User Oriented Trajectory Search for Trip Recommendation

Shuo Shang [†]   Ruogu Ding [§]   Bo Yuan [‡]   Kexin Xie[†]   Kai Zheng[†]   Panos Kalnis [§]

[†] **School of Information Technology and Electrical Engineering, The University of Queensland**
[§] **Division of Mathematical and Computer Sciences and Engineering,**
The King Abdullah University of Science and Technology
[‡] **Division of Informatics, Graduate School at Shenzhen, Tsinghua University**
[†]**{shangs,kexin,kevinz}@itee.uq.edu.au**   [§]**{ruogu.ding,panos.kalnis}@kaust.edu.sa**   [‡]**yuanb@sz.tsinghua.edu.cn**

ABSTRACT

Trajectory sharing and searching have received significant attentions in recent years. In this paper, we propose and investigate a novel problem called User Oriented Trajectory Search (UOTS) for trip recommendation. In contrast to conventional trajectory search by locations (spatial domain only), we consider both spatial and textual domains in the new UOTS query. Given a trajectory data set, the query input contains a set of intended places given by the traveler and a set of textual attributes describing the traveler's preference. If a trajectory is connecting/close to the specified query locations, and the textual attributes of the trajectory are similar to the traveler'e preference, it will be recommended to the traveler for reference. This type of queries can bring significant benefits to travelers in many popular applications such as trip planning and recommendation.

There are two challenges in the UOTS problem, (i) how to constrain the searching range in two domains and (ii) how to schedule multiple query sources effectively. To overcome the challenges and answer the UOTS query efficiently, a novel collaborative searching approach is developed. Conceptually, the UOTS query processing is conducted in the spatial and textual domains alternately. A pair of upper and lower bounds are devised to constrain the searching range in two domains. In the meantime, a heuristic searching strategy based on priority ranking is adopted for scheduling the multiple query sources, which can further reduce the searching range and enhance the query efficiency notably. Furthermore, the devised collaborative searching approach can be extended to situations where the query locations are ordered. The performance of the proposed UOTS query is verified by extensive experiments based on real and synthetic trajectory data in road networks.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Algorithms, Performance

Keywords

User Oriented Trajectory Search, Locations, Efficiency, Road Networks, Trip Recommendation

1. INTRODUCTION

The continuous proliferation of mobile devices and the rapid development of Global Positioning Systems (GPS) enable people to log their current geographic locations and share their trajectories to web sites such as Bikely[1], GPS-Waypoints[2], Share-My-Routes[3], Microsoft GeoLife[4]. In the meantime, more and more social network sites, including Twitter[5], Four-square[6] and Facebook[7], begin to support the applications of sharing locations/trajectories. The availability of such massive trajectory data creates various novel applications. An emerging one is trajectory search and recommendation, which is designed to find trajectories connecting/close to a set of query locations (e.g., a set of sightseeing places specified by the traveler) and recommend them to the traveler for reference. In existing works (e.g., [10]), the query is conducted in spatial domain only, which means that the spatial distance/similarity is considered as the sole influence factor for the query. However, in many real application scenarios, especially in modern recommendation systems, spatial distance itself is not sufficient to evaluate the relationship between trajectories and query locations, due to the particular preference of users. For example, the system may recommend a travel route with several tolled road segments, which may be unfavorable to some budget-sensitive travelers; or recommend a travel route containing off-road segments to the travelers without appropriate vehicles. Although the recommended routes are close to the query locations, it is possible that the travelers may not be fully satisfied with this trip recommendation as their preferences are not fulfilled.

Being aware of the weakness of existing trajectory search approaches, in this paper, we propose and investigate a novel

---

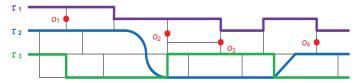[1]http://www.bikely.com/

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

[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/

	tollway	highway	off road	travel style	transport
K_q	0%	>70%	<10%	independent	by private vehicle
$K_{\tau 1}$	20%	60%	20%	grouped	by bus
$K_{\tau 2}$	0%	80%	0%	independent	by private vehicle
$K_{\tau 3}$	0%	70%	20%	grouped	by bus

Figure 1: An Example of User Oriented Trajectory Search

problem called User Oriented Trajectory Search (UOTS). Different from conventional trajectory search based on locations [10] (spatial domain only), in the User Oriented Trajectory Search, we take into account both the spatial distance between trajectories and query locations (spatial domain) and the textual attribute similarity between trajectories and the traveler's preference (textual domain). In the textual domain, not only the basic features of trajectories such as tollway, highway, off road, etc., but also the travel styles, such as independent or grouped, by bus or by private vehicle, are taken into consideration. It is reasonable to assume that travelers are likely to favor the route if their preferences are similar with the textual attributes of the route. **Remark.** The traveler's preference data are recorded as the personal information in the route-sharing sites and the textual attributes can be regarded as being part of the trajectory data. Users (e.g., travelers) only need to input their intended places (the same as [10]) to conduct the User Oriented Trajectory Search.

An example is demonstrated in Figure 1, where τ_1, τ_2, τ_3 are trajectories and $K_{\tau 1}$, $K_{\tau 2}$, $K_{\tau 3}$ are the corresponding textual attributes. $O_q = \{o_1, o_2, o_3, o_4\}$ is the query location set and K_q is the textual attributes of the user preference. If only the spatial domain is considered (e.g., [10]), it is easy to find that τ_1 is the closest trajectory to the query locations according to a distance metric (e.g., $Dist(O_q, \tau_1) = \sum_{i=1}^{i=4} Dist(o_i, \tau_1)$). However, in the textual domain, the trajectory τ_1's textual attributes (i.e., $K_{\tau 1}$) are not consistent with the user's preference (i.e., K_q). Therefore, τ_1 may not be a good trip reference for the traveler. In the meantime, we find that $K_{\tau 2}$ is matching K_q very well in the textual domain. Although τ_2 is not as good as τ_1 in the spatial domain (i.e., $Dist(O_q, \tau_2)$ is a bit greater than $Dist(O_q, \tau_1)$), we still consider τ_2 as the global best choice for trip recommendation by integrating both the spatial and textual attributes.

In this work, the proposed User Oriented Trajectory Search is applied in road networks, since in a large number of practical scenarios objects move in a constrained environment (e.g., roads, railways, rivers, etc.) rather than a free space. A trajectory is a sequence of sample points of a moving object. We assume that all sample points have already been aligned to the vertexes on the road network according to some map-matching methods [17, 2, 3, 30] and between two adjacent sample points a, b, the moving objects always follow

the shortest path connecting a, b. In the textual domain, the weight of each attribute can be calculated by the TF-IDF method [27] (for numerical features, we can simply use their original values. For values such as "by bus" and "by car", we can map them to "0" and "1" respectively. see Figure 1 for reference.) and the textual attributes K_τ of trajectory τ are transformed into a high dimensional vector. Conceptually, the textual attributes of the user preference K_q can be mapped as a point in the high dimensional space, and the textual searching process is finding the nearest data points (i.e.,vectors) to the query point K_q according to a certain distance metric.

A straightforward idea to solve the UOTS problem is called Spatial-First method. We search the trajectories close to the query locations in the spatial domain initially, and then compute the corresponding textual distances to K_q in the textual domain respectively. Through integrating the computation results in the two domains, the trajectory with the minimum spatial-textual distance (i.e., with the highest similarity) to the query input q (i.e., a set of query locations O_q and a set of textual attributes describing user preference K_q) can be found. The main drawback of the Spatial-First method is that the searching range in both spatial and textual domains can hardly be constrained (i.e., it is difficult to set a suitable stopping condition to constrain the searching range in the spatial domain, which results in a large number of trajectories in the data set to be processed). The extremely high computation cost prevents the query from being answered in real time. In addition, there is a lack of an effective scheduling strategy in the Spatial-First method for the multiple query locations, which may lead to inefficient searching effort. **Remark.** To the best of our knowledge, there is no existing method that can address the proposed UOTS problem.

