Searching Trajectories by Regions of Interest

Shuo Shang, Lisi Chen, Christian S. Jensen, *Fellow, IEEE,* Ji-Rong Wen, and Panos Kalnis

Abstract—With the increasing availability of moving-object tracking data, trajectory search is increasingly important. We propose and investigate a novel query type named trajectory search by regions of interest (TSR query). Given an argument set of trajectories, a TSR query takes a set of regions of interest as a parameter and returns the trajectory in the argument set with the highest spatial-density correlation to the query regions. This type of query is useful in many popular applications such as trip planning and recommendation, and location based services in general. TSR query processing faces three challenges: how to model the spatial-density correlation between query regions and data trajectories, how to effectively prune the search space, and how to effectively schedule multiple so-called query sources. To tackle these challenges, a series of new metrics are defined to model spatial-density correlations. An efficient trajectory search algorithm is developed that exploits upper and lower bounds to prune the search space and that adopts a query-source selection strategy, as well as integrates a heuristic search strategy based on priority ranking to schedule multiple query sources. The performance of TSR query processing is studied in extensive experiments based on real and synthetic spatial data.

Keywords—Trajectory search by regions, Spatial-density correlation, Spatial networks, Spatial databases

1 Introduction

The availability of GPS-equipped devices [24] (e.g., vehicle navigation systems and smart phones) and online map-based services (e.g., Google Maps[1], Bing Maps[2], and MapQuest[3]) enable people to capture their current location and to share their trajectories by means of services such as Bikely[4], GPS-Way-points[5], Share-My-Routes[6], and Microsoft GeoLife[7]. Also, more and more social networking sites, including Twitter[8], Four square[9], and Facebook[10], support the sharing of trajectories. The availability of massive trajectory data enables novel mobile applications. Such applications may utilize trajectory search, which finds trajectories that are similar in some specific sense to query parameters (a set or sequence of locations [5], [16], [17], [19], [25], or regions). This type of query can benefit popular services, such as travel planning and recommendation, and location-based services in general. For example, when planning a trip to multiple places in an unfamiliar city, a tourist may benefit from the experience of previous visitors. In particular, visitors with similar interests may have visited nearby landmarks that the user may not know, but may be interested in. Or others may have avoided a specific road because it is unpleasant, although it may seem like a good choice in terms of distance. Such experiences are captured in trajectories shared by previous visitors. In existing studies (e.g., [5], [16], [17], [19], [25]), all trajectories are treated the same, regardless of their frequencies of use. For example, some less traveled trajectories may be new or just less popular because the region they are in is less traveled. Such trajectories may still be of interest to users.

In most existing studies on trajectory search [5], [16], [19], the query parameters are a set or sequence of locations. However, in some cases, a place may not be a point location, but may be a region of interest that contains several spatial objects (e.g., a scenic area, a commercial district, or a dining area, where spatial objects can be points of interest (POIs), geo-tagged photos, or geo-tagged tweets). Moreover, especially when planning a trip in an unfamiliar city, users may fail to specify intended locations exactly and may use intended regions instead. These two common cases motivate our study.

Extending conventional trajectory search, we propose and investigate a novel query named trajectory search by regions (TSR). In our setting, a region is circular, and users can specify a region on a map simply by specifying a center and a radius. Given a trajectory set T, a user provides a set of regions of interest as query parameters, and the TSR query retrieves the trajectory from T with the highest spatial-density correlation with the query regions. Intuitively, a trajectory that is spatially close to regions with many spatial objects is more attractive to travelers than a further-away trajectory.

To the best of our knowledge, this is the first study of region-based trajectory search in spatial networks that takes

- *Shuo Shang is with King Abdullah University of Science and Technology, Saudi Arabia.*
  E-mail: jedi.shang@gmail.com
- *Lisi Chen is with Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China.*
  E-mail: chenlisi@comp.hkbu.edu.hk
- *Christian S. Jensen is with Department of Computer Science, Aalborg University, DK-9220 Aalborg Øst, Denmark.*
  E-mail: csj@cs.aau.dk
- *Ji-Rong Wen is with Beijing Key Laboratory of Big Data Management and Analysis Methods, and School of Information, Renmin University of China, P.R.China.*
  E-mail: jirong.wen@gmail.com
- *Panos Kalnis is with King Abdullah University of Science and Technology, Saudi Arabia.*
  E-mail: panos.kalnis@kaust.edu.sa

1. http://maps.google.com/
2. http://www.bing.com/maps/
3. http://www.mapquest.com
4. http://www.bikely.com/
5. http://www.gps-waypoints.net/
6. http://www.sharemyroutes.com/
7. http://research.microsoft.com/en-us/projects/geolife/
8. http://twitter.com/
9. http://foursquare.com/
10. http://www.facebook.com/

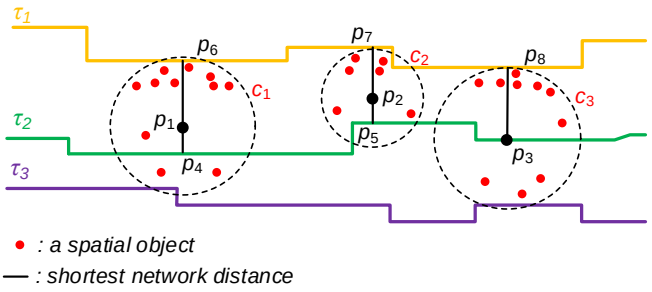


Fig. 1. An example of the TSR query

spatial-object density into account. Previous studies (e.g., [5], [19]) use spatial distance as the sole criterion when computing results. However, spatial distance in itself fails to fully capture the relationship between a trajectory and a set of regions. For example, a user may not be satisfied with a travel route with a relatively sparse distribution of nearby objects, although the route is spatially close to the centers of the given query regions.

This is illustrated in Figure 1, where c_1, c_2, and c_3 are TSR query regions, p_1, p_2, and p_3 are the corresponding region centers, and r_1, r_2, and r_3 are the radiuses. Points $p_3, p_4, ..., p_8$ are sample points in trajectories. Among the sample points in τ_1, p_6, p_7, and p_8 are the closest sample points to centers p_1, p_2, and p_3. Among the sample points in τ_2, p_4 and p_5 are the closest sample points to centers p_1 and p_2. Each region contains several spatial objects. If only spatial proximity to the region centers is considered [5], [19], trajectory τ_2 is returned because τ_2 is spatially closest to the region centers. If we consider the distributions of the spatial objects, τ_2 is less attractive than τ_1 because it is further away from the regions with high spatial-object density. When taking both spatial distances to the spatial objects within query regions and spatial-object density into account, τ_1 is the best choice for trajectory recommendation (although τ_2 is slightly better than τ_1 in spatial proximity).

The TSR query is studied in spatial networks, because in many practical scenarios, users (e.g., travelers and vehicles) move in spatial networks (e.g., road and railway networks) rather than in a Euclidean space. To the best of our knowledge, no existing method can compute the TSR query efficiently. Existing studies (e.g., [5], [16], [17], [19], [25]) do not take query regions and spatial-object distribution into account, and their associated query processing techniques are either not applicable or not effective for the TSR query.

We first develop a straight-forward approach to computing to the TSR query, called uniform-speed search (USS), which follows the filter-and-refine paradigm. At query time, each query region center p_i is used as a so-called query source, and network expansions (i.e., Dijkstra's expansion [6]) are performed from these query sources at the same speed to explore the spatial network. Uniform-speed search asks for trajectories spatially close to the dense subregions of the query regions. A pair of an upper and a lower bound on the spatial-density correlation between a trajectory and a set of query regions are defined to prune the search space. The main drawback of uniform-speed search is that it lacks effective scheduling for multiple query sources, which may lead to poor performance. In addition, if the centers of two query regions are close to each other and both are selected as query sources, their search spaces may overlap substantially. The trajectories in the overlap region will then be traversed more than once, which unnecessarily decreases performance.

To achieve better performance, a best-expansion search (BES) algorithm is proposed. First, we reuse an existing query-source selection strategy [18] to select a set of query sources from among the centers of the query regions. Second, we define new upper and lower bounds on the spatial-density correlation to prune the search space. Third, a heuristic search strategy based on priority ranking is developed to coordinate the use of multiple query sources. We maintain and make use of a dynamic priority ranking heap when processing the query. At each point in time, we expand the top-ranked query source until a new query source becomes top ranked. Compared to uniform-speed search, the best-expansion search algorithm has two major advantages: (i) it further prunes the search space for avoiding traversals of overlap regions; (ii) the effective heuristic search strategy focuses on trajectories more likely to be the solution and further improves query performance.

Next, it is also possible that a traveler may specify a visiting sequence for intended regions (e.g., c_1, c_2, and c_3 are the intended regions, and the visiting sequence is $c_1 \rightarrow c_2 \rightarrow c_3$). The proposed USS and BES algorithms are further extended to process the TSR queries with a sequence efficiently.

To sum up, the main contributions are as follows:

- We define a novel trajectory search by regions of interest (TSR) query, which is useful in trip planning and recommendation, and in location-based services in general.
- We define new measures to evaluate the spatial-density correlation between a trajectory and a set of regions of interest.
- We develop a best-expansion search (BES) algorithm to compute the TSR query efficiently with the support of upper and lower bounds and heuristic query-source scheduling.
- We further extend the BES algorithm to scenarios where a sequence of query regions is given.
- We conduct extensive experiments with real and synthetic spatial data to investigate the performance of the proposed algorithms.

The rest of the paper is organized as follows. Related work is covered in Section 2. Section 3 defines the spatial networks, trajectories, and metrics used in the paper; and it also gives problem definitions. A baseline uniform speed algorithm is introduced in Section 4, and the best-expansion search algorithm is covered in Section 5. The developed algorithms are further extended to practical scenarios in Section 6, which is followed by a coverage of experimental results in Section 7. Conclusions are drawn in Section 8.

2 Related Work

Trajectory search queries aim to find trajectories with the highest relevance to query arguments (e.g., a single spatial

point, multiple spatial points, or a trajectory) [5] [8] [16] [17] [25] [26]. Trajectory similarity functions may contain spatial [5], temporal [17], textual [16] [25], and density elements. The resulting queries are useful in many popular applications including travel planning, carpooling, friend recommendation in social networks, and location-based services in general.

