here $T_t(i)$ is a set of trajectories that are touched by the expansion from $v_i.t$ in the temporal domain, and rt_j is the radius of the expansion region of $v_j.t$.

Then, following the procedures introduced in Sections 4 and 5, other parameters such as the upper bound of spatio-temporal similarity $ST_{sim}(q,\tau).ub$, global upper bound UB, and heuristic functions, can be computed directly. The PTM query is processed using the NNT similarity function by substituting the corresponding functions and parameters into Algorithm 2.

Similarly, by integrating BCT with the spatial and temporal matching thresholds ϵ_s and ϵ_t and the significance of trajectory samples, the BCT similarity between a query trajectory q and a data trajectory τ is defined as follows.

$$I_s(v,\tau) = \begin{cases} 0 & \text{if } S_{dist}(v,\tau) > \epsilon_s \\ e^{-S_{dist}(v,\tau)} & \text{otherwise} \end{cases} \quad (31)$$

$$I_t(v,\tau) = \begin{cases} 0 & \text{if } T_{dist}(v,\tau) > \epsilon_t \\ e^{-T_{dist}(v,\tau)} & \text{otherwise} \end{cases} \quad (32)$$

The spatial similarity, temporal similarity, and spatio-temporal similarity are computed by substituting Equations 31 and 32 into Equations 26, 27, and 28. The upper bounds of the spatial and temporal similarities are given in Equations 33 and 34.

$$S_{sim}(q,\tau).ub = \sum_{\{v_i|\tau\in S_t(i)\}} v_i.w \cdot S_{dist}(v_i,\tau) + \sum_{\{v_j|\tau\notin S_t(j)\}} v_j.w \cdot e^{-rs_j} \quad (33)$$

$$T_{sim}(q,\tau).ub =$$

$$\sum_{\{v_i|\tau\in T_t(i)\}} v_i.w\cdot T_{dist}(v_i,\tau)+\sum_{\{v_j|\tau\notin T_t(j)\}} v_j.w\cdot e^{-rt_j} \quad (34)$$

Using the same procedure as for NNT, other parameters such as $ST_{sim}(q,\tau).ub$, UB, and heuristic functions, can be computed directly. The PTM query is computed using the BCT similarity function by substituting the corresponding functions and parameters into Algorithm 2.

The two-phase algorithm based on the LCSS similarity function (Equations 2–6, the main metric used in this work) is denoted as "PTM-LCSS-Order" since the order of trajectory samples is considered in trajectory matching. In addition, the two-phase algorithm based on NNT similarity [24] (Equations 24–28) and BCT similarity [8] (Equations 31–32) are denoted as "PTM-NNT" and "PTM-BCT", respectively. The performance of PTM query processing using the different similarity functions is shown in Figure 8. Compared to NNT and BCT (without an order constraint), LCSS-Order (with an order constraint) asks for more computation effort to determine the spatio-temporal similarity, upper and lower bounds, and heuristic functions. Moreover, in LCSS-Order, a sample on a query trajectory may not be matched to its nearest sample on a data trajectory, which consequently requires a scan of more trajectory samples to get the best matching. Therefore, it is inevitable that LCSS-Order requires more CPU time and higher numbers of visited trajectories. Furthermore, NNT and BCT have similar structures; thus, their curves are very close in Figure 8.

7 Conclusions and Research Directions

We proposed and investigated a novel problem called Personalized Trajectory Matching (PTM). To the best of our knowledge, this is the first work that investigates the general trajectory matching problem in spatial networks, and takes the significance of trajectory sample points into account. We believe that this type of query may bring significant benefits to users in many popular applications such as route planning, carpooling, friend recommendation, traffic analysis, urban computing, and location based services in general. To address the PTM query efficiently, an adaptive two-phase PTM search algorithm was proposed. A pair of bounds were developed to prune the search space while a heuristic strategy based on priority ranking was adopted to schedule multiple query sources effectively. Finally, the performance of the PTM query was demonstrated through extensive experiments based on real and synthetic spatial data sets.

