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

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/closed to the specified query locations, and the textual attributes of the trajectory are similar to the traveler’s 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. EDBT 2012, March 26-30, 2012, Berlin, Germany. Copyright 2012 ACM 978-1-4503-0790-1/12/03 ...$10.00.

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 Bikely1, GPS-Waypoints2, Share-My-Routes3, Microsoft GeoLife4. In the meantime, more and more social network sites, including Twitter5, Four-square6 and Facebook7, 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 toll 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
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 \( K_\tau \) 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 \( K_\tau \)) 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 \( K_\tau \) 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...
In this work, road networks are modeled by connected and undirected planar graphs \( G(V, E) \), where \( V \) is the set of vertices 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, the weight is associated with factors 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] or grouped, by bus or by private vehicle, etc. The weight of each textual attribute can be calculated by TF-IDF [27] hence \( K_r \) is transformed into a high dimensional vector (i.e., a point in high dimensional space).

2.3 Spatial-Textual Distance Function

The raw trajectory samples obtained from GPS devices are typically of the form of \((\text{longitude}, \text{latitude}, \text{time}})\). How to map the \((\text{longitude}, \text{latitude})\) pair onto a given road network is a problem that has been studied in various fields such as location-based services and navigation systems. In this work, we define the spatial attribute of a trajectory in 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 \(\tau\) 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 \(\tau\) has a set of textual attributes \(K_r\), 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 \(K_r\) is transformed into a high dimensional vector (i.e., a point in high dimensional space).

Given 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 \(\tau\) and a data point \(o\) in a road network, the minimum distance \(d_M(o, \tau)\) between data point \(o\) and trajectory \(\tau\) is defined as

\[
d_M(o, \tau) = \min_{v_i \in \tau} \{sd(o, v_i)\},
\]

where \(v_i\) is the vertex belonging to \(\tau\).

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 \(T_{dist}(K_q, K_r)\) 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 \([0, 1]\).

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

\[
T_{dist}(K_q, K_r) = 1 - \frac{K_q \cdot K_r}{\|K_q\|^2 + \|K_r\|^2 - K_q \cdot K_r}
\]

By combining Equation 2 and 3, the spatial-textual distance between \(q\) and \(\tau\) is defined as

\[
ST_{dist}(q, \tau) = \lambda \cdot S_{dist}(O_q, \tau) + (1 - \lambda) \cdot T_{dist}(K_q, K_r)
\]

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 \(\lambda\) 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.
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 \( ST_{dist}(q, \tau) \leq ST_{dist}(q, \tau') \), \( \forall \tau' \in T_r \setminus \tau \).

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 \( \tau \), we compute the corresponding textual distance \( T_{dist}(K_q, \tau) \) 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 \( \tau_1, \tau_2, \tau_3 \) 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 \( o_1 \). 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 speed 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 \( r_i \). 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 \( d_M(o_i, \tau) = \text{sd}(o_i, v) \). For example, \( v_5 \) is the closest vertex to \( o_1 \) in \( \tau_3 \) and \( d_M(o_1, \tau_3) = \text{sd}(o_1, v_5) \). Once a trajectory \( \tau \) has been scanned by the expansion wavefronts from every query location \( o_i \in O_q, i \in [1, 4] \), such as \( \tau_2 \) 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 \( \tau \) and query points \( O_q \) according to Equation 2: e.g.,

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

This type of trajectories (e.g., \( \tau_2 \)) is denoted as “fully scanned trajectory” in this section. Then, we map the corresponding textual attributes \( K_r \) to the high dimension space and calculate the textual distance \( T_{dist}(K_q, \tau) \) by Equation 3. Finally, through integrating the values of \( S_{dist}(O_q, \tau) \) and \( T_{dist}(K_q, \tau) \) 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 \( \tau \) is greater than another trajectory’s upper bound, \( \tau \) 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., \( \tau_2 \) in Figure 2), we define a global upper bound \( UB \) as

\[
UB = \min_{\tau' \in T_r} \{ST_{dist}(q, \tau')\}
\]

where \( T_r \) 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., \( \tau \) is a trajectory which has not been fully scanned, such as \( \tau_1 \) and \( \tau_3 \) in Figure 2). In particular, trajectories such as \( \tau_1 \) are denoted as “unscanned” trajectory and trajectories such as \( \tau_3 \) is denoted as “partly scanned” trajectory.) Since Dijkstra’s algorithm always chooses the vertex with the smallest distance label for expansion, if a trajectory \( \tau \) 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,

\[
\sum_{i=1}^{m} d_M(o_i, \tau) > \sum_{o_2 \in O_q} d_M(o_2, \tau) + \sum_{o_y \in O_q} r_y
\]