We classify the existing trajectory search queries into three categories according to their arguments. In the point-to-trajectory category, the query argument is a single spatial point, and the query finds trajectories spatially close to the query point. Zheng et al. [26] extended this query to cover spatial and textual domains and propose the TkSK query, which retrieves the trajectories that are spatially close to the query point and also meet semantic requirements defined by the query. In the points-to-trajectory category, the query takes a set of locations (e.g., sightseeing places) as argument and returns a trajectory that connects or is close to the query locations according to specific metrics. The concept of trajectory search by locations (TSL) was first proposed by Chen et al. [5]. This study considers the spatial domain only (Euclidean space). Shang et al. [16] observe that spatial similarity does not fully capture the relationship between query locations and trajectories due to specific preferences of users. They then propose user oriented trajectory search and extended the query to cover both the spatial and the textual domains. Intuitively, if a trajectory is close to specified query locations (spatial domain) and its textual attribute values are similar to the user's textual preferences (textual domain), it is recommended to the user. In the trajectory-to-trajectory category, queries retrieve trajectories that are most similar to a query trajectory (e.g., [1], [3], [4], [7], [12], [13], [21], [23]). For example, the PTM query [17] takes spatio-temporal similarity into account, and the ATSQ query [25] considers spatial-textual similarity.

Unlike the existing studies, the Trajectory Search by Regions of interest (TSR) query aims to find a trajectory with the highest spatial-density correlation to a set or sequence of query regions. The existing TSL solution is ineffective for the TSR query due to two reasons. First, TSL only takes spatial proximity into account, while TSR considers both spatial distance and spatial-object density. Second, TSL is conducted in Euclidean space, and a spatial index (e.g., an R-tree [11]) is used to enhance the query efficiency. In our work, the objects' movements are constrained to a spatial network. When weights of network edges model a number of aspects of travel (e.g., fuel consumption and travel time), the lower bound of network distance may not be the corresponding Euclidean distance; hence, spatial indexes such as the R-tree [11] are ineffective. This is the main reason why we use network expansion (i.e., Dijkstra's expansion [6]).

The most related work is arguably the path nearby cluster (PNC) query [18], which we therefore cover in some detail. The TSR query and its solution differ from the PNC query and its solution in six respects. (i) **Query type**: the PNC query is a spatial-density query that is conducted in the spatial and density domains, while the TSR query is a spatial query (density is also considered but the query processing occurs in the spatial domain only). (ii) **Query argument and result**: the PNC query takes a route as argument, and it returns the top-k clusters with the highest distance-and-density evaluation factor with respect to the query route, while the TSR query takes a set of regions of interest as argument and returns the trajectory with the highest spatial-density correlation to the argument regions. (iii) **Similarity function**: the similarity function used for the PNC query evaluates the distance-and-density relevance in the spatial and density domains, and it combines these linearly. In the spatial domain, it measures the network distance between a cluster center and a route; and in the density domain, it computes the density of clusters. The similarity function of the TSR query evaluates the spatial-density correlation between a trajectory and a set of query regions in the spatial domain. The network distances between a trajectory and all spatial objects in query regions are taken into account. (iv) **Data model and algorithm structure**: for the PNC query, the densities of clusters are mapped to a one-dimensional space (the density domain), and PNC query processing searches this domain to finds the clusters with high spatial-object density. The TSR query has no sperate density domain. The density of spatial objects is the sum of the distances between a trajectory and the spatial objects in query regions. Due to these differences, PNC and TSR call for different algorithms. (v) **Optimization techniques**: due to the above differences from PNC, TSR needs specific optimization techniques, including upper and lower bounds (Equations 5–11, 15–19, 21–30), and heuristic functions (Equations 20, 30). Because the TSR query has multiple query regions as argument, a strategy is needed to schedule multiple query regions. TSR reuses and extends the query-source selection method of PNC (Equations 12–14) to select query sources from query regions. (vi) **Experimental spatial data sets**: different spatial data sets are used. For the PNC query, spatial objects are geo-tagged micro-blog posts, and trajectory data is not used, while for the TSR query, the spatial objects are POIs, and real and synthetic trajectory data is used. Due to these six differences, the TSR query and its solution are new. The PNC solution does not work for the TSR problem.

3 PRELIMINARIES

3.1 Spatial Networks

A spatial network is modeled by a connected and undirected graph $G(V, E, F, W)$, where V is a vertex set and $E \subseteq V \times V$ is an edge set. A vertex $v_i \in V$ denotes a road intersection or a termination of a road. An edge $e_k = (v_i, v_j) \in E$ is defined by two vertices and represents a road segment that enables travel between vertices v_i and v_j. Function $F : V \cup E \rightarrow Geometries$ records geometrical information of the spatial network G. In particular, it maps a vertex and an edge to the point location of the corresponding road intersection and to a polyline representing the corresponding road segment, respectively.

Function $W : E \rightarrow R$ is a function that assigns a real-valued weight to each edge. The weight $W(e)$ of an edge e represents the corresponding road segment's length or some other relevant property such as its fuel consumption [10], [22] or travel time [9], which can be acquired by mining historic

traffic data. When weights model aspects (e.g., fuel consumption and travel time), the lower bound of network distance may not be the corresponding Euclidean distance; therefore, spatial indexes such as the R-tree [11] are ineffective.

Spatial objects may located off edges or vertices in a spatial network. We assume that all spatial of the objects have already been mapped to the spatial network (on edges or vertices) based on some map-matching algorithm(e.g., [2], [14]). Next, each spatial object is assigned to its nearest vertex. For each vertex $p \in G.V$, the number of spatial objects that are assigned to p is assigned an attribute of p, which is denoted by $p.g$. A vertex and its assigned spatial objects constitute the smallest unit in spatial-object density computations, and thus we need not access individual spatial objects during TSR query processing

We share the spatial network modeling and spatial-object preprocessing with a previous study [18].

3.2 Trajectories and Regions of Interest

Raw trajectory samples obtained from GPS devices are typically of the form of $(longitude, latitude, time)$. We assume that all trajectory sample points have already been map matched onto the vertices of the spatial network by some map-matching algorithm (e.g., [2], [14]), and that between two adjacent sample points p_a and p_b, the object movement always follows the shortest path connecting p_a and p_b. A trajectory is defined as follows.

Definition: Trajectory
A trajectory is a finite, time-ordered sequence $\langle v_1, v_2, ..., v_n \rangle$, where $v_i = (p_i, t_i)$, with p_i being a sample point (at a vertex) in G and t_i being a timestamp. In this study, we only consider the spatial attribute of trajectories.

Definition: Region of interest
A region of interest is a subgraph $c \subseteq G$ that contains the vertices $c.V$ and edges $c.E$ from G that are in a region defined by a center v_m and a radius r, where $c.v_m$ is a vertex in $G.V$ and r is the network distance from c to the boundary of the region.

3.3 Spatial-density Correlation

Given any two vertices p_a and p_b in a spatial network, the network shortest path between them is denoted by $SP(p_a, p_b)$, and its length is denoted by $sd(p_a, p_b)$. Given a trajectory τ and a vertex o in a spatial network, the minimum distance $d_M(o, \tau)$ between vertex o and trajectory τ is defined by

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

where p_i is a vertex belonging to τ.

Given two spatial points p_1 and p_2, the spatial influence factor $I(p_1, p_2)$ is defined as follows.

$$I(p_1, p_2) = \begin{cases} 0 & \text{if } sd(p_1, p_2) > \epsilon \\ e^{-sd(p_1, p_2)} & \text{otherwise} \end{cases} \tag{2}$$

Here ϵ is a threshold. The value of $I(p_1, p_2)$ is inversely proportional to $sd(p_1, p_2)$. If the distance between p_1 and p_2 exceeds the threshold, the influence factor between them is set to 0. Intuitively, the threshold is used to prune trajectories faraway from the query regions. The value of $I(p_1, p_2)$ is in range $[0, e^{-\epsilon}]$, $e^{-\epsilon} \in (0, 1)$.

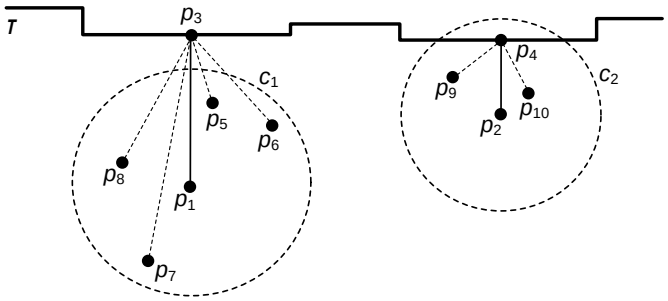


Fig. 2. Spatial-density correlation

The spatial-density correlation $C_{sd}(c, \tau)$ between a region c and a trajectory τ is defined as follows.

$$C_{sd}(c, \tau) = \sum_{p_i \in c} p_i.g \cdot I(p_i, p) \tag{3}$$

Here, p_i is a vertex belonging to c, $p \in \tau$ is the closest vertex to the region center $c.m$, and $p_i.g$ is the number of spatial objects attached to p_i. Spatial distance and spatial-object density are both taken into account. These functions extend the well known Longest Common Subsequence (LCSS) [20] by taking the density of spatial objects into account.

An example is shown in Figure 2, where τ is a trajectory and c_1 and c_2 are two regions and p_1 and p_2 are their centers correspondingly. Vertices $\{p_3, p_4\} \in \tau$ are the closest vertices to p_1 and p_2, and $\{p_5, p_6, p_7, p_8\} \in c_1$ and $\{p_9, p_{10}\} \in c_2$. According to Equations 4, the spatial-density correlations $C_{sd}(c_1, \tau)$ and $C_{sd}(c_2, \tau)$ are computed as $C_{sd}(c_1, \tau) = p_1.g \cdot I(p_1, p_3) + p_5.g \cdot I(p_5, p_3) + p_6.g \cdot I(p_6, p_3) + p_7.g \cdot I(p_7, p_3) + p_8.g \cdot I(p_8, p_3)$, and $C_{sd}(c_2, \tau) = p_2.g \cdot I(p_2, p_4) + p_9.g \cdot I(p_9, p_4) + p_{10}.g \cdot I(p_{10}, p_3)$.

In our settings, each region plays an equally significant role in the TSR query processing, so we use the Sigmoid function [15] to normalize the spatial-density correlation $C_{sd}(c, \tau)$ to a range $[0, 1]$, i.e., $(\frac{2}{1+e^{-C_{sd}(c,\tau)}} - 1) \in [0, 1]$. By combining the spatial-density correlation of each region $c_i \in C$, the spatial-density correlation between a set of regions C and a trajectory τ is given by

$$C_{sd}(C, \tau) = \sum_{c_i \in C} (\frac{2}{1 + e^{-C_{sd}(c_i, \tau)}} - 1). \tag{4}$$

Frequently used notations are given in Table I.