To overcome the weakness of the Spatial-First method and address the UOTS problem efficiently, an adaptive collaborative searching approach is proposed. In this approach, the trajectory search is conducted in the spatial and textual domains alternately. To constrain the global searching range in the two domains, a pair of bounds (i.e., upper and lower bounds of the spatial-textual distance to q) is devised. In the meantime, a heuristic searching strategy based on priority ranking is adopted to schedule the multiple query sources (i.e., a set of query locations O_q in the spatial domain and a query point K_q in the textual domain). Conceptually, we carefully 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 with the Spatial-First method introduced above, the devised collaborative searching approach has two major advantages. First, the searching range in the two domains can be constrained into a comparatively smaller area. Second, due to the adaption of an effective heuristic searching strategy, we can avoid devoting unnecessary searching effort to the trajectories unlikely to be the optimal choice and further enhance the query efficiency.

Extension: In some practical scenarios, the traveler may specify a preferred visiting order for intended places (e.g., A, B, C are intended places and the visiting order is $A \rightarrow B \rightarrow C$). The proposed collaborative searching approach

can be further extended to address the queries with an order efficiently with the help of a series of optimization techniques.

To sum up, the main contributions of this paper are as follows:

- We define a novel type of query to find the closest trajectories according to the proposed spatial-textual distance. It provides new features for advanced spatial-temporal information systems, and benefits users in many popular applications such as trip planning and recommendation.

- We propose a series of new metrics to evaluate the spatial and textual distance/similarity.

- We devise an adaptive collaborative approach to answer the User Oriented Trajectory Search efficiently, with the support of a pair of comparably tight bounds and a heuristic scheduling strategy based on priority ranking. Furthermore, the proposed techniques can also be extended to situations where the query locations are ordered.

- We conduct extensive experiments on real and synthetic trajectory data to investigate the performance of the proposed approaches.

The rest of the paper is organized as follows. Section 2 introduces the road networks, trajectories and distance metrics used in this paper as well as problem definitions. The baseline method is introduced in Section 3, and the UOTS query processing is described in Section 4, which is followed by the experimental results in Section 5. This paper is concluded in Section 7 after discussions on related work in Section 6.

2. PRELIMINARIES

2.1 Road Networks

In this work, road networks are modeled by connected and undirected planar graphs $G(V, E)$, where V is the set of vertexes and E is the set of edges. A weight can be assigned to each edge to represent length or application specific factors such as traveling time obtained by mining the historic traffic data [16]. Given two locations a, b in a road network, the network distance between them is the length of their shortest network path (i.e., a sequence of edges linking a and b where the accumulated weight is minimal). When the weight is associated with factors such as traveling time, the lower bound of network distance is not necessarily the corresponding Euclidean distance; thus the spatial indexes such as R-tree are not effective. The data points are embedded in networks and they may be located in edges. If the network distances to the two end vertexes of an edge are known, it is straightforward to derive network distance to any point in this edge. Thus, we assume that all data points are in vertexes for the sake of clear description.

2.2 Trajectory

The raw trajectory samples obtained from GPS devices are typically of the form of $(longitude, latitude, time\text{-}stamp)$. How to map the $(longitude, latitude)$ pair onto a given road

network is an interesting research problem itself but outside the scope of this paper. We assume that all trajectory sample points have already been aligned to the vertexes on the road network by some map-matching algorithm [2, 3, 17, 30], and between two adjacent sample points a, b, the moving objects always follow the shortest path connecting a and b. As the trajectory's time-stamp attribute is not related to this work, we define the spatial attribute of a trajectory in the following format:

Definition: Trajectory
A trajectory of a moving object τ in road network G is a finite sequence of positions: $\tau = \{p_1, p_2, ..., p_n\}$, where p_i is the sample point in G, for $i = 1, 2, .., n$. □

In the meantime, every trajectory τ has a set of textual attributes K_τ, to describe its basic features, such as tollway, highway, off road, etc., and travel styles, such as independent or grouped, by bus or by private vehicle, etc. The weight of each textual attribute can be calculated by TF-IDF [27] hence K_τ is transformed into a high dimensional vector (i.e., a point in high dimensional space).

2.3 Spatial-Textual Distance Function

Given any two locations a, b in a road network, the shortest network path between them is denoted as $SP(a, b)$ and the length of $SP(a, b)$ is denoted as $sd(a, b)$. Given a trajectory τ and a data point o in a road network, the minimum distance $d_M(o, \tau)$ between data point o and trajectory τ is defined as

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

where v_i is the vertex belonging to τ.

Given a trajectory $\tau \in T_r$ and a query input q, including a set of query locations O_q and a set of user-preference attributes K_q, the spatial distance $S_{dist}(O_q, \tau)$ and textual distance $T_{dist}(K_q, K_\tau)$ are defined by the following equations. In Equation 2, m is the number of query locations. A Sigmoid function [24] is adopted here to normalize the spatial distance to the range $[0, 1]$. In Equation 3, the Jaccard distance [26] is used to measure the textual similarity, and also map the results to the range $[0, 1]$.

$$S_{dist}(O_q, \tau) = \frac{2}{1 + e^{- \sum_{i=1}^{m} d_M(o_i, \tau)}} - 1 \tag{2}$$

$$T_{dist}(K_q, K_\tau) = 1 - \frac{K_q \cdot K_\tau}{\|K_q\|^2 + \|K_\tau\|^2 - K_q \cdot K_\tau} \tag{3}$$

By combining Equation 2 and 3, the spatial-textual distance between q and τ is defined as

$$ST_{dist}(q, \tau) = \lambda \cdot S_{dist}(O_q, \tau) + (1 - \lambda) \cdot T_{dist}(K_q, K_\tau) \tag{4}$$

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

The distances (i.e., spatial distance, textual distance and spatial-textual distance) defined above are used to evaluate the similarity between two objects, and a lower value of distance means a higher similarity.
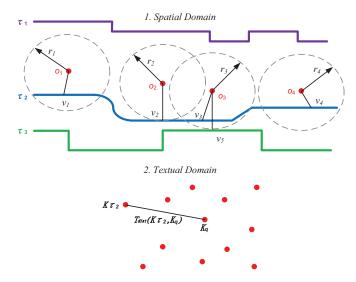
τ₁, τ₂, τ₃ labels with spatial domain showing o₁, o₂, o₃, o₄ and r₁, r₂, r₃, r₄ and v₁, v₂, v₃, v₄, v₅

1. Spatial Domain

2. Textual Domain

$K\tau_2$, $T_{dist}(K\tau_2, K_q)$, K_q

Figure 2: An Example of the Spatial First Search

2.4 Problem Definition

Given a trajectory set T_r, a query input q, including a location set O_q and a textual attribute set K_q, User Oriented Trajectory Search (UOTS) finds the trajectory $\tau \in T_r$ with the minimum value of $ST_{dist}(q, \tau)$, such that $ST_{dist}(q, \tau) \leq ST_{dist}(q, \tau')$, $\forall \tau' \in T_r \setminus \tau$. $\qquad \square$

3. BASELINE METHOD

In this section, we introduce the baseline method adopted in this work. "Spatial-First" is a straightforward idea to address the UOTS problem. Given a trajectory data-set T_r and a query input q (including a set of query points O_q and a set of textual attributes K_q), the proposed Spatial-First approach includes two steps. First, we browse the road network and find the trajectories close to the query locations in the spatial domain. Second, for each browsed trajectory τ, we compute the corresponding textual distance $T_{dist}(K_\tau, K_q)$ respectively. Through integrating the computation results in both spatial and textual domains, the trajectory with the minimum spatial-textual distance to q can be found.