where \( m \) is the size of query location set \( O_q \) and \( O_r \) is the set of locations whose expansion wavefronts have scanned \( \tau \) and \( O_q \) is the set of location whose expansion wavefronts have not scanned \( \tau \). \( O_t \cup O_q = O_r \). For instance, in Figure 2, \( \sum_{o_2 \in O_q} d_M(o_2, \tau_3) = d_M(o_3, \tau_3) \) and \( \sum_{o_y \in O_q} r_y = r_1 + r_2 + r_4 \). Obviously, we have

\[
\begin{align*}
\text{d}_M(o_1, \tau_1) &> r_1 \\
\text{d}_M(o_2, \tau_2) &> r_2 \\
\text{d}_M(o_2, \tau_3) &> r_3 + r_4 + r_2 + r_4 \\
\text{d}_M(o_3, \tau_3) &> r_4
\end{align*}
\]

According to Equation 6, we can use \( \sum_{o_2 \in O_q} d_M(o_2, \tau) + \sum_{o_y \in O_q} r_y \) to replace \( \sum_{i=1}^{m} d_M(o_i, \tau) \) in Equation 2 and have

\[
S_{dist}(O_q, \tau) = \frac{2}{1 + e^{-\sum_{i=1}^{4} d_M(o_i, \tau)}} - 1
\]
age load, we only compute and maintain the partly scanned dynamic value and continuously updated during the searching process. The trajectory $\tau$ with the minimum value of $ST(q, \tau)$ (i.e., $UB$) will be returned (line 14-15).

\begin{algorithm}
\caption{Spatial-First Trajectory Search}
\begin{algorithmic}[1]
\State $LB \leftarrow +\infty; UB \leftarrow +\infty$
\While{true}
  \For{each $o_i \in O_q$}
    \State $v \leftarrow \text{Expand}(a_i)$
    \For{each trajectory $\tau \in v.trajList$}
      \If{$\tau.scan(o_i) = \text{false}$}
        \State $\tau.scan(o_i) \leftarrow \text{true}$
        \State $\text{Update } S_{\text{dist}}(O_q, \tau)\text{.lb}$
        \State $UB \leftarrow \min(\lambda \cdot S_{\text{dist}}(O_q, \tau)\text{.lb})$
      \EndIf
    \EndFor
    \State $\text{if } \tau.scan(o) \text{ is true, } \forall a_i \in O_t$, \text{Calculate } $ST_{\text{dist}}(q, \tau)$
    \If{$ST_{\text{dist}}(q, \tau) < UB$}
      \State $UB \leftarrow ST_{\text{dist}}(q, \tau)$
    \EndIf
    \If{$LB > UB$}
      \State \text{return } UB \text{ and the corresponding } \tau$
    \EndIf
  \EndFor
\EndWhile
\end{algorithmic}
\end{algorithm}

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 4) is loose since only the spatial domain is considered and the influence of the textual domain is totally ignored. The parameter $\lambda$ is used to adjust the relative importance of spatial proximity factor and textual similarity factor (Equation 4). The smaller the value of $\lambda$, 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).
In the collaborative searching approach, the trajectory search is conducted in the first place, to group the data points into clusters based on their distribution. For each cluster \( C_i \), \( i \in \{1,2,3\} \), a reference point \( m_i \) is selected. Then, we compute and record the textual distance (Equation 3) between \( m_i \) and every data point \( v \) 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 hyper-sphere centered at \( K_q \). We use \( \Delta r \) as the initial searching radius (i.e., \( R = \Delta r \)), and the search radius is increased by \( \Delta r \) (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 \( \Delta r \), 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, \( \tau_1, \tau_2, \tau_3 \) 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 \( v \) has been scanned by the expansion wavefront from \( o_i, i \in \{1,4\}, \) \( v \) is just the closest vertex to \( o_i \) in \( \tau \) (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 vertices to \( o_1, o_2, o_3, o_4 \) respectively, and \( \{v_2, v_3\} \in \tau_1 \) are the closest vertices to \( o_2, o_3 \) respectively. \( v_6 \in \tau_3 \) is the closest vertex to \( o_4 \). If a trajectory \( \tau \) has been scanned by the expansion wavefronts from every query point \( o_i \in O_q \), 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 \( \tau \) that has not been fully scanned in spatial (e.g., \( \tau_1, \tau_3 \), in particular, this type of trajectories is denoted as “partly scanned in spatial"), its textual 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 \( \Delta r \), (i.e., \( R = R + \Delta r \)). If a data point \( K_r \) is inside the searching sphere, its textual distance \( T_{dist}(K_r, K_q) \) to \( K_q \) can be retrieved easily. Otherwise, its textual distance to \( K_q \) can be estimated by a lower bound

\[
T_{dist}(K_q, K_r).lb = R
\]  