3.4 Problem Definition

Given a spatial network $G(V, E, F, W)$, a set of trajectories T, and a set of regions of interest C, the TSR query finds the trajectory $\tau \in T$ with the maximum spatial-density correlation $C_{sd}(C, \tau)$: $\forall \tau' \in T \setminus \{\tau\}(C_{sd}(C, \tau) \geq C_{sd}(C, \tau'))$.

Table I: A List of Notions

Notion	Description
$G.V$	the set of vertices in graph G
$G.E$	the set of edges in graph G
$p.g$	the number of spatial objects attached to p
$c.m$	the center of region c
$sd(p_i, p_j)$	network distance between vertices p_i and p_j
$d_M(p, \tau)$	network distance between vertex p and trajectory τ
$I(p_i, p_j)$	spatial influence factor between vertices p_i and p_j
$C_{sd}()$	spatial-density correlation factor
$C_{sd}().lb, C_{sd}().ub$	the lower and upper bounds of spatial-density correlation
UB, LB	the global upper and lower bounds
T_f, T_p	a set of fully and partially scanned trajectories
$p.c$	the cluster of region centers that are attached to query source p
$p.l$	the priority label of query source p

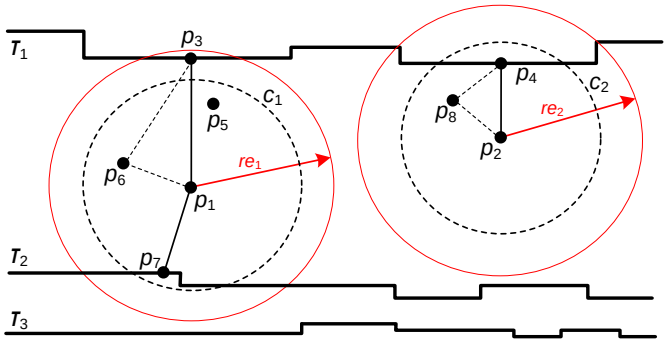


Fig. 3. An example of uniform-speed search

4 Baseline Method

4.1 Basic Idea

The uniform-speed search (USS) is a straightforward approach to computing the TRS query based on the filter-and-refine paradigm. Given a trajectory data set T and a query region set C, USS asks for the trajectories spatially close to the spatial-object dense areas within the query regions. Each region center $c_i.m$ ($c_i \in C$) is selected as a so-called query source, and network expansions (i.e., Dijkstra's expansion [6]) are performed from the query sources at the same speed to explore the network. A pair of upper and lower bounds of spatial-density correlation are defined to prune the search space. By integrating the results, the trajectory with the maximum spatial-density correlation to C is found.

Consider the example in Figure 3. Regions c_1 and c_2 are two query regions, and τ_1, τ_2, and τ_3 are trajectories. Vertices p_1 and p_2 are centers of regions c_1 and c_2; p_3 and p_4 in τ_1 are the closest vertices to p_1 and p_2; and $p_7 \in \tau_2$ is the closest vertex to p_1. In USS, we use region centers p_1 and p_2 as query sources and apply Dijkstra's expansion [6] from these at the same speed. The explored circular regions (Figure 3), where the radii are the shortest network distances from query sources p_1 and p_2 to the corresponding expansion boundaries (re_1 and re_2). Here, $re_1 = re_2$ because the expansion speeds are the same.

According to Dijkstra's algorithm that selects the vertex with the minimum distance label for expansion, if $p \in \tau$ is the first vertex scanned by the expansion from p', p is just the closest vertex to p' (i.e., $d_M(p', \tau) = sd(p, p')$). For instance, p_3 is the closest vertex to p_1, thus $d_M(p_1, \tau) = sd(p_1, p_3)$. When a trajectory such as τ_1 in Figure 3 is covered by the expansions from all query sources, the trajectory is marked as fully scanned. A trajectory such as τ_2 is marked as partially scanned, and a trajectory such as τ_3 is marked as unscanned.

4.2 Upper and Lower Bounds

To prune the search space, a pair of a lower and an upper bound of the spatial-density correlation is defined[11]. If the upper bound of a trajectory τ is less than the lower bound of any other trajectories, τ cannot be the trajectory with the maximum spatial-density correlation to the query regions; thus, it can be pruned safely.

Once trajectory τ is scanned by the expansion from region center $c.m$ (e.g., τ_1 in Figure 3), we obtain the value of $d_M(c.m, \tau)$, and we estimate the upper and lower bounds of $sd(p_i, p)$ according to the characterization of triangle inequality for the shortest path, where $p_i \in c.V$ and $p \in \tau$ is the closest vertex to $c.m$ ($d_M(c.m, \tau) = sd(c.m, p)$). The triangle inequality in spatial networks is represented as follows.

$$sd(p, p') + sd(p', p'') > sd(p, p'')$$

$$sd(p, p') - sd(p', p'') < sd(p, p'')$$

Here, p, p', and p'' are vertices in G, and p'' is not on the shortest path between p and p'. Otherwise, we have that $sd(p, p') + sd(p', p'') = sd(p, p'')$.

In Figure 3, $p_3 \in \tau_1$ is the closest vertex to region center p_1 ($d_M(p_1, \tau_1) = sd(p_1, p_3)$), and p_6 is a vertex in c_1. According to the triangle inequality, we have $sd(p_6, p_3) < sd(p_1, p_3) + sd(p_6, p_1)$ and $sd(p_6, p_3) > sd(p_1, p_3) - sd(p_6, p_1)$. By substituting these inequalities into Equation 2, we derive the upper and lower bounds of $I(p_6, p_3)$ as follows.

$$e^{-sd(p_6,p_3)} > e^{-(sd(p_1,p_3)+sd(p_6,p_1))} = I(p_6, p_3).lb$$

$$e^{-sd(p_6,p_3)} < e^{-(sd(p_1,p_3)-sd(p_6,p_1))} = I(p_6, p_3).ub$$

Then we substitute $I(p_6, p_3).lb$ and $I(p_6, p_3).ub$ into Equation 4 and derive the upper and lower bounds of $C_{sd}(c_1, \tau_1)$.

$$C_{sd}(c_1, \tau_1).lb = \sum_{p_i \in c_1} p_i.g \cdot e^{-(d_M(p_1,\tau_1)+sd(p_i,p_1))} \quad (5)$$

$$C_{sd}(c_1, \tau_1).ub = \sum_{p_i \in c_1} p_i.g \cdot e^{-(d_M(p_1,\tau_1)-sd(p_i,p_1))} \quad (6)$$

If a trajectory is fully scanned (e.g., τ_1 in Figure 3), we compute the lower bound of its spatial-density correlation $C_{sd}(C, \tau_1).lb$ by substituting Equation 5 into Equation 4.

$$C_{sd}(C, \tau_1).lb = \sum_{c_i \in C} \left(\frac{2}{1 + e^{-C_{sd}(c_i,\tau).lb}} - 1\right). \quad (7)$$

Among all fully scanned trajectories, we define a global lower bound LB as

$$LB = \max_{\tau \in T_f} \{C_{sd}(C, \tau).lb\}, \quad (8)$$

11. In the following computation, we only consider the situations where $\forall c_i \in C(re_i < \epsilon)$. If $re_i \geq \epsilon$, according to Equations 2, for vertex p outside the browsed region, we have $I(p, c_i.m) = 0$.

where T_f is the set of fully scanned trajectories. Note that the value of LB changes dynamically during query processing.

If a trajectory τ has not been scanned by the expansion from region center p_1 (e.g., τ_3 in Figure 3), we have that $d_M(p_1, \tau) > re_1$. Hence, we can use the value of re_1 to replace that of $d_M(p_1, \tau)$ in Equation 6, and derive that

$$C_{sd}(c_1, \tau).ub = \sum_{p_i \in c_1} p_i.g \cdot e^{-(re_1 - sd(p_i, p_1))}, \quad (9)$$

where p_1 is the center of region c_1.

By merging Equations 6 and 9 into Equation 4, the upper bound of the spatial-density correlation $C_{sd}(C, \tau).ub$ is computed by

$$C_{sd}(C, \tau).ub = \sum_{c_i \in C} \left(\frac{2}{1 + e^{-C_{sd}(c_i, \tau).ub}} - 1\right). \quad (10)$$

$$C_{sd}(c_i, \tau).ub = \begin{cases} \sum_{p_j \in c_i} p_j.g \cdot e^{-(d_M(p_i, \tau) - sd(p_i, p_j))} & \text{if } C_1 \\ \sum_{p_j \in c_i} p_j.g \cdot e^{-(re_i - sd(p_i, p_j))} & \text{if } C_2 \end{cases}$$

C_1: τ is scanned by the expansion from $p_i = c_i.m$.
C_2: τ is not scanned by the expansion from $p_i = c_i.m$.

Among all fully scanned trajectories, we define a global upper bound UB as

$$UB = \min_{\tau \in T_f} \{C_{sd}(C, \tau).ub\}, \quad (11)$$

where T_f is the set of fully scanned trajectories. The trajectories in T_f are sorted according to their upper bounds. As for LB, the value of UB dynamically changes.

Note that to reduce the time and space costs, we only generate and update the lower bounds of fully scanned trajectories, because partially scanned and unscanned trajectories cannot have upper bounds of the spatial-density correlation that exceeds those of fully scanned trajectories (refer to Equation 10).

4.3 Filter and Refine

Once we have $LB > UB$, network expansions terminates. Fully scanned trajectories whose upper bound exceeds LB and all partially scanned and unscanned trajectories are pruned. The remaining trajectories in T_f are sorted according to $C_{sd}(C, \tau).ub$ and are refined from the maximum to the minimum. Intuitively, a trajectory τ with larger $C_{sd}(C, \tau).ub$ may have a higher probability of being the trajectory with the maximum spatial-density correlation to C.

For a trajectory $\tau \in T_f$, suppose $\{p_1, p_2, ..., p_i\} \in \tau$ are the vertices closest to region centers $\{c_1.m, c_2.m, ..., c_i.m\}$. We perform Dijkstra's expansion [6] from $\{p_1, p_2, ..., p_i\}$ to compute the network distances between p_i and vertices within region c_i. The value of $C_{sd}(C, \tau)$ is derived according to Equation 4. Once we have

$$\max_{\tau \in T_r} \{C_{sd}(C, \tau)\} \geq \max_{\tau' \in T_u} \{C_{sd}(C, \tau').ub\},$$

the refinement terminates, and the trajectory with the maximum $C_{sd}(C, \tau)$ is returned. Here T_r is the set of refined trajectories, T_u is the set of unrefined trajectories, and $T_r \cup T_u = T_f$.