Consider the schematic example demonstrated in Figure 2. $O_q = \{o_1, o_2, o_3, o_4\}$ is the set of query points and τ_1, τ_2, τ_3 are trajectories. $\{v_1, v_2, v_3, v_4\} \in \tau_2$ are the closest vertexes to o_1, o_2, o_3, o_4 respectively, and $v_5 \in \tau_3$ is the closest vertex to o_3. To browse the road network and find the trajectories close to the query locations, Dijkstra's expansion [12] is adopted here. From each query point $o_i \in O_q$, a browsing wavefront is expanded by Dijkstra's algorithm. The browsing speeds from different query points are the same. Conceptually, the browsed region is restricted as a circle as shown in Figure 2, where the radius is the shortest network distance from the center o_i to the browsing wavefront, denoted as $r_i, i \in [1, 4]$. If a vertex $v \in \tau$ is the first vertex scanned by the expansion wavefront from o_i, v is just the closest vertex to o_i. That is $d_M(o_i, \tau) = sd(o_i, v)$. For example, v_5 is the closest vertex to o_3 in τ_3 and $d_M(o_3, \tau_3) = sd(o_3, v_5)$. Once a trajectory τ has been scanned by the expansion wavefronts from every query location $o_i \in O_q, i \in [1, 4]$, such as τ_2 in Figure 2, we can obtain the values of $d_M(o_i, \tau), i \in [1, 4]$ and

compute the spatial distance $S_{dist}(O_q, \tau)$ between trajectory τ and query points O_q according to Equation 2: e.g.,

$$S_{dist}(O_q, \tau_2) = \frac{2}{1 + e^{-\sum_{i=1}^{4} d_M(o_i, \tau_2)}} - 1$$

$$= \frac{2}{1 + e^{-(sd(o_1, v_2) + sd(o_2, v_2) + sd(o_3, v_3) + sd(o_4, v_4))}} - 1$$

This type of trajectories (e.g., τ_2) is denoted as "fully scanned trajectory" in this section. Then, we map the corresponding textual attributes K_τ to the high dimension space and calculate the textual distance $T_{dist}(K_q, K_\tau)$ by Equation 3. Finally, through integrating the values of $S_{dist}(O_q, \tau)$ and $T_{dist}(K_q, K_\tau)$ according to Equation 4, the spatial-textual distance $ST_{dist}(q, \tau)$ is found.

To constrain the searching range in the spatial domain, a pair of upper and lower bounds of the spatial-textual distance $ST_{dist}(q, \tau)$ is proposed. If the lower bound of trajectory τ is greater than another trajectory's upper bound, τ must not be the trajectory with the minimum spatial-textual distance to q and can be pruned safely. Among all trajectories fully scanned by the searching approach stated above (e.g., τ_2 in Figure 2), we define a global upper bound UB as

$$UB = \min_{\forall \tau \in T_s} \{ST_{dist}(q, \tau)\} \qquad (5)$$

where T_s is the set of fully scanned trajectories. Obviously, UB is a dynamic value, and continuously updated during the searching process. In the following paragraphs, we introduce our method to estimate the lower bound of $ST_{dist}(q, \tau)$. (i.e., τ is a trajectory which has not been fully scanned, such as τ_1 and τ_3 in Figure 2. In particular, trajectories such as τ_1 are denoted as "unscanned" trajectory and trajectories such as τ_3 is denoted as "partly scanned" trajectory.) Since Dijkstra's algorithm always chooses the vertex with the smallest distance label for expansion, if a trajectory τ has not been scanned by the expansion wavefront from o_i, we have $d_M(o_i, \tau) > r_i$. The radius r_i is the network distance from center o_i to the current expansion wavefront(e.g., $d_M(o_1, \tau_3) > r_1$, $d_M(o_2, \tau_3) > r_2$ and $d_M(o_4, \tau_3) > r_4$). Thus,

$$\sum_{i=1}^{m} d_M(o_i, \tau) > \sum_{o_x \in O_t} d_M(o_x, \tau) + \sum_{o_y \in O_n} r_y \qquad (6)$$

where m is the size of query location set O_q and O_t is the set of locations whose expansion wavefronts have scanned τ and O_n is the set of location whose expansion waves have not scanned τ. $O_t \cup O_n = O_q$. For instance, in Figure 2, $\sum_{o_x \in O_t} d_M(o_x, \tau_3) = d_M(o_3, \tau_3)$ and $\sum_{o_y \in O_n} r_y = r_1 + r_2 + r_4$. Obviously, we have

$$\begin{cases} d_M(o_1, \tau_3) > r_1 \\ d_M(o_2, \tau_3) > r_2 \\ d_M(o_4, \tau_3) > r_4 \end{cases} \Rightarrow \sum_{i=1}^{4} d_M(o_i, \tau_3) > d_M(o_3, \tau_3) + r_1 + r_2 + r_4$$

According to Equation 6, we can use $(\sum_{o_x \in O_t} d_M(o_x, \tau) + \sum_{o_y \in O_n} r_y)$ to replace $(\sum_{i=1}^{m} d_M(o_i, \tau))$ in Equation 2 and have

$$S_{dist}(O_q, \tau) = \frac{2}{1 + e^{-\sum_{i=1}^{m} d_M(o_i, \tau)}} - 1$$

$$> \frac{2}{1 + e^{-(\sum_{o_x \in O_t} d_M(o_x, \tau) + \sum_{o_y \in O_n} r_y)}} - 1$$

$$S_{dist}(O_q, \tau).lb = \frac{2}{1 + e^{-(\sum_{o_x \in O_t} d_M(o_x, \tau) + \sum_{o_y \in O_n} r_y)}} - 1 \tag{7}$$

Since $ST_{dist}(q, \tau) = \lambda \times S_{dist}(O_q, \tau) + (1 - \lambda) \times T_{dist}(K_q, K_\tau)$ (i.e., Equation 4) and $T_{dist}(K_q, K_\tau) \geq 0$, the lower bound of trajectory τ is estimated as

$$ST_{dist}(q, \tau) > \lambda \cdot S_{dist}(O_q, \tau) > \lambda \cdot S_{dist}(O_q, \tau).lb$$

$$ST_{dist}(q, \tau).lb = \frac{2\lambda}{1 + e^{-(\sum_{o_x \in O_t} d_M(o_x, \tau) + \sum_{o_y \in O_n} r_y)}} - \lambda \tag{8}$$

Among all trajectories which have not been fully scanned, a global lower bound LB is defined as

$$LB = \min_{\forall \tau \in T_n} \{ST_{dist}(q, \tau).lb\} \tag{9}$$

where T_n is the set of trajectories that have not been fully scanned, and $T_n = T_r - T_s$. Similar to UB, LB is also a dynamic value and continuously updated during the searching process. **Remark.** To reduce the computation and storage load, we only compute and maintain the partly scanned trajectories' lower bounds (e.g., τ_3 in Figure 2). For other trajectories completely outside the browsed region (e.g., τ_1 in Figure 2), they must not have the lower bounds less than the partly scanned trajectories's lower bounds. For example, $ST_{dist}(q, \tau_1).lb = \frac{2\lambda}{1 + e^{-\sum_{i=1}^{4} r_i}} - \lambda$ and $ST_{dist}(q, \tau_3).lb = \frac{2\lambda}{1 + e^{-(d_M(o_3, \tau_3) + r_1 + r_2 + r_4)}} - \lambda$. Thus, $ST_{dist}(q, \tau_1).lb$ is greater than $ST_{dist}(q, \tau_3).lb$. The expansion of browsing wavefronts will be stopped once LB is greater than the global upper bound UB. All trajectories that have not been fully scanned, including partly scanned trajectories and the trajectories completely outside the browsed region, can be pruned safely. The current value of UB is just the minimum spatial-textual distance to q and the corresponding trajectory τ will be recommended to the user for reference. The searching process of the Spatial-First method is described as Algorithm 1.