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

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

1. trajectory \( \tau \) is fully scanned in the spatial domain but unscanned in the textual domain.
2. trajectory \( \tau \) is partly scanned in the spatial domain and fully scanned in the textual domain.
3. trajectory \( \tau \) 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 \( \tau \) 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 \( \tau_2 \) has been fully scanned (i.e., scanned by the expansion wavefronts from every query point \( o_i \in O_q \)), while trajectory \( \tau_1, \tau_3 \) are both partly scanned (\( \tau_1 \) is only scanned by the expansion wavefronts from \( o_1 \) and \( o_2 \), and \( \tau_3 \) is only scanned by the expansion wavefront from \( o_3 \)).

In the textual domain, the radius of the searching sphere is \( \tau \), where \( \tau \) is required, and the labels can be calculated as

\[
\text{label}_{ST}(s_i) = \sum_{m \in \{T_p-T_l(i)\}} e^{-ST_{dist}(q, \tau_m).lb} \tag{13}
\]

where \( T_p \) is the set of partly scanned trajectories in both spatial and textual domains (e.g., \( \tau_1, \tau_3 \)), and \( T_l(i) \) is the set of trajectories scanned by the expansion wavefront/searching sphere from query source \( s_i \) (e.g., when \( s_i = K_q \), \( T_l(i) = \{\tau_1, \tau_3\} \)), and \( \tau_m \) is the maximum label from the corresponding trajectory (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.

The complete procedure of the collaborative searching method is described in Algorithm 2. Initially, the values of \( \text{labels}(o_i), \forall o_i \in O_q \) and \( \text{labels}_{ST}(s_i), \forall s_i \in (O_q \cup K_q) \) 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 \( \text{labels}(o_i) \) is selected as the expansion center \( Ec \) (line 3-4) and the expansion wavefront is expanded from \( Ec \) (line 6). Each trajectory \( \tau \) passing through \( v \) will be checked. If \( \tau \) has not been scanned by the expansion wavefront from \( o_i \), \( \tau \) 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 \( \text{labels}(o_j) \) is greater than \( \text{label}_{ST}(o_i) \), 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 \( K_q \) 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 \( \text{label}_{ST}(s_i) \) is selected as the expansion center \( Ec \) (line 19-20). If \( s_i = o_i \in O_q \), 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 \( \tau \) is labeled as \( \tau.scan(s_i) = true \), and the corresponding \( ST_{dist}(q, \tau).lb \), all labels \( \text{label}_{ST} \) and \( LB \) need to be updated (line 23-27). If there exists a query source \( s_j \in (O_q \cup K_q) \) and \( \text{label}_{ST}(s_j) > \text{label}_{ST}(s_i) \), \( s_j \) 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 \( \tau \) 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:** $T, q$

**Result:** $\min_{t \in T} ST_{dist}(q, \tau)$

1. $LB \leftarrow +\infty; UB \leftarrow +\infty; R \leftarrow 0$
2. $\text{label}\_S(o_i) = 0, \forall o_i \in O_q; \text{label}\_S(s_i) = 0, \forall s_i \in \{O_q \cup K_q\}$
3. Select $o_i \in O_q$ with the maximum $\text{label}\_S(o_i)$
4. $Ec \leftarrow o_i$
5. while true do
   6. $v \leftarrow \text{Expand} (Ec)$
   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 $\exists \text{label}\_S(o_i) > \text{label}\_S(o_j), o_j \in O_q$ then
         12. $Ec \leftarrow o_i$
      13. $\text{Expand}(K_q); \; /R \leftarrow R + \Delta r$
      14. for each scanned data point $K_q$ do
         15. $\tau$.scan$(K_q) \leftarrow$ true;
         16. Update $ST_{dist}(q,\tau).lb$ and all labels;
         17. if $\exists \tau$.scan$(o) = \text{true}, o \in O_q$ then
            18. Break;
      19. Select $s_i \in \{O_q \cup K_q\}$ with the maximum $\text{label}\_S(t)$
      20. $Ec \leftarrow s_i$
      21. while true do
         22. $\text{Expand}(Ec)$;
         23. for each scanned trajectory $\tau$ do
            24. if $\tau$.scan$(s_i) = \text{false}$ then
               25. $\tau$.scan$(s_i) \leftarrow$ true;
               26. Update $ST_{dist}(q,\tau).lb$ and all labels;
               27. $LB \leftarrow \min\{ST_{dist}(q,\tau).lb\}$
            28. if $\exists \text{label}\_S(s_i) > \text{label}\_S(s_j), s_j \in \{O_q \cup K_q\}$ then
               29. $Ec \leftarrow s_i$
            30. if $\tau$.scan$(s_i)$ 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 $\tau$;