4.4 Algorithm

Uniform-speed search is detailed in Algorithm 1. Initially, the global lower bound LB is set to 0, and the global upper bound UB is set to $+\infty$. Network expansion is performed from each center p_i of query regions in turn using Dijkstra's algorithm [6], which conducts expansion by selecting the vertex with the minimum distance label (lines 1–4). For each newly scanned trajectory τ, if it has not been scanned by the expansion from p_i, it is labeled as having been scanned by p_i, and we compute its spatial-density upper bound $C_{sd}(C, \tau).ub$ and lower bound $C_{sd}(C, \tau).lb$. If $C_{sd}(C, \tau).lb > LB$, LB is updated to $C_{sd}(C, \tau).lb$. Moreover, if $C_{sd}(C, \tau).ub < UB$, UB is updated to $C_{sd}(C, \tau).ub$ (lines 5–14). If $LB > UB$ or the search radius exceeds the threshold ϵ, network expansions terminate, and trajectories whose $C_{sd}(C, \tau).ub$ is less than LB are removed from T_f (lines 15–18). Trajectories in T_f are sorted according to the value of $C_{sd}(C, \tau).ub$ and are refined from the maximum to the minimum. Once $\max_{\tau \in T_r}\{C_{sd}(C, \tau)\} \geq \max_{\tau' \in T_u}\{C_{sd}(C, \tau').ub\}$, the refinement terminates, and the trajectory with the maximum spatial-density correlation is returned (lines 19–22).

Algorithm 1: Uniform-Speed Search

```
Data: graph G(V, E, F, W), trajectory set T, query region set C
Result: trajectory τ with the maximum value of C_sd(C, τ)
1  LB ← 0; UB ← +∞; T_f ← null;
2  while true do
3      for each query source p_i do
4          expand(p_i);
5          for each newly scanned trajectory τ do
6              if τ.scanned(p_i) = false then
7                  τ.scanned(p_i) ← true;
8                  if τ is fully scanned then
9                      T_f.add(τ);
10                     compute C_sd(C, τ).ub and C_sd(C, τ).lb;
11                     if C_sd(C, τ).lb > LB then
12                         LB ← C_sd(C, τ).lb;
13                     if C_sd(C, τ).ub < UB then
14                         UB ← C_sd(C, τ).ub;
15             if (LB > UB) ∨ (re ≥ ε) then
16                 for each τ ∈ T_f do
17                     if C_sd(C, τ).ub < LB then
18                         T_f.remove(τ);
19 for each trajectory τ ∈ T_f do
20     compute C_sd(C, τ);
21     if max_{τ∈T_r}{C_sd(C, τ)} ≥ max_{τ'∈T_u}{C_sd(C, τ').ub}
       then
22         return trajectory τ with max_{τ∈T_r}{C_sd(C, τ)};
```

4.5 Complexity Analysis

Uniform-speed search (USS) follows the filter-and-refine paradigm. In the filtering phase, we perform $|C|$ Dijkstra's expansions to explore the network. Thus, its time complexity is $O(|C|(|V| \log |V| + |E|))$, where $|C|$ is the number of query regions (a constant), and $|V|$ and $|E|$ are the vertex and edge counts in graph G.

In the refinement phase, we perform $|C|$ Dijkstra's expansions to refine each trajectory in candidate set T_f, yielding the time complexity $O(|C||T_f|(|V|\log|V| + |E|))$. In the worst case, $|T_f| = |T|$. When combining the two phases, the time complexity of uniform-speed search becomes $O(|C|(|V|\log|V| + |E|)) + O(|C||T_f|(|V|\log|V| + |E|)) = O(|T|(|V|\log|V| + |E|))$.

5 TSR Query Processing

The main drawback of the uniform-speed search (USS) is its lack of an effective scheduling strategy for multiple query sources, which may lead to poor performance. In addition, if the centers of two query regions are close to each other and both are selected as query sources, their search spaces may overlap substantially, which again decreases performance.

Motivated by this, we develop a best-expansion search (BES) algorithm. First, we reuse an existing query-source selection strategy [18] to select a set of query sources from the set of centers of query regions (Section 5.1). Second, we define new upper and lower bounds on the spatial-density correlation to enable pruning (Section 5.2). Third, we propose a heuristic search method to schedule expansion from the query sources effectively. We establish and maintain a dynamic priority ranking heap during query processing. At each step, we expand from the top-ranked query source until a new top-ranked query source appears (Section 5.3). The BES algorithm is detailed in Section 5.4. Compared to USS, the BES algorithm has two major advantages: (i) it further prunes the search space for avoiding the traversal of overlap areas; (ii) it is able to focus trajectories more likely to be the solution and to further improve the query performance.

5.1 Query-Source Selection Strategy

We reuse the query-source selection strategy from PNC query processing [18] (Equations 12–14). This strategy aims to reduce the search space during query processing.

Linear programming is adopted to select query sources from query region centers. We assume trajectories and query regions are uniformly distributed. Given a set C of query regions ($|C| = n$), $p_1, p_2, .., p_{n-1}$, and p_n are centers of query regions. Matrix M is an $n \times n$ matrix where $m_{ij} = 1$ if the centers of the i^{th} and the j^{th} regions are adjacent query sources. Otherwise, $m_{ij} = 0$. We estimate the area ω of search space in Equation 12, and our goal is to minimize the value of ω.

$$\omega = \frac{\pi}{4}\sum m_{ij} \cdot (sd_{ij} + 2\epsilon)^2, \tag{12}$$

where $i < j$, $\sum_{j=1}^{n} a_{0j} = 1$, $\sum_{i=1}^{n} a_{i0} = 1$, $\sum_{j=1}^{n} a_{ij} \leq 1$, $\sum_{i=1}^{n} m_{ij} \leq 1$, $\sum_{j=1}^{n} m_{ij} = \sum_{k=1}^{n} m_{ki}$. Here sd_{ij} is the network distance between the centers of the i^{th} and the j^{th} query regions. We use $\sum m_{ij}$ to estimate the number of query sources. Here ϵ is a threshold, and $\frac{sd_{ij}}{2} + \epsilon$ is the search radius of a query source in worst case. If the distance between two vertices exceeds this threshold, the influence factor between them is set to 0 (refer to Equation 2).

In an online scenario, Equation 12 can be simplified if we assume that the gap between any two adjacent query sources is the same. Subsequently, our objective is changed to find the optimal number of query sources.

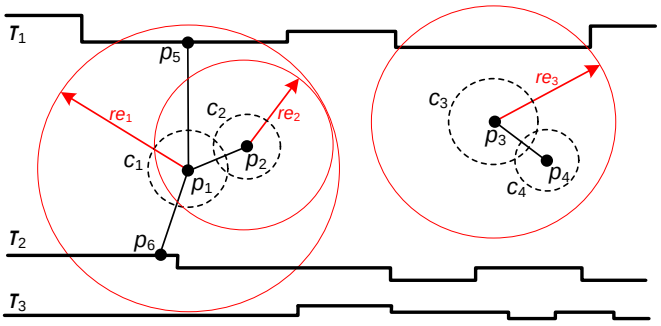


Fig. 4. An example of the best-expansion search

$$\omega(x) = \frac{\pi \cdot x}{4} \cdot (\frac{l \cdot (n-1)}{x-1} + 2\epsilon)^2, \tag{13}$$

where $l = sd(p_{n-1}, p_n)$ is the average network distance between two adjacent region centers. The value of x corresponding to the minimum $\omega(x)$ is acquired by means of the derivative of the function in Equation 13.

$$\omega(x)' = \frac{\partial \omega}{\partial x} = 0 \quad \Rightarrow$$

$$2l^2(n-1)^2(x-1)^{-3} + (l^2(n-1)^2 + 4\epsilon l(n-1))(x-1)^{-2} + 4\epsilon = 0 \tag{14}$$

We solve the cubic equation in Equation 14 by applying the general formula of roots, so x uniformly distributed region centers are selected as query sources. The need for performing the query-source selection depends on the value of x. If there is at most x region centers (i.e., $n \leq x$), query-source selection is not necessary. The query sources that define the minimum search space can be found by following the aforementioned procedure.

5.2 Upper and Lower Bounds

The region centers that are not selected are attached to their closest query sources. We estimate their upper and lower bounds correspondingly. An example is shown in Figure 4, where τ_1, τ_2, and τ_3 are trajectories, and c_1, c_2, c_3, and c_4 are query regions. Region centers p_1 and p_3 are selected as query sources, and p_2 and p_4 are attached to them. If a trajectory is fully scanned (e.g., τ_1), for a non-query-source region center p_m (e.g., p_2 and p_4), the upper and lower bound of its network distance to τ is estimated as follows.

$$d_M(p_m, \tau).lb = d_M(p_n, \tau) - sd(p_m, p_n) \tag{15}$$

$$d_M(p_m, \tau).ub = d_M(p_n, \tau) + sd(p_m, p_n) \tag{16}$$

Here, p_n is p_m's closest query source, so p_m is attached to p_n, and re_n is the expansion radius of p_n.

Consider the example in Figure 4, where τ_1 has been scanned by the expansion from p_1 and where $p_5 \in \tau_1$ is the closest vertex to p_1 ($d_M(p_1, \tau_1) = sd(p_1, p_5)$). Hence, τ_1 is tangent to a circular region that is defined by $(p_1, d_M(p_1, \tau_1))$ (p_1 is the center and $d_M(p_1, \tau_1)$ is the radius), and p_5 is

the tangent point. If $re_2 = d_M(p_1, \tau_1) - sd(p_1, p_2)$, we can guarantee that the circular region (p_2, re_2) is enclosed by the circular region $(p_1, d_M(p_1, \tau_1))$, and thus we have that $d_M(p_2, \tau_1) \geq re_2 = d_M(p_1, \tau_1) - sd(p_1, p_2)$. On the other hand, $SP(p_2, p_1) + SP(p_1, p_5)$ is a path from p_2 to τ_1; hence, we have that $d_M(p_m, \tau) \leq d_M(p_n, \tau) + sd(p_m, p_n)$.