Several interesting directions for future research exist. First, it is of interest to incorporate trajectory combination into personalized trajectory recommendation to further enhance the effectiveness of the paper's proposal. Conceptually, a recommended trajectory can be composed of segments from different data trajectories. This is a challenging problem, where the difficulties lie in how to effectively and efficiently divide and combine data trajectories. Second, the sampling rate and consequent uncertainty of the data trajectories can be taken into account during PTM query processing. A probabilistic model based on moving objects patterns and corresponding algorithms are called for. Third, the effectiveness of PTM at supporting pertinent use cases can be studied. It is possible to conduct case studies, by giving visualized results to users and collecting and studying the user feedback, thus analyzing user satisfaction. Fourth, the linear combination method utilized in this work to combine the similarity between two domains can be refined. Combination methods that encompass linear and non-linear relationships between two (or more) domains can be considered. Fifth, it is of interest to extend PTM by integrating more parameters into the trajectory similarity functions (e.g., road type and road conditions), thus giving users a more suitable recommendation. Parameters like road type and road conditions can possibly be learned from the corresponding trajectory data.

References

1. R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, pages 69–84, 1993.
2. H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. In *SODA*, pages 589–598, 2003.
3. S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, pages 853–864, 2005.
4. Y. Cai and R. Ng. Indexing spatio-temporal trajectories with Chebyshev polynomials. In *SIGMOD*, pages 599–610, 2004.
5. K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.
6. L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, pages 792–803, 2004.
7. L. Chen, M. T. Ozsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.
8. Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching trajectories by locations: an efficiency study. In *SIGMOD*, pages 255–266, 2010.
9. E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Math*, 1:269–271, 1959.
10. C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.