In Algorithm 1, from each query point $o_i \in O_q$, browsing wavefronts are expanded in turn (*line* 3). The search process is according to the Dijkstra's algorithm [12], which always selects the vertex with the minimum distance label for expansion, and the selected vertex is denoted as v (*line* 4). Then, all trajectories passing through vertex v are checked (*line* 5). If trajectory τ has not been scanned by the expansion wavefront from o_i before, τ will be labeled as being scanned by o_i and the corresponding $S_{dist}(O_q, \tau).lb$ will be updated (*line* 6-8). Among all the partly scanned trajectories, the one with the minimum value of $\lambda \cdot S_{dist}(O_q, \tau).lb$ will be selected as the global lower bound LB (i.e., Equation 9) (*line* 9). Once we find a trajectory τ that has been fully scanned by the expansion wavefronts from every query point $o \in O_q$, we can calculate the value of $ST_{dist}(q, \tau)$ according to Equation 4. If the value of $ST_{dist}(q, \tau)$ is less than the global upper bound UB, the value of UB will be replaced by the value of $ST_{dist}(q, \tau)$ (*line* 10-13). When the value of LB is greater than UB, the query processing is terminated.

The trajectory τ with the minimum value of $ST(q, \tau)$ (i.e., UB) will be returned (*line* 14-15).

Algorithm 1: Spatial-First Trajectory Search

Data: T_r, q
Result: $\min_{\tau \in T_r} ST_{dist}(q, \tau)$

1 $LB \leftarrow +\infty; UB \leftarrow +\infty;$
2 **while** *true* **do**
3   **for** *each $o_i \in O_q$* **do**
4          $v \leftarrow Expand(o_i);$
5   **for** *each trajectory $\tau \in v.trajList$* **do**
6   **if** $\tau.scan(o_i) = false$ **then**
7                  $\tau.scan(o_i) \leftarrow true;$
8   Update $S_{dist}(O_q, \tau).lb;$
9                  $LB \leftarrow \min\{\lambda \cdot S_{dist}(O_q, \tau).lb\};$
10   **if** $\tau.scan(o)$ *is true*, $\forall o \in O_q$ **then**
11   Calculate $ST_{dist}(q, \tau);$
12   **if** $ST_{dist}(q, \tau) < UB$ **then**
13                 $UB \leftarrow ST_{dist}(q, \tau);$
14   **if** $LB > UB$ **then**
15   **return** UB *and the corresponding τ;*

---

4. UOTS QUERY PROCESSING

The main weakness of the Spatial-First approach introduced in Section 3 is that the searching range in both spatial and textual domains can hardly be constrained. The lower bound (Equation 8) is loose since only the spatial domain is considered and the influence of the textual domain is totally ignored. The parameter λ is used to adjust the relative importance of spatial proximity factor and textual similarity factor (Equation 4). The smaller the value of λ, the less important the spatial domain, and even looser the lower bound. The loose lower bound results in poor pruning effectiveness and a large number of trajectories in the data-set have to be processed. The extremely high computation cost may prevent the query from being answered in real time.

To overcome the weakness of the Spatial-First method and address the UOTS query efficiently, an adaptive collaborative searching approach is proposed in this section. In this approach, the trajectory search is conducted in the spatial and textual domains alternately. i.e., searching the closest trajectories in the spatial domain while searching the nearest data points (vectors) in the textual domain. The main component of this section can be divided into the following three parts. First, we introduce the concept of the collaborative searching approach and propose a comparably tight lower bound to tighten the searching range in both spatial and textual domains (Section 4.1). Second, we propose a heuristic searching strategy based on priority ranking to schedule the multiple query sources including query locations in the spatial domain and the query point in the textual domain, to avoid devoting unnecessary searching effort to the trajectories unlikely to be the optimal choice and further enhance the query efficiency (Section 4.2). Third, we extend the collaborative searching approach to situations where the query locations are ordered. (Section 4.3).
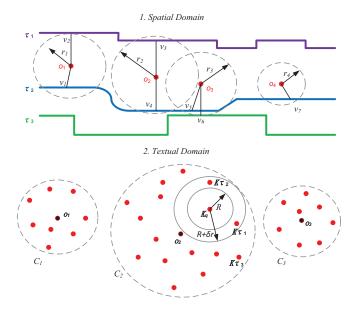
Figure 3: An Example of the Collaborative Trajectory Search

4.1 Collaborative Searching Approach

In the collaborative searching approach, the trajectory search is conducted in the spatial and textual domain alternately. In the spatial domain, similar to the baseline method, Dijkstra's expansion [12] is adopted to browse the road network and find the nearest trajectories. On the other hand, we use the iDistance indexing method [21] in the textual domain to index the data points (i.e., high dimensional vectors describing the textual attributes). Note that the utilization of the well known iDistance indexing is only for browsing distance between high dimensional vectors. Other spatial indexes can also be easily adapted. In iDistance, a data partition/clustering method (e.g., k-means, k-medoids, etc.) is conducted in the first place, to group the data points into k clusters based on their distribution. For each cluster $C_i, i \in [1, k]$, a reference point m_i is selected. Then, we compute and record the textual distance (Equation 3) between m_i and every data point $p \in C_i$. A B+ tree is adopted to index the data points using the textual distance to the corresponding reference point as a key. To find the data points close to the query point K_q, we browse the high dimensional space by expanding the radius R of a hyper-sphere centered at K_q. We use Δr as the initial searching radius (i.e., $R = \Delta r$), and the search radius is increased by Δr (i.e., $R = R + \Delta r$), step by step, to form a larger searching sphere, until the target points are found. In our implementation, to find the suitable values for the cluster number k and the initial radius δr, and achieve a good performance, we conducted extensive experiments when establishing the iDistance index (the same as the approaches introduced in [20]).

An example is demonstrated in Figure 3. In the spatial domain, τ_1, τ_2, τ_3 are trajectories and o_1, o_2, o_3, o_4 are query locations. Similar to the Spatial-First method specified in Section 3, from each query point $o_i, i \in [1, 4]$, a browsing wavefront is expanded according to Dijkstra's algorithm

[12] to find the trajectories close to the query points, and $r_i, i \in [1, 4]$ is the radius of the corresponding expansion circle range. To further constrain the searching range and achieve a higher efficiency, a heuristic method is employed here to schedule the network expansion from multiple query points, which is the main difference from the baseline method. The details will be introduced in Section 4.2. If a vertex $v \in \tau$ has been scanned by the expansion wavefront from $o_i, i \in [1, 4]$, v is just the closest vertex to o_i in τ (i.e., $d_M(o_i, \tau) = sd(o_i, v)$). In this example, $\{v_1, v_4, v_5, v_7\} \in \tau_2$ are the closest vertexes to o_1, o_2, o_3, o_4 respectively, and $\{v_2, v_3\} \in \tau_1$ are the closest vertexes to o_2, o_3 respectively. $v_6 \in \tau_3$ is the closest vertex to o_3. If a trajectory τ has been scanned by the expansion wavefronts from every query point $o_i \in O_q$ (e.g., τ_2), its spatial distance to O_q can be obtained based on Equation 2. This kind of trajectories is denoted as "fully scanned in spatial" in this section. For a trajectory τ that has not been fully scanned in spatial (e.g., τ_1, τ_3, in particular, this type of trajectories is denoted as "partly scanned in spatial"), its spatial distance to the query points O_q can be estimated by a lower bound $S_{dist}(O_q, \tau).lb$, calculated by Equation 7.

In the textual domain, all data points have been indexed according to the iDistance method [21]. o_1, o_2, o_3 are the reference points of clusters C_1, C_2, C_3 respectively. To find the closest data points, we browse the space by expanding the sphere centered at query point K_q, and R is the radius of the corresponding searching sphere. At each time, R is increased by Δr, (i.e., $R = R + \Delta r$). If a data point K_τ is inside the searching sphere, its textual distance $T_{dist}(K_q, K_\tau)$ to K_q can be retrieved easily. Otherwise, its textual distance to K_q can be estimated by a lower bound

$$T_{dist}(K_q, K_\tau).lb = R \qquad (10)$$

By combining Equation 7 and Equation 10, the lower bound of $ST_{dist}(q, \tau)$ is given as

$$ST_{dist}(q, \tau).lb = \begin{cases} \lambda \cdot S_{dist}(O_q, \tau) + (1 - \lambda)T_{dist}(K_q, K_\tau).lb & 1) \\ \lambda \cdot S_{dist}(O_q, \tau).lb + (1 - \lambda)T_{dist}(K_q, K_\tau) & 2) \\ \lambda \cdot S_{dist}(O_q, \tau).lb + (1 - \lambda)T_{dist}(K_q, K_\tau).lb & 3) \end{cases}$$
$$(11)$$

1. trajectory τ is fully scanned in the spatial domain but unscanned in the textual domain.