We substitute Equation 15 into Equation 6 and Equation 16 into Equation 5, and we derive the upper and lower bounds of $C_{sd}(c_2, \tau_1)$.

$$C_{sd}(c_2, \tau_1).lb = \sum_{p_i \in c_2} p_i.g \cdot e^{-(d_M(p_1, \tau_1) + sd(p_1, p_2) + sd(p_i, p_2))} \quad (17)$$

$$C_{sd}(c_2, \tau_1).ub = \sum_{p_i \in c_2} p_i.g \cdot e^{-(d_M(p_1, \tau_1) - sd(p_1, p_2) - sd(p_i, p_2))} \quad (18)$$

Here, p_1 and p_2 are the centers of regions c_1 and c_2. Region center p_1 is a query source, p_2 is a non-query-source region, and p_2 is attached to p_1.

If a trajectory τ has not been scanned by the expansion from region center p_1 (e.g., τ_3 in Figure 4), we have that $d_M(p_1, \tau_3) > re_1$. Hence, we can use the value of re_1 to replace that of $d_M(p_1, \tau_3)$ in Equation 18, and derive that

$$C_{sd}(c_2, \tau_3).ub = \sum_{p_i \in c_2} p_i.g \cdot e^{-(re_1 - sd(p_1, p_2) - sd(p_i, p_2))}, \quad (19)$$

where p_1 is the center of region c_1.

By merging Equations 5 and 17 into Equation 7 and Equations 6, 18, and 19 into Equation 10, the lower and upper bounds of $C_{sd}(C, \tau_1)$ are computed.

5.3 Query-Source Scheduling

We proceed to introduce a heuristic scheduling strategy based on a priority ranking of the query sources, which is helpful to avoid devoting unnecessary search efforts to trajectories that are unlikely to be the optimal choice.

Consider the scenario in Figure 4. Trajectory τ_1 is fully scanned, while trajectories τ_2 is partially scanned, and τ_3 is unscanned. Each query source p_n is given a label $p.l$ to describe its priority. We maintain a dynamic priority heap ordered on $p.l$ that contains these query sources. In each step, we search the query source on the top of the heap until a new center takes its place. Then we search the new top-ranked query source. The priority of each query source $p.l$ is defined as follows.

$$p.l = |p.c| \sum_{\tau \in T_p \setminus T_s(p)} e^{C_{sd}(C, \tau).ub} \quad (20)$$

Here, $p.c$ is a cluster that contains query source p and all non-query-source region centers that have p as their closest query source, and $|p.c|$ is its size. T_p is the set of partially scanned trajectories (e.g., $T_p = \{\tau_2\}$) and $T_s(p)$ is a set of trajectories scanned by the expansion from p (e.g., $T_s(p_1) = \{\tau_1, \tau_2\}$, $T_s(p_3) = \{\tau_1\}$). Fully scanned trajectories (e.g., τ_1) and unscanned trajectories (e.g., τ_3) are not taken into account in this ranking model.

Fundamentally, the priority of p should reflect the size of cluster $|p.c|$. The larger it is, the higher the priority. We share the similar idea with the UOTS and PTM queries [16], [17], and we aim to transform partially scanned trajectories into fully scanned trajectories as quickly as possible because we only compute the lower bound of fully scanned trajectories (refer to Equation 7). To be fully scanned, a trajectory should be scanned by the expansions from all query sources. The priority of p is proportional to its "margin" (i.e., the size of $T_p \setminus T_s(p)$). For example, in Figure 4, $T_p = \{\tau_2\}$, $T_s(p_3) = \{\tau_1\}$, and $T_p \setminus T_s(p_3) = \{\tau_2\}$. As a result, the margin of p_3 is 1. Moreover, $C_{sd}(C, \tau).ub$ is used to estimate the spatial-density correlation between C and τ. Intuitively, a trajectory τ with larger $C_{sd}(C, \tau).ub$ may have a higher probability to be the trajectory with the maximum value of $C_{sd}(C, \tau)$. If $\tau \in T_p \setminus T_s(p)$, the value of $C_{sd}(C, \tau).ub$ should be proportional to p's priority.

5.4 Algorithm

Algorithm 2: Best-Expansion Search

Data: graph $G(V, E, F, W)$, trajectory set T, query region set C
Result: trajectory τ with the maximum value of $C_{sd}(C, \tau)$

1 $LB \leftarrow 0$; $UB \leftarrow +\infty$; $T_f \leftarrow null$;
2 select query sources;
3 $\forall p_i \in E_c$ ($p.l \leftarrow 0$);
4 $p \leftarrow E_c.top()$;
5 **while** *true* **do**
6 $expand(p)$;
7 **for** *each newly scanned trajectory* τ **do**
8 **if** $\tau.scanned(p_i)$ = *false* **then**
9 $\tau.scanned(p_i) \leftarrow$ *true*;
10 **if** τ *is fully scanned* **then**
11 $T_f.add(\tau)$;
12 compute $C_{sd}(C, \tau).ub$ and $C_{sd}(C, \tau).lb$;
13 update UB and LB;
14 **if** $(LB > UB) \vee (\forall p_i \in E_c(re_i \geq \epsilon + \frac{p.dist)}{2}))$ **then**
15 **for** *each* $\tau \in T_f$ **do**
16 **if** $C_{sd}(C, \tau).ub < LB$ **then**
17 $T_f.remove(\tau)$;
18 **if** $p \neq E_c.top()$ **then**
19 $p \leftarrow E_c.top()$;
20 **for** *each trajectory* $\tau \in T_f$ **do**
21 compute $C_{sd}(C, \tau)$;
22 **if** $\max_{\tau \in T_r}\{C_{sd}(C, \tau)\} \geq \max_{\tau' \in T_u}\{C_{sd}(C, \tau').ub\}$ **then**
23 **return** trajectory τ with $\max_{\tau \in T_r}\{C_{sd}(C, \tau)\}$;

Best-expansion search is detailed in Algorithm 2. Initially, the global lower bound LB is set to 0, and the global upper bound UB is set to $+\infty$. We select a set of query sources according to the method described in Section 5.1 (lines 1–2). The priority labels of all query sources are set to 0, and at each step, we search the top-ranked query source (the one with the maximum priority label), and the network expansion follows Dijkstra's algorithm [6] (lines 3–6). For each newly scanned trajectory τ, if it has not been scanned by the expansion from p, it is labeled as having been scanned by p. We then compute its spatial-density upper bound $C_{sd}(C, \tau).ub$ and lower bound $C_{sd}(C, \tau).lb$, and update UB and LB correspondingly (lines

7–13). If $LB > UB$ or all search radiuses exceed the value of $\epsilon + \frac{p.dist)}{2}$) ($p.dist = \max\{sd(p,p'), sd(p,p'')\}$, and p' and p'' are adjacent query sources of p), network expansions terminate, and the trajectories whose $C_{sd}(C,\tau).ub$ is less than LB are removed from T_f (lines 14–17). If p is not the top-ranked query source in E_c, the network expansion from p terminates, and we begin to search from the new top-ranked query source (lines 18–19). The refinement phase is similar to that of Algorithm 1. Trajectories in T_f are sorted according to the value of $C_{sd}(C,\tau).ub$ and are refined from the maximum to the minimum. Once $\max_{\tau\in T_r}\{C_{sd}(C,\tau)\} \geq \max_{\tau'\in T_u}\{C_{sd}(C,\tau').ub\}$, where T_r is a set of refined trajectories and T_u is a set of unrefined trajectories and $T_r \cup T_u = T_f$, the refinement terminates, and all unrefined trajectories are pruned. The trajectory with the maximum spatial-density correlation is returned (lines 20–23).

5.5 Complexity Analysis

Best-expansion search (BES) also follows the filter-and-refine paradigm. In the filtering phase, we perform x Dijkstra's expansions, getting a time complexity of $O(x(|V|\log|V|+|E|))$, where $x \leq |C|$ is the (constant) number of query sources (cf. Section 5.1), and $|V|$ and $|E|$ are the vertex and edge counts in graph G. Recall that the filtering phase of USS has a time complexity of $O(|C|(|V|\log|V|+|E|))$ and $x \leq |C|$.

In the refinement phase, we perform x Dijkstra's expansions to refine each trajectory in candidate set T'_f, which has a time complexity of $O(x|T'_f|(|V|\log|V| + |E|))$. Recall that the refinement phase of USS has a time complexity of $O(|C||T_f|(|V|\log|V| + |E|))$ and $x \leq |C|$ and $|T'_f| \leq |T_f|$ (cf. Section 5.1).

When combining the two phases, the time complexity of best-expansion search is $O(x(|V|\log|V| + |E|)) + O(x|T'_f|(|V|\log|V| + |E|)) = O(|T|(|V|\log|V| + |E|))$. In the worst case, the time complexity of BES is the same as that of USS. However, in experiments on real data sets, we shall see that BES outperforms USS by a factor of 2–3 due to its pruning capabilities (cf. Tables III and IV).

6 Extension

In some scenarios, users may specify a preferred visiting sequence for the query regions of interest. In that case, the order of the regions needs to be taken into account. Here, the proposed uniform-speed search and best-expansion search algorithms are extended to cover this case.

Given a sequence of query regions $C = \langle c_1, c_2, ..., c_i\rangle$ and a trajectory $\tau = \langle v_1, v_2, ..., v_j\rangle$, where the vertices $\{p_1, p_2, ..., p_i\}$ are the centers of regions $\{c_1, c_2, ..., c_i\}$, the spatial-density correlation between C and τ is defined recursively [17] as follows.

$$C'_{sd}(c,v) = \sum_{\forall p\in c} p.g \cdot e^{-sd(p,v)} \tag{21}$$

$$C'_{sd}(C,\tau) = \max\begin{cases} (\frac{2}{1+e^{-C'_{sd}(C.head,\tau.head)}} - 1) \\ +C'_{sd}(C.tail,\tau) \\ \\ C'_{sd}(C,\tau.tail), \end{cases} \tag{22}$$

where $C'_{sd}(c,v)$ is the spatial-density correlation between region c and vertex $v \in \tau$, $*.head$ is the first item of $*$, (e.g., $C.head = c_1$ and $\tau.head = v_1$) and $*.tail$ indicates the list obtained by removing $*.head$ from $*$ (e.g., $C.tail = \langle c_2, c_3, ..., c_i\rangle$ and $\tau.tail = \langle v_2, v_3, ..., v_j\rangle$). In Equation 22, the value of $C'_{sd}(c,v)$ is normalized to the range of $[0,1]$.