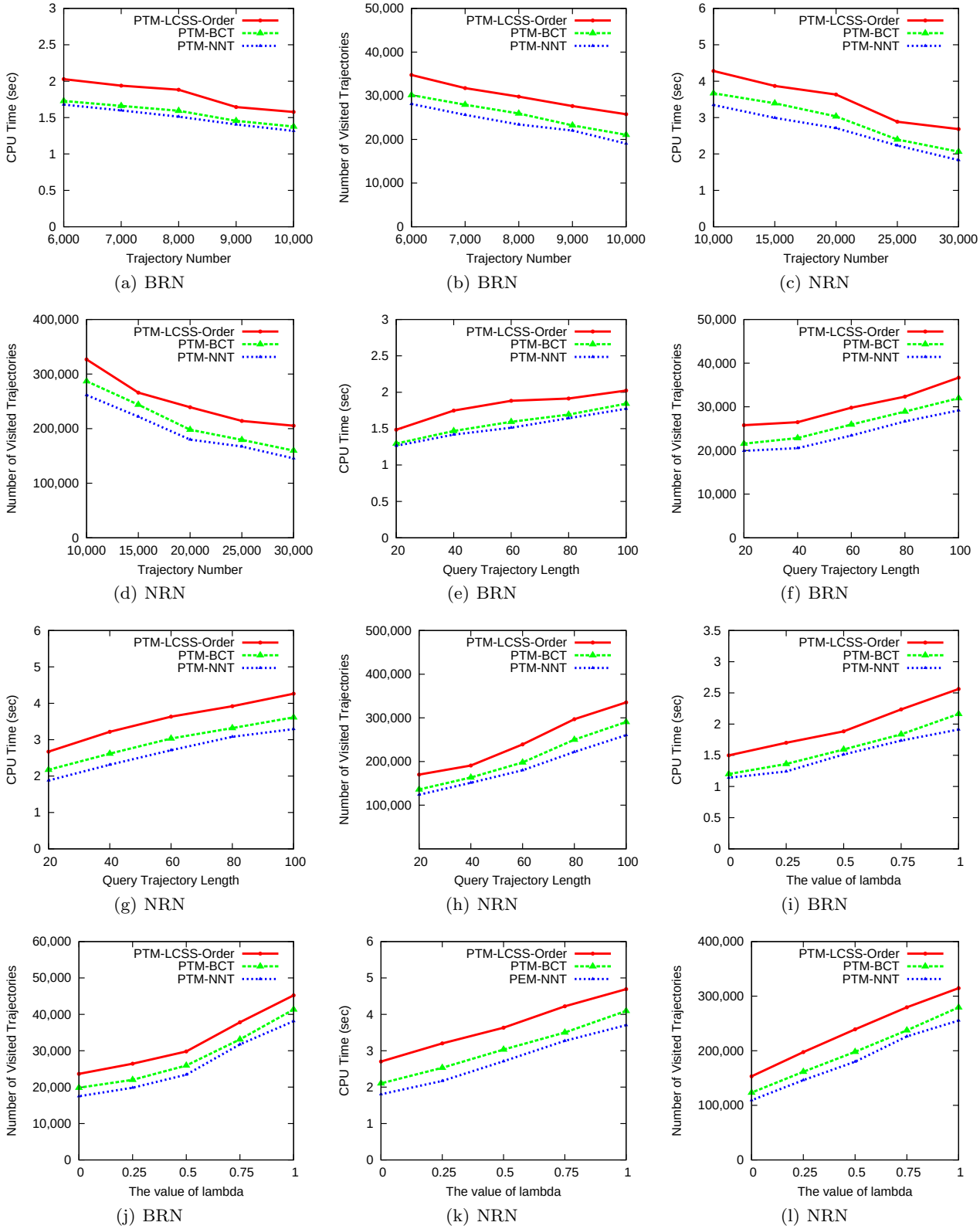

Fig. 8 Effect of Different Similarity Functions

11. E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Algorithms for nearest neighbor search on moving object trajectories. *Geoinformatica*, 11(2):159–193, 2007.
12. E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based most similar trajectory search. In *ICDE*, pages 816–825, 2007.
13. H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. Sondag. Adaptive fastest path computation on a road network: A traffic mining approach. In *VLDB*, pages 794–805, 2007.
14. J. Greenfeld. Matching GPS observations to locations on a digital map. In *81th Annual Meeting of the Transportation Research Board*, 2002.
15. A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.
16. H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. idistance: An adaptive b+-tree based indexing method for nearest neighbor search. *ACM TODS*, 30(2):364–397, 2005.
17. E. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417, 2002.
18. B. Lin and J. Su. Shapes based trajectory queries for moving objects. In *ACM GIS*, pages 21–30, 2005.
19. K. Liu, K. Deng, Z. Ding, M. Li, and X. Zhou. Moir/mt: Monitoring large-scale road network traffic in real-time. In *VLDB*, pages 1538–1541, 2009.
20. K. Liu, Y. Li, F. He, J. Xu, and Z. Ding. Effective map-matching on the most simplified road network. In *SIGSPATIAL GIS*, pages 609–612, 2012.
21. M. D. Morse and J. M. Patel. An efficient and accurate method for evaluating time series similarity. In *SIGMOD*, pages 569–580, 2007.
22. S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis. User oriented trajectory search for trip recommendation. In *EDBT*, pages 156–167, 2012.
23. R. Sherkat and D. Rafiei. On efficiently searching trajectories and archival data for historical similarities. *PVLDB*, 1(1):896–908, 2008.
24. L. A. Tang, Y. Zheng, X. Xie, J. Yuan, X. Yu, and J. Han. Retrieving k-nearest neighboring trajectories by a set of point locations. In *SSTD*, pages 223–241, 2011.
25. E. Tiakas, A. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, D. Stojanovic, and S. Djordjevic-Kajan. Searching for similar trajectories in spatial networks. *Journal of Systems and Software*, 82(5):772–788, 2009.
26. E. Tiakas, A. N. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, D. Stojanovic, and S. Djordjevic-Kajan. Trajectory similarity search in spatial networks. In *IDEAS*, pages 185–192, 2006.
27. M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684, 2002.
28. C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing globalb curve-matching algorithms. In *SSDBM*, pages 379–388, 2006.
29. Y. Yanagisawa, J. Akahani, and T. Satoh. Shape-based similarity query for trajectory of mobile objects. In *Mobile Data Management*, pages 63–77, 2003.
30. B.-K. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208, 1998.
31. P. Zarchan. Global positioning system theory and applications. In *Progress in Astronautics and Aeronautics*, volume 163, pages 1–781. American Institute of Aeronautics and Astronautics, 1996.
32. Y. Zheng, X. Xie, and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.
33. Y. Zheng and X. Zhou, editors. *Computing with Spatial Trajectories*. Springer, 2011.