2. trajectory τ is partly scanned in the spatial domain and fully scanned in the textual domain.

3. trajectory τ is partly scanned in the spatial domain and unscanned in the textual domain.

Based on Equation 9, the global lower bound LB can be calculated. In the meantime, if a trajectory τ is fully scanned in both spatial and textual domains, we can obtain the value of $ST_{dist}(q, \tau)$. Among all fully scanned trajectories in two domains, the global upper bound UB can be calculated according to Equation 5. The searching stop criteria in both spatial and textual domains is whether the maximum lower bound is greater than the minimum lower bound (i.e., $LB > UB$). By integrating the computation results (i.e., the spatial-textual distance of the trajectories that have been fully scanned in

both domains), the trajectory with the minimum spatial-textual distance to the query q can be found and recommended to the user.

4.2 Heuristic Trajectory Search

In this section, we introduce a heuristic scheduling strategy based on priority ranking for multiple query sources (i.e., a set of query locations O_q in the spatial domain and a query point K_q in the textual domain) in the collaborative searching approach. A competent scheduling strategy is able to avoid devoting unnecessary searching effort to the trajectories unlikely to be the optimal choice and further enhance the query efficiency.

Consider the scenario demonstrated in Figure 3. In the spatial domain, trajectory τ_2 has been fully scanned (i.e., scanned by the expansion wavefronts from every query point $o_i \in O_q$), while trajectory τ_1, τ_3 are both partly scanned (τ_1 is only scanned by the expansion wavefronts from o_1 and o_2, and τ_3 is only scanned by the expansion wavefront from o_3). In the textual domain, the radius of the searching sphere is R and data points $K_{\tau 1}, K_{\tau 2}, K_{\tau 3}$ are not contained by the current searching sphere. Each query point $o_i \in O_q$ is given a label $label_S(o_i)$ to describe its priority. We carefully maintain a dynamic priority heap containing these query points and the point with the maximum label will be put on the top of the priority heap. At each time, we select the point from the top of the heap, and expand the corresponding wavefront, until its position (i.e., heap top) is replaced by another point, and we will expand the expansion wavefront from the new top-ranked point. The label of each query point $o_i \in O_q$ is defined as the following equation.

$$label_S(o_i) = \sum_{m \in \{T_p - T_t(i)\}} e^{-S_{dist}(O_q, \tau_m).lb} \qquad (12)$$

where T_p is the set of all partly scanned trajectories in the spatial domain (e.g., τ_1, τ_3), and $T_t(i)$ is the set of trajectories scanned by the expansion wavefront from o_i (e.g., $T_t(1) = \{\tau_1, \tau_2\}$, $T_t(3) = \{\tau_2, \tau_3\}$, etc.). The fully scanned trajectories (e.g., τ_2) and all unscanned trajectories (i.e., the trajectories that have not been scanned by any expansion wavefronts in the spatial domain) are not taken into consideration in this ranking model.

For partly scanned trajectories, our target is to transform them into fully scanned trajectories as soon as possible. To be a fully scanned trajectory, a trajectory should be scanned by the expansion wavefronts from every query point $o_i \in O_q$. Thus, for each query point o_i, its priority should be directly proportional to its "margin" (i.e., the size of $T_p - T_t(i)$). For example, in Figure 3, $T_p = \{\tau_1, \tau_3\}$, $T_t(1) = \{\tau_1, \tau_2\}$, and $T_p - T_t(1) = \{\tau_1\}$. As a result, the "margin" of o_1 is 1. On the other hand, $T_t(4) = \{\tau_2\}$ and $T_p - T_t(4) = \{\tau_1, \tau_3\}$. Hence the "margin" of o_4 is 2. Compared to o_1, o_4 should have a higher priority. In the meantime, $S_{dist}(O_q, \tau).lb$ is used to estimate the spatial distance between O_q and τ. Intuitively, a trajectory τ with the smallest value of $S_{dist}(O_q, \tau).lb$ has the highest probability to be the closest trajectory to O_q. If $\tau \in T_p - T_t(i)$, the value of $S_{dist}(O_q, \tau).lb$ should be inversely proportional to the priority of o_i. Through considering the two reasons stated above, the query point priority is defined as Equation 12.

In the above, we only consider the situations in which there is no trajectory that has been scanned in both spatial and textual domains (in the textual domain, the searching sphere is expanded as $R = R + \Delta r$). Once a trajectory has been scanned in both domains, a new ranking method to evaluate the priority of each query source $s_i \in S$ (i.e., $S = O_q \cup \{K_q\}$) is required, and the labels can be calculated as

$$label_{ST}(s_i) = \sum_{m \in \{T_p - T_t(i)\}} e^{-ST_{dist}(q, \tau_m).lb} \qquad (13)$$

where T_p is the set of partly scanned trajectories in both spatial and textual domains (e.g., τ_1, τ_3), and $T_t(i)$ is the set of trajectories scanned by the expansion wavefront/searching sphere from query source s_i (e.g., when $s_i = K_q$, $T_t(i) = \{\tau_1, \tau_2\}$; when $s_i = o_1$, $T_t(i) = \{\tau_1, \tau_2\}$). The fully scanned trajectories (e.g., τ_2) and all unscanned trajectories (i.e., the trajectories that have not been scanned by the expansion wavefronts/searching sphere in any of the two domains) are not taken into consideration in this ranking model. In contrast to the spatial priority ranking method (Equation 12) stated above, $S_{dist}(O_q, \tau).lb$ is replaced by $ST_{dist}(q, \tau_m).lb$ in the new model. The trajectory τ with the smallest value of $ST_{dist}(q, \tau).lb$ has the highest probability to be the query result (i.e., the one with the minimum value of $ST_{dist}(q, \tau)$).

The complete procedure of the collaborative searching method is described in Algorithm 2. Initially, the values of $label_S(o_i)$, $\forall o_i \in O_q$ and $label_{ST}(s_i), \forall s_i \in \{O_q \cup K_q\}$ are set to 0 (*line* 2). In the first searching phase (i.e., there is no trajectory that has been scanned in any of the two domains), the query point with the maximum $label_S(o_i)$ is selected as the expansion center Ec (*line* 3-4) and the expansion wavefront is expanded from Ec (*line* 6). Each trajectory τ passing through v will be checked. If τ has not been scanned by the expansion wavefront from o_i, τ will be labeled as being scanned by o_i. In the meantime, the corresponding $ST_{dist}(q, \tau).lb$ and all labels need to be updated (*line* 7-10). If there exists a query point $o_j \in O_q$ and the value of $label_S(o_j)$ is greater than $label_S(o_i)$, the expansion center will be replaced by o_j and the expansion wavefront from o_i will be terminated (*line* 11-12). In the textual domain, the searching sphere is expanded as $R = R + \Delta r$. Every scanned keyword set K_τ is labeled as $\tau.scan(K_q)$, and the value of $ST_{dist}(q, \tau).lb$ and all labels are updated (*line* 13-16). Once a trajectory has been scanned in both spatial and textual domains, the first searching phase terminates (*line* 17-18).