The lower bound of $C'_{sd}(c,v)$ is estimated as follows. If the center of region c is a query source, $C'_{sd}(c,v).lb$ is derived by merging Equation 21 into Equation 5.

$$C'_{sd}(c,v).lb = \sum_{\forall p\in c} p.g \cdot e^{-(sd(p,c.m)+sd(c.m,v))} \tag{23}$$

Otherwise, if $c.m$ is not a query source, we substitute Equation 21 into Equation 17 and $C'_{sd}(c,v).lb$ is derived by

$$C'_{sd}(c,v).lb = \sum_{p\in c} p.g \cdot e^{-(sd(p',v)+sd(c.m,p')+sd(p,c.m))}, \tag{24}$$

where $c.m$ is attached to its closest query source p'. By merging Equations 23 and 24 into Equation 22, the lower bound of $C'_{sd}(C,\tau)$ is derived.

$$C'_{sd}(C,\tau).lb = \max\begin{cases} (\frac{2}{1+e^{-C'_{sd}(C.head,\tau.head).lb}} - 1) \\ +C'_{sd}(C.tail,\tau) \\ \\ C'_{sd}(C,\tau.tail) \end{cases} \tag{25}$$

If all vertices in τ have been visited by the expansions from all query sources, we get the network distances between each $c.m \in E_c$ and each $v_i \in \tau$. Hence, we can compute the value of $C'_{sd}(C,\tau).lb$. This type of trajectory is fully scanned. Other trajectories are partially scanned (i.e., part of its vertices have been scanned) or unscanned (i.e., no vertex has been scanned). Among all fully scanned trajectories, the global lower bound is defined by

$$LB' = \max_{\tau\in T_f}\{C'_{sd}(C,\tau).lb\}. \tag{26}$$

For a region c, if its center $c.m$ is a query source, the upper bound of $C'_{sd}(c,v)$ is estimated as follows.

$$C'_{sd}(c,v).ub = \begin{cases} \sum_{p\in c} p.g \cdot e^{-(sd(c.m,v)-sd(p,c.m))} & \text{if } C_1 \\ \\ \sum_{p\in c} p.g \cdot e^{-(re-sd(p,c.m))} & \text{if } C_2 \end{cases} \tag{27}$$

C_1: τ is scanned by the expansion from $c.m$.
C_2: τ is not scanned by the expansion from $c.m$.

If $c.m$ is a query source, the value of $C'_{sd}(c,v)$ is computed as follows.

$$C'_{sd}(c,\tau).ub = \begin{cases} \sum_{p\in c} p.g \cdot e^{-(sd(p',v)-sd(c.m,p')-sd(p,c.m))} & \text{if } C_1 \\ \\ \sum_{p\in c} p.g \cdot e^{-(re'-sd(c.m,p')-sd(p,c.m))} & \text{if } C_2 \end{cases} \tag{28}$$

C_1: τ is scanned by the expansion from p'.
C_2: τ is not scanned by the expansion from p'.

Here $c.m$ is attached to its closest query source p', and re' is the expansion radius of p'. By merging Equations 27 and 28 into Equation 22, the upper bound of $C'_{sd}(C,\tau)$ is derived.

$$C'_{sd}(C,\tau).ub = \max \begin{cases} (\frac{2}{1+e^{-C'_{sd}(C.head,\tau.head).ub}} - 1) \\ +C'_{sd}(C.tail,\tau) \\ \\ C'_{sd}(C,\tau.tail) \end{cases} \quad (29)$$

Among all fully scanned trajectories, the global upper bound is defined by

$$UB' = \min_{\tau \in T_f} \{C'_{sd}(C,\tau).ub\}. \quad (30)$$

By merging Equation 29 into Equation 20, the priority label of query source p is defined by

$$p.l = |p.c| \sum_{\tau \in T_p \setminus T_s(p)} e^{C'_{sd}(C,\tau).ub}, \quad (31)$$

where T_p is a set of partially scanned trajectories.

The TSR query processing for the query regions with a sequence is conducted by merging Equations 21–31 into Algorithms 1 and 2.

7 Experimental Study

We report on extensive experiments with real and synthetic spatial data sets that offer insight into the efficiency and scalability of the proposed algorithms.

7.1 Settings

We use graphs extracted from two spatial networks, namely the Beijing Road Network (BRN) and the North America Road Network (NRN)[12], which contain 28,342 vertices and 27,690 edges, and 17,813 vertices and 179,179 edges. The graphs are indexed by adjacency lists. For BRN, we use a real trajectory data set of Beijing taxis and a real data set of points of interest (spatial objects), which contain 800,000 trajectories and 300,000 POIs. Raw POIs have longitude and latitude. They are mapped to the spatial network and assigned to their nearest vertices. For each vertex p in BRN, we record the number of objects having it as their nearest vertex. Therefore, we are not required to access individual spatial objects during TSR query processing. We share the POI setting with a previous study [18]. For NRN, larger synthetic data is used to study scalability. NRN contains 4,000,000 trajectories. For each vertex p' in NRN, we derive the number of attached spatial objects, and we store this number as one of its attributes. We have 1,800,000 derived spatial objects. In BRN, the default distance threshold ϵ (refer to Equation 2) is set to 10 km, while in NRN, it is set to 200 km by default.

In the experiments, the graphs are stored in memory when running Dijkstra's algorithm [6], as the memory cost of BRN and NRN is less than 20 MB. All algorithms are implemented in Java and run on a Windows 8 platform with an Intel Core i7-3520M Processor (2.90 GHz) and 8 GB memory. All experimental results are averaged over 20 independent trails with different inputs. We evaluate CPU time and the count of visited trajectories. The count of visited trajectories is selected as an evaluation metric since it captures the amount of data accesses.

12. http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm

Table II: Parameter Settings

	BRN	NRN
Number of trajectories $\|T\|$	500,000–800,000 /default 600,000	1,000,000–4,000,000 /default 1,000,000
Trajectory length $\tau.l$	10–50 vertices /default 20 vertices	50–200 vertices /default 100 vertices
Number of regions $\|C\|$	2–10 /default 6	2–10 /default 6
Radius of region $c.r$	2 km–10 km /default 6 km	50 km–250 km /default 150 km
ϵ	3 km–15 km /default 10 km	30 km–200 km /default 150 km

Table III: Pruning Effectiveness

	USS	BES	BES-w/o-h
Pruning ratio (BRN)	0.32	0.76	0.70
Candidate ratio (BRN)	0.68	0.24	0.30
Pruning ratio (NRN)	0.37	0.69	0.63
Candidate ratio (NRN)	0.63	0.31	0.37

Table IV: Pruning Effectiveness for Sequential Scenarios

	USS	BES	BES-w/o-h
Pruning ratio (BRN)	0.23	0.71	0.62
Candidature ratio (BRN)	0.77	0.29	0.38
Pruning ratio (NRN)	0.25	0.64	0.58
Candidature ratio (NRN)	0.75	0.36	0.42

The parameter settings are presented in Table II. By default, the trajectory set size was set to 600,000 in BRN and 1,000,000 in NRN, the trajectory length (the number of vertices in a trajectory) was set to 20 in BRN and 100 in NRN, and the number of query regions was set to 6 in both BRN and NRN. The average radius of the query regions varies from 2 km to 10 km in BRN (default 6 km) and from 50 km to 250 km in NRN (default 150 km). The best-expansion search algorithm (Section 5) is denoted by BES, the uniform-speed search (Section 4) denoted is by USS, and the best-expansion search algorithm without heuristic scheduling strategies is denoted by BES-w/o-h.

7.2 Pruning Effectiveness

First, we investigate the pruning effectiveness of the three algorithms with default settings. The experimental results are shown in Tables III (TSR query) and IV (TSR query for sequential scenarios, Section 6), and the pruning ratio and candidate ratio are defined as follows.

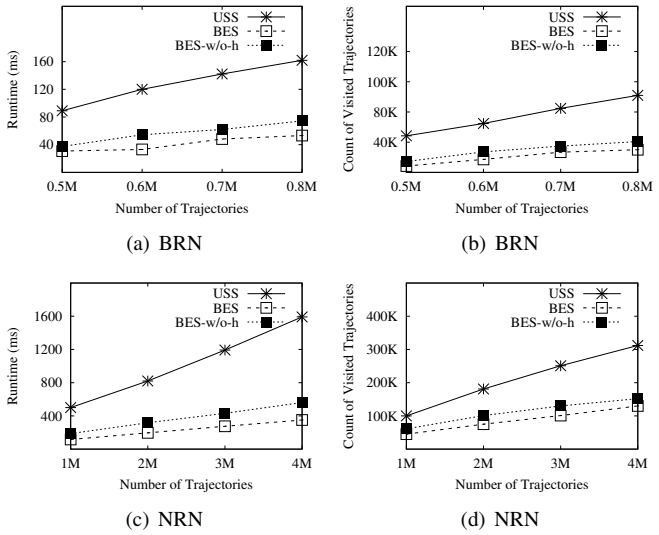
$$\textit{Candidate ratio} = \frac{|T_f|}{|T|}$$

$$\textit{Pruning ratio} = 1 - \textit{Candidate ratio}$$

Here, T_f is the candidate set and T is the trajectory set (cf. Section 4.3). By comparing the pruning and candidate ratios of BES-w/o-h to those of USS, we see that the pruning effectiveness is improved by approximately a factor of 2–3 with the support of the new upper and lower bounds based on the query-source selection strategy (cf. Section 5.1), which prune the search space and to avoid the traversal of overlap areas. Then, we compare the pruning and candidate ratios

of BES-w/o-h to those of BES and find that the pruning effectiveness is improved by a factor of 1.2–1.3 with the support of the heuristic scheduling strategy (cf. Section 5.3), which enables TSR search to focus on trajectories more likely to be the optimal choice.