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 $\tau$ is defined in a recursive way as

$$S^o_{sim}(O_q, \tau) = \max \left\{ \begin{array}{l} e^{-sd(O_q,head,\tau.head)} + S^o_{sim}(O_q,\tau) \\ S^o_{sim}(O_q,\tau) \end{array} \right\}$$

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 $\tau$ is defined as follows

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

(15)

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

To obtain the exact spatial distance between $O_q$ and $\tau$, 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 $\tau$ have been scanned in the spatial domain) or “unsanned in spatial” (i.e., no point in $\tau$ 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’s algorithm [12]. Conceptually, the browsed region is constrained within a circle centered at $o_i$ (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 $\tau$ (e.g., $r_1, r_2, r_3$ in Figure 3), its spatial distance lower bound $S^o_{dist}(O_q, \tau).lb$ can be computed as follows.

If $v \in \tau$ has been scanned by the browsing wavefront from $o \in O_q$, the network distance $sd(v, o)$ between $v$ and $o$ 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.e., $sd(v, o) > r_i$), 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 $S^o_{sim}(O_q, \tau).ub$ can be computed as:

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

$$S^o_{sim}(O_q, \tau) = \max \left\{ \begin{array}{l} \epsilon^{-sd(O_q,head,\tau.head)} + S^o_{sim}(O_q,\tau) \\ S^o_{sim}(O_q,\tau) \end{array} \right\}$$

$$\leq \max \left\{ \begin{array}{l} S^o_{sim}(O_q,\tau) \\ S^o_{sim}(O_q,\tau) \end{array} \right\} = S^o_{sim}(O_q,\tau).ub$$

$$\alpha = \left\{ \begin{array}{l} e^{-sd(O_q,head,\tau.head)} + S^o_{sim}(O_q,\tau) \\ e^{-r_i} + S^o_{sim}(O_q,\tau) \end{array} \right\}$$

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

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

$$S_{\text{dist}}(O_q, \tau) = \frac{1}{1 + S_{\text{sim}}(O_q, \tau)} \geq \frac{1}{1 + S_{\text{sim}}(O_q, \tau).ub}$$

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 $T_{\text{dist}}(K_q, K_r).lb$ can be computed by Equation 10. Hence, the spatial-textual distance lower bound for queries with an order is defined as

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

1. trajectory $\tau$ is fully scanned in the spatial domain but unsanned in the textual domain.
2. trajectory $\tau$ is partly scanned in the spatial domain and fully scanned in the textual domain.
3. trajectory $\tau$ is partly scanned in the spatial domain and unsanned 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 $\text{label}_S(o_i), o_i \in O_q$ and spatial textual priority label $\text{label}_S(s_i), s_i \in \{O_q, K_r\}$ can be identified.

$$\text{label}_S^o(a_i) = \sum_{m \in (T_p - T_t(i))} e^{-S_{\text{dist}}(O_q, r_m).lb} (18)$$

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 $o_i$.)

$$\text{label}_S^o(T_s(i)) = \sum_{m \in (T_p - T_t(i))} e^{-ST_{\text{dist}}(q, r_m).lb} \quad (19)$$

where $T_p$ is the set of partly scanned trajectories in both spatial and textual domains (e.g., $\tau_1, \tau_2, \tau_3$), 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 $s_i = o_1$, $T_t(i) = \emptyset$).

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) and North America Road Network (NRN), which contain 28,342 vertices and 175,812 vertices 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 vertices (sample points), such as $\tau = \{v_1, v_2, \ldots, v_m\}$, 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 $v_{i,.traj}$ 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:

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

Once the vertex $v$ is scanned by a network browsing wavefront, the related trajectories (i.e., the trajectories in the pointer-list $v_{.traj}$) 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 $|T_s|$ 

First of all, we investigated the effect of trajectory number $|T_s|$ on the performance of the three UOTS search ap-

---

8: http://www.iscas.ac.cn/
9: http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm
approaches (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 keyword search is 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. Difference 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
Figure 4: Performance for queries without an order

Figure 5: Performance for queries with an order
set 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

We thank Dr. Ke Deng and Prof. Xiaofang Zhou for their insightful comments. Dr. Bo Yuan was supported by the National Natural Science Foundation of China (No.60905030).

9. REFERENCES