In the second searching phase, the query source with the maximum $label_{ST}(s_i)$ is selected as the expansion center Ec (*line* 19-20). If $s_i = o_i, o_i \in O_q$, the expansion wavefront is expanded using Dijkstra's algorithm. Otherwise, $s_i = K_q$, the searching sphere is expanded as $R = R + \Delta r$. Every scanned trajectory τ is labeled as $\tau.scan(s_i) = true$, and the corresponding $ST_{dist}(q, \tau).lb$, all labels $label_{ST}$ and LB need to be updated (*line* 23-27). If there exists a query source $s_j \in \{O_q \cup K_q\}$ and $label_{ST}(s_j) > label_{ST}(s_i)$, s_j will replace s_i as the new expansion center and the expansion search from s_i terminates. When a trajectory is scanned by all the query sources, the value of $ST_{dist}(q, \tau)$ can be calculated and UB need to be updated (*line* 28-31). Once the value of LB is greater than UB, the trajectory τ with the minimum value of $ST_{dist}(q, \tau)$ (i.e., UB) is returned and the search process terminates (*line* 32-35).

Algorithm 2: Collaborative Trajectory Search

Data: T_r, q
Result: $\min_{\tau \in T_r} ST_{dist}(q, \tau)$

1   $LB \leftarrow +\infty; UB \leftarrow +\infty; R \leftarrow 0$;
2   $label_S(o_i) = 0, \forall o_i \in O_q$; $label_{ST}(s_i) = 0, \forall s_i \in \{O_q \cup K_q\}$;
3   Select $o_i \in O_q$ with the maximum $label_S(o_i)$;
4   $Ec \leftarrow o_i$;
5   **while** *true* **do**
6      $v \leftarrow Expand(Ec)$;
7   **for** *each trajectory* $\tau \in v.trajList$ **do**
8   **if** $\tau.scan(o_i) = false$ **then**
9            $\tau.scan(o_i) \leftarrow true$;
10   Update $ST_{dist}(q, \tau).lb$ and all labels;
11   **if** $\exists label_S(o_j) > label_S(o_i), o_j \in O_q$ **then**
12         $Ec \leftarrow o_j$;
13      $Expand(K_q)$; //$R \leftarrow R + \Delta r$
14   **for** *each scanned data point* K_τ **do**
15         $\tau.scan(K_q) \leftarrow true$;
16   Update $ST_{dist}(q, \tau).lb$ and all labels;
17   **if** $\exists \tau.scan(o) = true, o \in O_q$ **then**
18   Break;

19   Select $s_i \in \{O_q \cup K_q\}$ with the maximum $label_{ST}(o_i)$;
20   $Ec \leftarrow s_i$;
21   **while** *true* **do**
22      $Expand(Ec)$;
23   **for** *each scanned trajectory* τ **do**
24   **if** $\tau.scan(s_i) = false$ **then**
25            $\tau.scan(s_i) \leftarrow true$;
26   Update $ST_{dist}(q, \tau).lb$ and all $label_{ST}$;
27            $LB = \min\{ST_{dist}(q, \tau).lb\}$;
28   **if** $\exists label_{ST}(s_j) > label_{ST}(s_i), s_j \in \{O_q \cup K_q\}$ **then**
29         $Ec \leftarrow s_j$;
30   **if** $\tau.scan(s)$ *is true,* $\forall s \in \{O_q \cup K_q\}$ **then**
31   Calculate $ST_{dist}(q, \tau)$;
32   **if** $ST_{dist}(q, \tau) < UB$ **then**
33            $UB \leftarrow ST_{dist}(q, \tau)$;
34   **if** $LB > UB$ **then**
35   **return** UB and the corresponding τ;

4.3   Extension to Queries with an Order

In some practical scenarios, the user may specify a preferred visiting order for the intended places. In that case, the order of a trajectory needs to be taken into consideration. In this section, the proposed collaborative searching algorithm is extended to situations where the query locations are ordered. Given a sequence of query locations $O_q = \{o_1, o_2, ..., o_m\}$, and a trajectory $\tau = \{v_1, v_2, ..., v_n\}$, the spatial similarity between O_q and τ is defined in a recursive way as

$$S_{sim}^o(O_q, \tau) = \max \begin{cases} e^{-sd(O_q.head, \tau.head)} + S_{sim}^o(O_q.rest, \tau) \\ S_{sim}^o(O_q, \tau.rest) \end{cases}$$
$$(14)$$

where $*.head$ is the head point of $*$, (e.g., $O_q.head = o_1$ and $\tau.head = v_1$) and $*.rest$ indicates the points after the head point (e.g., $O_q.rest = \{o_2, o_3, ..., o_m\}$ and $\tau.rest = v_2, v_3, ..., v_n$). This function is an extension of the similarity function proposed in [10] (i.e., extended to spatial networks

from Euclidean space). Intuitively, in the spatial domain, the higher the similarity, the less the spatial distance. Based on the spatial similarity function, the spatial distance between ordered query points O_q and trajectory τ is defined as follows

$$S_{dist}^o(O_q, \tau) = \frac{1}{1 + S_{sim}^o(O_q, \tau)} \qquad (15)$$

In Equation 15, the value of $S_{dist}^o(O_q, \tau)$ is normalized to range $[0, 1]$.

To obtain the exact spatial distance between O_q and τ, it is necessary to compute the network distance between every $o_i \in O_q$ and every $v_i \in \tau$ (i.e., every point $v_i \in \tau$ should be scanned by the browsing wavefronts expanded from every $o_i \in O_q$). This type of trajectories is denoted as "fully scanned in spatial". Other trajectories can only be called as "partly scanned in spatial" (i.e., part of points in τ have been scanned in the spatial domain) or "unscanned in spatial" (i.e., no point in τ has been scanned in the spatial domain). According to the collaborative searching method stated above (i.e., Section 4.1), from each query point $o_i \in O_q$, a browsing wavefront is expanded using Dijkstra' algorithm [12]. Conceptually, the browsed region is constrained within a circle centered at o_q (as shown in Figure 3), whose radius r_i is defined as the network distance from center o_i to the expansion wavefront. For a partly scanned trajectory τ (e.g., τ_1, τ_2, τ_3 in Figure 3), its spatial distance lower bound $S_{dist}^o(O_q, \tau).lb$ can be computed as follows.

If $v \in \tau$ has been scanned by the browsing wavefront from $o_i \in O_q$, the network distance $sd(v, o_i)$ between v and o_i can be acquired easily. Otherwise, we can use the expansion range's radius r_i to estimate the lower bound of $sd(v, o_i)$ (i.e., $sd(v, o_i) > r_i$), since Dijkstra's algorithm always selects the vertex with the minimum distance label for expansion. In Equation 14, suppose $O_q.head = o_i$ and $\tau.head = v_i$. If the value of $sd(o_i, v_i)$ cannot be obtained, it will be replaced by r_i and the spatial similarity upper bound $S_{sim}^o(O_q, \tau).ub$ can be computed as:

$$sd(v, o_i) > r_i \Rightarrow e^{-sd(v, o_i)} < e^{-r_i}$$

$$S_{sim}^o(O_q, \tau) = \max \begin{cases} e^{-sd(O_q.head, \tau.head)} + S_{sim}^o(O_q.rest, \tau) \\ S_{sim}^o(O_q, \tau.rest) \end{cases}$$

$$\leq \max \begin{cases} \alpha \\ S_{sim}^o(O_q, \tau.rest) \end{cases} = S_{sim}^o(O_q, \tau).ub$$

$$\alpha = \begin{cases} e^{-sd(O_q.head, \tau.head)} + S_{sim}^o(O_q.rest, \tau) & 1) \\ e^{-r_i} + S_{sim}^o(O_q.rest, \tau) & 2) \end{cases}$$

1. the value of $-sd(O_q.head, \tau.head)$ is available.

2. the value of $-sd(O_q.head, \tau.head)$ is not available and replaced by the value of r_i.