7.3 Effect of the Number of Trajectories



(a) BRN (b) BRN

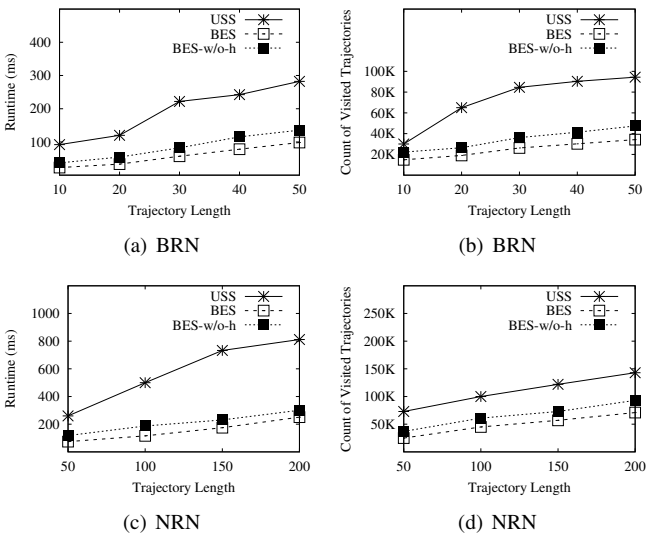
(c) NRN (d) NRN

Fig. 5. Effect of the number of trajectories

Figure 5 presents the performance of the algorithms when varying the number of trajectories $|T|$. Intuitively, a larger $|T|$ causes more trajectories to be processed and yields a larger search space. As a result, both the CPU time and the count of visited trajectories are expected to be higher for all three algorithms. However, Figure 5 shows that the BES algorithm is able to outperform the USS algorithm by almost a factor of 8 in NRN (for both CPU time and visited trajectories). The upper and lower bounds (cf. Section 5.2) based on the query-source selection strategy can improve the efficiency of USS by a factor of 4–6, and the heuristic scheduling strategy can further improve the efficiency by a factor of 2, in terms of both CPU time and the count of visited vertices. It is worth noting that the CPU time is not fully aligned with the count of visited trajectories. To prune the search space, the algorithms need more computational effort to maintain their bounds. In some cases, the increased computation cost may offset the benefits of the reduction in the count of visited trajectories.

7.4 Effect of Trajectory Length

Next, we vary the trajectory length $\tau.l$. Longer trajectories cause more sample points (vertices) to be processed. Consequently, the CPU time and the count of visited vertices are expected to increase for all three algorithms. Figure 6 shows that the CPU time and the count of visited vertices of the USS algorithm increase much faster than those of the BES algorithm. This occurs because USS lacks effective query-source selection and scheduling strategies. For example, with the trajectory length $\tau.l = 200$ in NRN, the BES algorithm



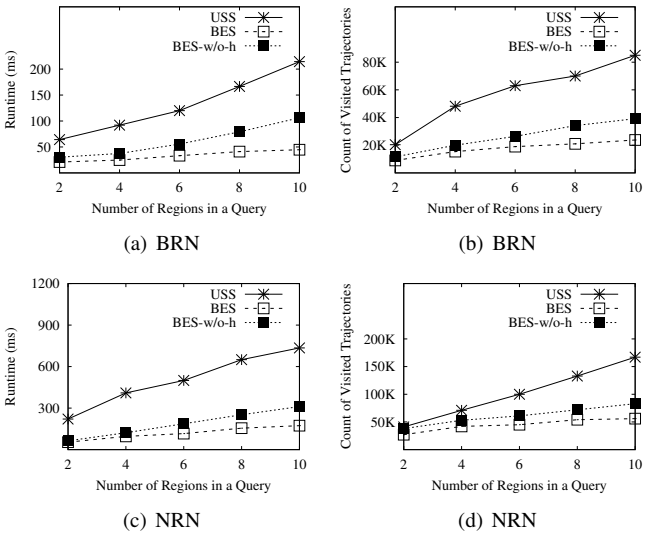
(a) BRN (b) BRN

(c) NRN (d) NRN

Fig. 6. Effect of trajectory length

outperforms the USS algorithm by almost a factor of 4 (for both CPU time and visited trajectories).

7.5 Effect of the Number of Regions



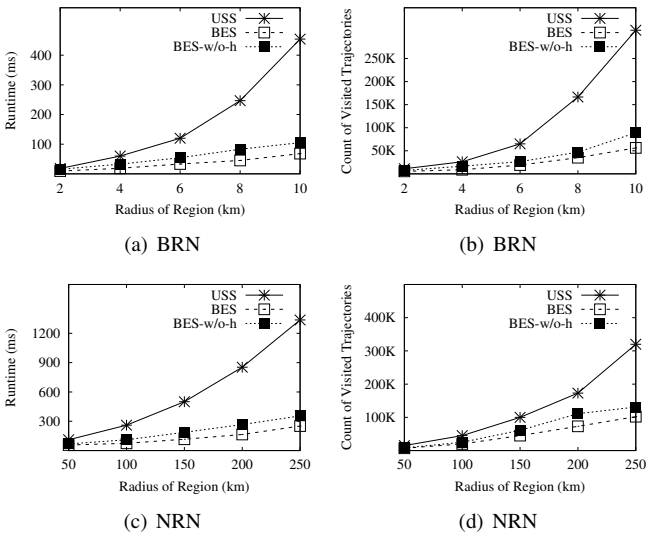
(a) BRN (b) BRN

(c) NRN (d) NRN

Fig. 7. Effect of the number of regions

Figure 7 covers the effect of varying the number of query regions $|C|$. More query regions cause more query sources to be processed and have a larger search space. Thus, the CPU time and the count of visited trajectories for all three algorithms increase with $|C|$, and the increase of the USS algorithm is much faster than those of the BES algorithms. The CPU time and the count of visited trajectories required by the USS algorithm are at least 4–6 times higher than those needed by the BES algorithm. The heuristic search strategy improves the efficiency by almost a factor of 2 (for both CPU time and visited trajectories).

7.6 Effect of the Region Radius



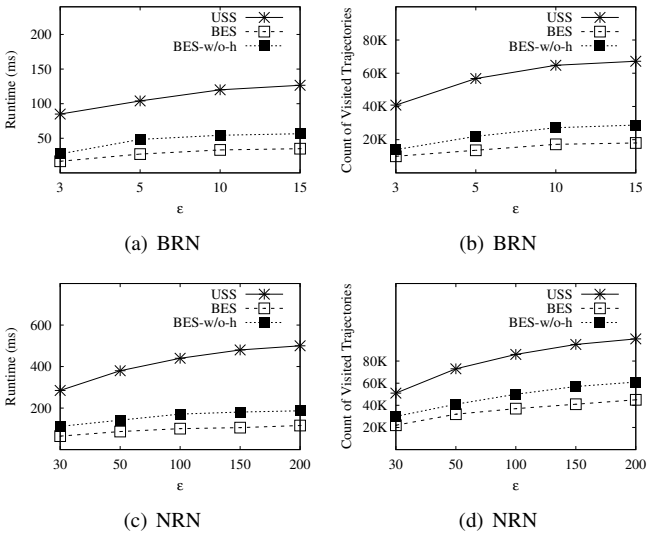
(a) BRN (b) BRN

(c) NRN (d) NRN

Fig. 8. Effect of the region radius

Figure 8 shows the effect of varying region radius $c.r$ on the efficiency of the algorithms. A larger value of $c.r$ means that more spatial objects have to be processed and implies a larger search space; thus, more CPU time and visited trajectories are required. When $c.r$ =10 km in BRN, BES outperforms USS by almost an order of magnitude (for both CPU time and visited trajectories).

7.7 Effect of ϵ



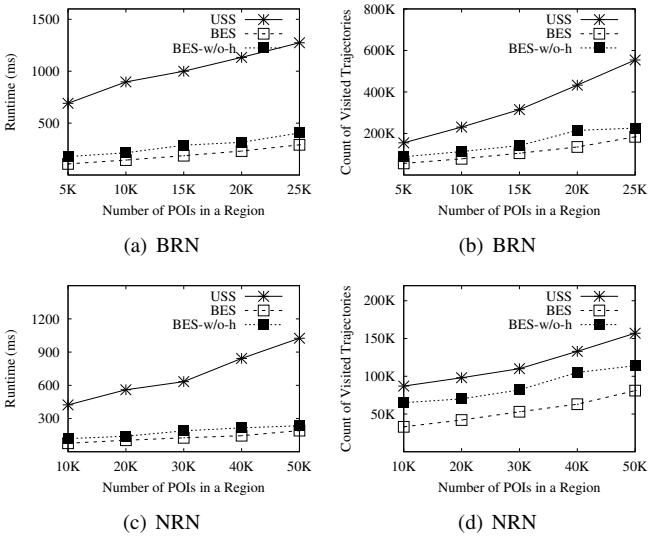
(a) BRN (b) BRN

(c) NRN (d) NRN

Fig. 9. Effect of the ϵ

Figure 9 shows the effect of varying threshold ϵ on the efficiency of the algorithms. A larger ϵ implies a larger search space with more trajectories to be processed; thus, more CPU time and trajectories are involved. The effect on query efficiency is negligible when $\epsilon \geq 10$ km in BRN and $\epsilon \geq 150$ km in NRN.

7.8 Effect of Object Density



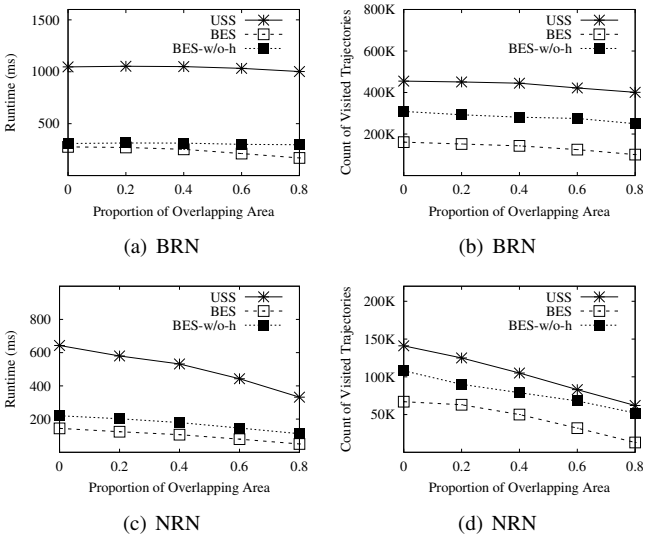
(a) BRN (b) BRN

(c) NRN (d) NRN

Fig. 10. Effect of object density

We study the effect of object density on query performance. In BRN, the number of spatial objects in each region varies from 5,000 to 25,000, while in NRN, the number of spatial objects in each region varies from 10,000 to 50,000. The number of query regions and region radiuses are both with default settings (see Table II). Intuitively, a higher object density results in higher computation cost and more trajectories to be visited (Equation 3). As a result, both the CPU time and the count of visited trajectories are expected to be higher for all three algorithms. In Figure 10, we see that the BES algorithm outperforms the USS algorithm by almost an order of magnitude (for both CPU time and visited trajectories).