Based on Equation 15, the value of $S_{dist}^o(O_q, \tau)$ is inversely proportional to that of $S_{sim}^o(O_q, \tau)$. We can use $S_{sim}^o(O_q, \tau).ub$ to replace $S_{sim}^o(O_q, \tau)$ and get the spatial distance lower

bound $S_{dist}^o(O_q, \tau).lb$.

$$S_{dist}^o(O_q, \tau) = \frac{1}{1 + S_{sim}^o(O_q, \tau)}$$

$$\geq \frac{1}{1 + S_{sim}^o(O_q, \tau).ub}$$

$$S_{dist}^o(O_q, \tau).lb = \frac{1}{1 + S_{sim}^o(O_q, \tau).ub} \quad (16)$$

In the textual domain, the search process is the same as the collaborative searching approach introduced in Section 4.1. The textual distance lower bound $T_{dist}(K_q, K_\tau).lb$ can be computed by Equation 10. Hence, the spatial-textual distance lower bound for queries with an order is defined as

$$ST_{dist}^o(q, \tau) = \begin{cases} \lambda \cdot S_{dist}^o(O_q, \tau) + (1-\lambda)T_{dist}(K_q, K_\tau).lb & 1) \\ \lambda \cdot S_{dist}^o(O_q, \tau).lb + (1-\lambda)T_{dist}(K_q, K_\tau) & 2) \\ \lambda \cdot S_{dist}^o(O_q, \tau).lb + (1-\lambda)T_{dist}(K_q, K_\tau).lb & 3) \end{cases}$$
$$(17)$$

1. trajectory τ is fully scanned in the spatial domain but unscanned in the textual domain.

2. trajectory τ is partly scanned in the spatial domain and fully scanned in the textual domain.

3. trajectory τ is partly scanned in the spatial domain and unscanned in the textual domain.

According to Equation 9 and Equation 5, the global lower bound LB and upper bound UB can be found. Then, based on Equation 12 and Equation 13, the spatial priority label $label_S(o_i), o_i \in O_q$ and spatial textual priority label $label_S(s_i), s_i \in \{O_q, K_q\}$ can be identified.

$$label_S^o(o_i) = \sum_{m \in \{T_p - T_t(i)\}} e^{-S_{dist}^o(O_q, \tau_m).lb} \quad (18)$$

where T_p is the set of all partly scanned trajectories in the spatial domain, and $T_t(i)$ is the set of trajectories fully scanned by the expansion wavefront from o_i. (i.e., every vertex $v \in \tau$ has been scanned by the expansion wavefront from o_i.)

$$label_{ST}^o(s_i) = \sum_{m \in \{T_p - T_t(i)\}} e^{-ST_{dist}^o(q, \tau_m).lb} \quad (19)$$

where T_p is the set of partly scanned trajectories in both spatial and textual domains (e.g., τ_1, τ_2, τ_3), and $T_t(i)$ is the set of trajectories scanned by the expansion wavefront/searching sphere from query source s_i (e.g., when $s_i = K_q$, $T_t(i) = \{\tau_1, \tau_2\}$; when $s_i = o_1$, $T_t(i) = \varnothing$).

The searching process for the queries with an order is conducted by substituting Equation 15 - 19 into Algorithm 2.

5. EXPERIMENTS
In this section, we conducted extensive experiments on real spatial data sets to demonstrate the performance of the proposed User Oriented Trajectory Search. The two data sets used in our experiments were Beijing Road Network (BRN)

Table 1: Parameter setting

	BRN	NRN
Trajectory Number $\lvert T_r \rvert$	6,000 - 10,000 (default 8,000)	10,000 - 30,000 (default 20,000)
Query Location Number $\lvert O_q \rvert$	2-10 (default 8)	2-10 (default 8)

[8] and North America Road Network (NRN)[9], which contain 28,342 vertexes and 175,812 vertexes respectively, stored as adjacency lists. In BRN, we adopted the real trajectory data collected by the MOIR project [22]. In ORN, the synthetic trajectory data were used. All algorithms were implemented in C++ and tested on a Windows platform with Intel Core i5-2410M Processor (2.67GHz, 3MB L3) and 4GB memory.

In our experiments, the road networks resided in the memory when running Dijkstra's algorithm as the storage memory occupied by BRN/NRN was less than 20MB, which is trivial for most hand-held devices in nowadays. On the other hand, the trajectory data were stored in the disk due to their large size. To achieve high data access efficiency, a trajectory indexing approach was adopted. All trajectories were represented as a sequence of vertexes (sample points), such as $\tau = \{v_1, v_2, ..., v_m\}$, where $v_i, i \in [1, m]$ is the vertex in the road network $G(V, E)$. For each vertex $v_i \in V$, we maintained a pointer-list $v_{i \cdot traj}$ to identify the trajectories that contain the vertex v_i (i.e., pointing to the positions of the trajectories in the disk). An example is demonstrated as the following:

$$\begin{cases} v \in \tau_1 \\ v \in \tau_2 \quad \Rightarrow v_{\cdot traj} = \{\tau_1, \tau_2, \tau_3\} \\ v \in \tau_3 \end{cases}$$

Once the vertex v is scanned by a network browsing wavefront, the related trajectories (i.e., the trajectories in the pointer-list $v_{\cdot traj}$) can be accessed efficiently.

In this work, all experiment results were averaged over 20 independent trails with different query inputs. The main performance metrics were CPU time and the number of visited trajectories. The number of visited trajectories was selected as a metric for two reasons: (i) it can describe the exact amount of data access; (ii) it can reflect the real disk I/O requirement to a certain degree. The parameter settings are listed in table 1. By default, the number of trajectories were 8,000 and 20,000 in BRN and NRN respectively. In the meantime, the number of query locations was set to 8 for both BRN and NRN. The query locations were randomly selected from road networks. For the purpose of comparison, two naive algorithms were also implemented: the spatial first searching method (Section 3) denoted as "Spatial First" in Figures 4 & 5 and the collaborative searching method without the heuristic searching strategy denoted as "Without heuristic" in Figures 4 & 5.

5.1 Effect of Trajectory Number $\lvert T_r \rvert$
First of all, we investigated the effect of trajectory number $\lvert T_r \rvert$ on the performance of the three UOTS search ap-

---

[8] http://www.iscas.ac.cn/
[9] http://www.cs.utah.edu/ lifeifei/SpatialDataset.htm

proaches (i.e., Spatial First, Collaborative and Collaborative without heuristic searching strategy) with the default settings. For collaborative searching approaches, the higher the density of data objects, the smaller the required search range. In Figures 4(a) 4(b) 4(c) 4(d), the CPU time and the number of visited trajectories (Disk I/O times) for both the collaborative searching algorithms (with and without heuristic searching strategy) decreased with the increasing number of total trajectories. From Figures 4(a) 4(b) 4(c) 4(d), it is clear that the CPU time and the number of visited trajectories (Disk I/O times) required by the Spatial First method were more than one order of magnitude higher than that of the collaborative searching approach. In addition, with the help of the heuristic searching strategy (Section 4.2), the performance of the collaborative searching method was improved by 2-4 times in terms of both CPU time and the number of visited trajectories. These results clearly demonstrated the importance of the smart selection of tight bounds (to constrain the global searching area in a smaller range) and the necessity of the heuristic searching strategy.

5.2 Effect of Query Location Number $|O_q|$

Figures 4(e) 4(f) 4(g) 4(h) present the performance of the proposed three UOTS search approaches with varying numbers of query locations. Since more location candidates cause more query sources to be processed, the CPU time and the number of visited trajectories (Disk I/O times) are expected to be higher for all three searching approaches. However, the CPU time and the Disk I/O time of the Spatial First method increased much faster than the proposed collaborative searching approach for two reasons. The first one is due to its loose upper/lower bounds causing even more trajectories to be scanned during the query processing. The second one is that the Spatial First method treats all query sources equally since no heuristic searching strategy is adopted. For instance, with $|O_q| = 10$, the proposed collaborative searching approach outperformed Spatial First by almost two orders of magnitude (for both CPU time and Disk I/O times).

5.3 Queries with an order

Compared to queries without an order, when a query comes with an order constraint, more computation effort is needed to figure out the upper/lower bounds and the spatial similarity between a trajectory and a sequence of query locations according to Equation 14. Furthermore, a query location may not be matched with the nearest point on a trajectory, which consequently requires a scan of more trajectory points to get the best matching. Therefore, more CPU time and higher number of visited trajectories (Disk I/O times) are inevitable, as shown in Figure 5. However, the relative performance patterns were still similar to Figure 4.