7.9 Effect of Region Overlap



(a) BRN (b) BRN

(c) NRN (d) NRN

Fig. 11. Effect of region overlap

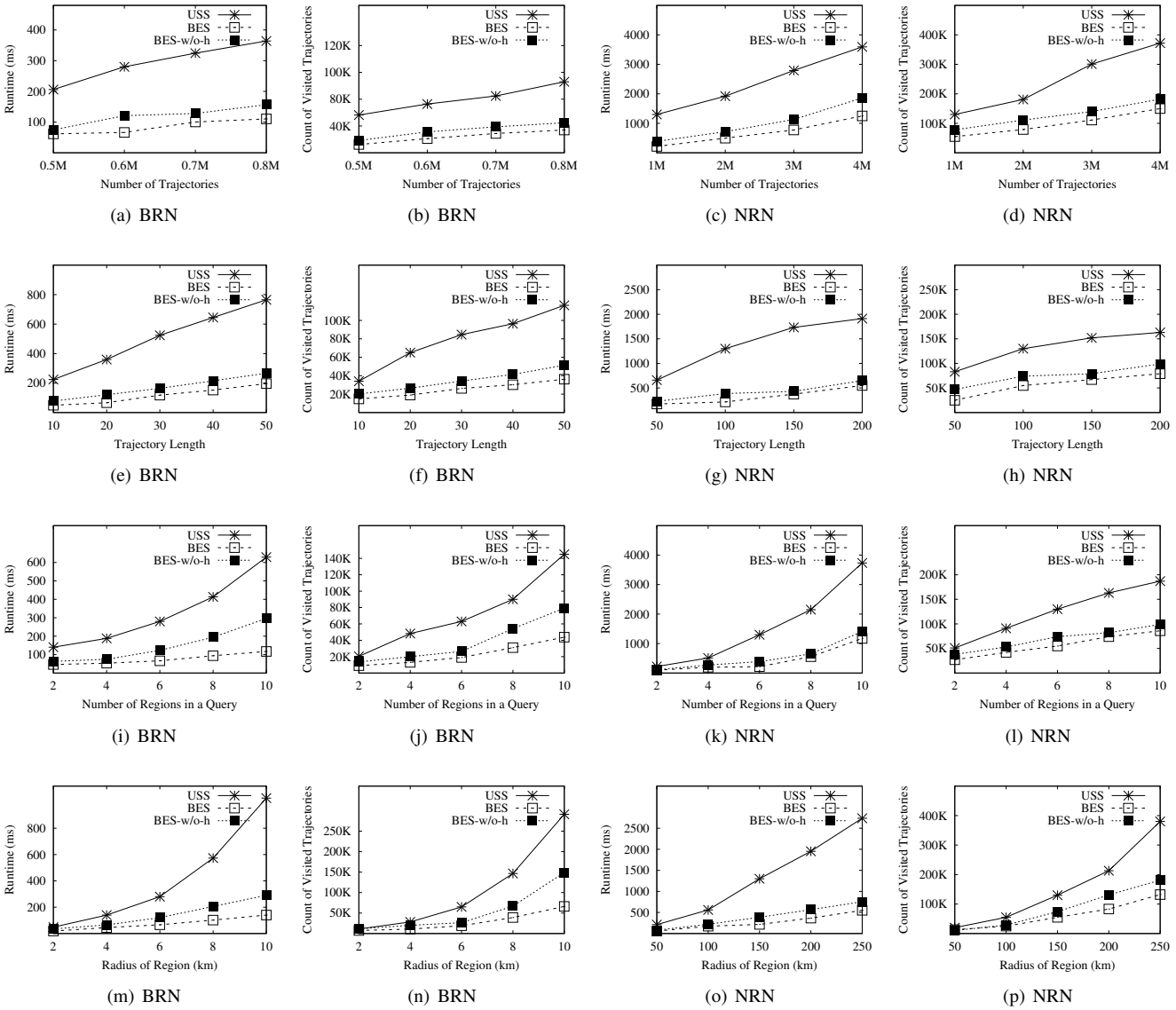


Fig. 12. Performance for the sequential TSR query

We study the effect of region overlap on query performance. The proportion of overlapping area is defined as follows.

$$\textit{Proportion of Overlapping Area} = \frac{\textit{Overlapping Area}}{\textit{Total Area}}$$

A higher proportion of overlapping area means that query regions are closer to each other, which is equivalent to reducing the number of query regions. Thus, the CPU time and the count of visited trajectories are expected to be lower. In Figure 11, when *Proportion of Overlapping Area* $= 0.8$, the BES algorithm is able to process the query in 100 ms (for both BRN and NRN). Moreover, the query-source selection strategy (cf. Section 5.1) demonstrates its effectiveness as we increase the proportion of overlapping area.

7.10 Performance of the Sequential TSR Query

We conducted experiments to study the performance of processing the sequential TSR query (where query regions are ordered, cf. Section 6). Compared to the original TSR query, the sequential TSR query needs more computational efforts to compute the upper and lower bounds, due to the more complex distance measures. In addition, a query source may not be matched with the closest vertex on a trajectory, which consequently requires a scan of more trajectory vertices to get the best match. Therefore, more query time and trajectory accesses are incurred. However, the trend of Figure 12 is still similar to that of the original TSR query in Figures 6–9. In Figure 12, the BES algorithm can still outperform the USS algorithm by a factor of 4–6 times in terms of both CPU time and the count of visited trajectories.

8 Conclusion and Future Directions

We propose and study a novel problem, namely trajectory search by regions of interest (TSR query), that finds the trajectory with the highest spatial-density correlation to a

sequence of query regions. Compared to existing studies of trajectory search by locations, we take the concept of query region and the density of spatial objects into account. This type of query is useful in many popular applications such as trip planning and recommendation, and location based services in general. To compute the TSR query efficiently, we develop a best-expansion search algorithm that exploits upper and lower bounds to prune the search space and adopts a query-source selection strategy, as well as a heuristic search strategy based on priority ranking to schedule multiple query sources. The performance of the TSR query was investigated through extensive experiments on both real and synthetic spatial data.

Three directions for future research are promising. First, users may assign different significance for different query regions, making it of interest to take the significance of query regions into account. The upper and lower bounds, query-source selection strategy, and the heuristic search strategy must be reworked correspondingly. Second, it is of interest to take temporal information into account and further extend the TSR query into a spatiotemporal query. The resulting query aims to find the trajectory with the highest spatial-temporal-density correlation to the query regions. Third, it is of interest to study how to effectively split and combine trajectories in order to return better results.

REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, pages 69–84, 1993.

[2] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, pages 853–864, 2005.

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

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

[5] 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.

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

[7] E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Algorithms for nearest neighbor search on moving object trajectories. *Geoinformatica*, 11(2):159–193, 2007.

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

[9] 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.

[10] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul. Ecomark: evaluating models of vehicular environmental impact. In *SIGSPATIAL/GIS*, pages 269–278, 2012.

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

[12] E. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417, 2002.

[13] B. Lin and J. Su. Shapes based trajectory queries for moving objects. In *ACM GIS*, pages 21–30, 2005.

[14] K. Liu, Y. Li, F. He, J. Xu, and Z. Ding. Effective map-matching on the most simplified road network. In *SIGSPATIAL GIS*, pages 609–612, 2012.

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

[16] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis. User oriented trajectory search for trip recommendation. In *EDBT*, pages 156–167, 2012.

[17] S. Shang, R. Ding, K. Zheng, C. S. Jensen, P. Kalnis, and X. Zhou. Personalized trajectory matching in spatial networks. *VLDB J.*, 23(3):449–468, 2014.

[18] S. Shang, K. Zheng, C. S. Jensen, B. Yang, P. Kalnis, G. Li, and J. Wen. Discovery of path nearby clusters in spatial networks. *IEEE Trans. Knowl. Data Eng.*, 27(6):1505–1518, 2015.

[19] L. A. Tang, Y. Zheng, X. Xie, J. Yuan, X. Yu, and J. Han. Retrieving k-nearest neighboring trajectories by a set of point locations. In *SSTD*, pages 223–241, 2011.

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

[21] Y. Yanagisawa, J. Akahani, and T. Satoh. Shape-based similarity query for trajectory of mobile objects. In *Mobile Data Management*, pages 63–77, 2003.

[22] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang. Stochastic skyline route planning under time-varying uncertainty. In *ICDE*, pages 136–147, 2014.

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

[24] P. Zarchan. Global positioning system theory and applications. In *Progress in Astronautics and Aeronautics*, volume 163, pages 1–781, 1996.

[25] K. Zheng, S. Shang, N. J. Yuan, and Y. Yang. Towards efficient search for activity trajectories. In *ICDE*, pages 230–241, 2013.

[26] K. Zheng, B. Zheng, J. Xu, G. Liu, A. Liu, and Z. Li. Popularity-aware spatial keyword search on activity trajectories. *World Wide Web*, 19(6):1–25, online first, 2016.

Shuo Shang is a research scientist at KAUST and a professor of computer science at China University of Petroleum-Beijing. He was a research assistant professor at department of computer science, Aalborg University, Denmark. He obtained his Ph.D. in computer science from The University of Queensland, Australia. His research interests include efficient query processing in spatiotemporal databases, spatial trajectory computing, and location based social media. He has served as PC member, session chair, and invited reviewer for many prestigious conferences and journals, including SIGMOD, ICDE, TKDE, The VLDB Journal, ACM TIST, IEEE TITS, ACM TSAS, GeoInformatica, KAIS, WWW Journal, DKE, JCST, and IEICE Transactions.

Lisi Chen is an assistant professor of computer science at Hong Kong Baptist University. He obtained his Ph.D. in computer science from Nanyang Technological University. His research interests include geo-textual data management, spatial keyword query evaluation, and location based social networks.

Christian S. Jensen is Obel Professor of Computer Science at Aalborg University, Denmark. He was recently at Aarhus University for three years and at Google Inc. for one year. His research concerns data management and data-intensive systems, and its focus is on temporal and spatiotemporal data management. Christian is an ACM and an IEEE fellow, and he is a member of the Academia Europaea, the Royal Danish Academy of Sciences and Letters, and the Danish Academy of Technical Sciences. He is editor-in-chief of ACM Transactions on Database Systems and was an editor-in-chief of The VLDB Journal from 2008 to 2014.

Ji-Rong Wen is a professor at Renmin University of China. He is also a National “1000 Plan” Expert of China. His main research interest lies on Web big data management, information retrieval, data mining and machine learning. He was a senior researcher at MSRA, and he has 50+ U.S. patents in Web search and related areas. He has published extensively on prestigious international conferences and journals. He is currently an associate editor of TKDE and TOIS. He is an IEEE senior member.

Panos Kalnis is a professor at KAUST. He received his Diploma in Computer Engineering from the Computer Engineering and Informatics Department, University of Patras, and PhD from HKUST. His research interests include Database outsourcing and Cloud Computing, Mobile Computing, and Spatiotemporal and High-dimensional Databases. He is an associate editor of TKDE, and The VLDB Journal.