6. RELATED WORK
6.1 Trajectory Similarity Search

The problem of trajectory similarity search [1, 13, 32, 7, 4, 15] has been extensively studied in the last two decades. Generally, the query processing takes two steps. First, a similarity/distance function is defined by some kind of aggregation of distances between trajectory points, to evaluate the similarity between a trajectory and a given sample. Second, an efficient solution is proposed to search the result over a large trajectory data set. Several types of trajectory similarity functions have been proposed in existing studies for different applications, including Euclidean Distance [1], Dynamic Time Warping [32], Longest Common Subsequence [29], Edit Distance [10], Edit Distance with Real Penalty [8], Edit Distance on Real Sequences [9], and the techniques for time series data similarity/approximation evaluation are also studied in [25, 28].

A similarity function is normally application-specific, and despite the bulk of literature on trajectory similarity [1, 13, 32, 7, 4, 8, 15, 10], none of them fulfills the requirements of our applications, in which the query input consists of a set of query locations and a set of keywords describing the textual attributes of a trajectory. Consequently, both spatial and textual domains should be considered in the trajectory similarity function. For query processing, most of the existing works are conducted in free spaces (e.g., Euclidean space [10]) and a spatial index (e.g., R-tree [18]) is adopted to accelerate the query efficiency. In our work, the object's movement is constrained in road networks, rather than a free space. The optimization techniques in the free space may fail to solve the problem in spatial networks since the bounds proposed in the free space is not always valid in spatial networks. This is the main reason why the network expansion approach (i.e., Dijkstra's expansion [12]) is adopted in our work.

6.2 Spatial Keyword Search

Spatial Keyword Search [35, 19, 14, 11, 5, 31, 23, 6] (i.e., queries on spatial objects associated with textual information, which can be seen as a combination of spatial query and textual matching search) has received significant attentions in recent years, due to the prevalence of spatial web objects on the Internet. In general, the keyword matching search in the textual domain can be classified into two categories. In the first category, the query keywords are used as a Boolean filter [35, 19, 14] to determine whether a spatial object contains this keyword or not. In the second category, the textual relevancy to a query is computed by language models and a probabilistic ranking function, such as the location-aware top-k text retrieval(LkT) query [11] and its variants (e.g., MkSK query [31] and RSTkNN query [23], which study the spatial keyword queries over moving query locations and the reverse spatial-textual kNN search).

To address the spatial keyword search efficiently, several hybrid indexing methods [14, 5, 11, 33, 34] were proposed, which can be regarded as the integration of a spatial index (e.g., the R-tree) and a text index (e.g., inverted lists). However, these indexing approaches are not suitable for our problem, since (i) the optimization techniques in Euclidean space fail to solve the problem in spatial networks (the bounds may be invalid or even loose); and (ii) there is a lack of an effective scheduling strategy to handle multiple query sources.

7. CONCLUSION

In this paper, we proposed and investigated a novel User Oriented Trajectory Search (UOTS) for trip recommendation. Different from traditional trajectory search by locations (spatial similarity only), in the new UOTS query, both the spatial similarity and user-preference were taken into consideration. If a trajectory connects or is close to a set
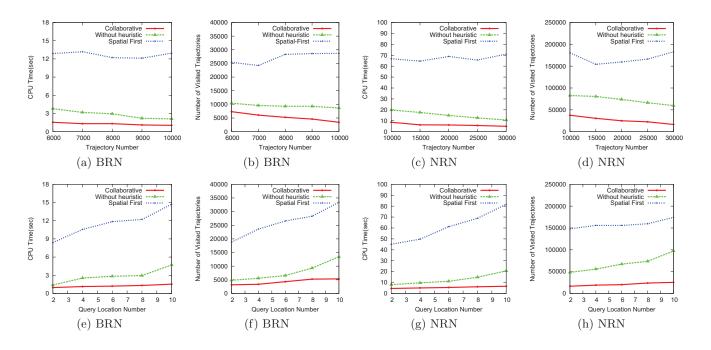
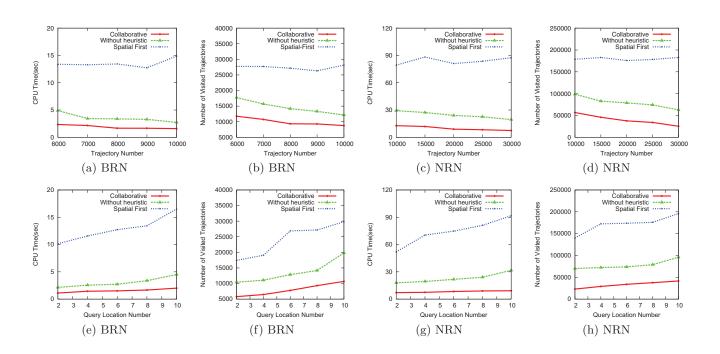Figure 4: Performance for queries without an order

Figure 5: Performance for queries with an order

of traveler-specified places, and the textual attributes of the trajectory are similar to the traveler's preference, it will be recommended to the traveler for consideration. This type of queries can bring significant benefits to travelers in many popular applications such as trip planning and recommendation. To address the UOTS query efficiently, a collaborative searching approach was proposed. A pair of bounds was devised to constrain the searching range while a heuristic strategy based on priority ranking was adopted to schedule the multiple query sources. In addition, the proposed collaborative searching approach can be further extended to situations where the query locations are ordered. Finally, the performance of the proposed UOTS query was demonstrated through extensive experiments.

8. ACKNOWLEDGEMENT

9. 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] X. Cao, G. Cong, and C. Jensen. Retrieving top-k prestige-based relevant spatial web objects. In *VLDB*, volume 3, pages 373–384, 2010.

[6] X. Cao, G. Cong, C. Jensen, and B. Ooi. Collective spatial keyword querying. In *SIGMOD*, 2011.

[7] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.

[8] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, pages 792–803, 2004.

[9] L. Chen, M. T. Ozsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.

[10] 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.

[11] G. Cong, C. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. In *PVLDB*, volume 2, pages 337–348, 2009.

[12] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Math*, 1:269–271, 1959.

[13] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.

[14] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, 2008.

[15] E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based most similar trajectory search. In *ICDE*, pages 816–825, 2007.

[16] 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.

[17] J. Greenfeld. Matching gps observations to locations on a digital map. In *81th Annual Meeting of the Transportation Research Board*, 2002.

[18] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.

[19] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *SSDBM*, page 16, 2007.

[20] H. Jagadish, B. Ooi, K.-L. Tan, C. Yu, and R. Zhang. idistance: An adaptive b+-tree based indexing method for nearest neighbour search. *ACM TODS*, 30(2):364–397, 2005.

[21] 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.

[22] 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.

[23] J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In *SIGMOD*, 2011.

[24] T. M. Mitchell. Artificial neural networks. *Machine Learning, WCB-McGraw-Hill*, 1997.

[25] M. D. Morse and J. M. Patel. An efficient and accurate method for evaluating time series similarity. In *SIGMOD*, pages 569–580, 2007.

[26] T. P. Ning, S. Michael, and K. Vipin. Introduction to data mining. 2005.

[27] Salton. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, pages 513–523, 1988.

[28] R. Sherkat and D. Rafiei. On efficiently searching trajectories and archival data for historical similarities. In *PVLDB*, 2008.

[29] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684, 2002.

[30] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing globalb curve-matching algorithms. In *SSDBM*, 2006.

[31] D. Wu, M. Yiu, C. Jensen, and G. Cong. Efficient continuously moving top-k spatial keyword query processing. In *ICDE*, 2011.

[32] B.-K. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208, 1998.

[33] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *ICDE*, pages 688–699, 2009.

[34] D. Zhang, B. C. Ooi, and A. K. H. Tung. Locating mapped resources in web 2.0. In *ICDE*, pages 521–532, 2010.

[35] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W. Ma. Hybrid index structures for location-based web search. In *CIKM*, pages 155–162, 2005.