Discovery of Path Nearby Clusters in Spatial Networks

Shuo Shang, Kai Zheng, Christian S. Jensen, *Fellow, IEEE*, Bin Yang,
Panos Kalnis, Guohe Li, and Ji-Rong Wen, *Senior Member, IEEE*

Abstract—The discovery of regions of interest in large cities is an important challenge. We propose and investigate a novel query called the path nearby cluster (PNC) query that finds regions of potential interest (e.g., sightseeing places and commercial districts) with respect to a user-specified travel route. Given a set of spatial objects O (e.g., POIs, geo-tagged photos, or geo-tagged tweets) and a query route q, if a cluster c has high spatial-object density and is spatially close to q, it is returned by the query (a cluster is a circular region defined by a center and a radius). This query aims to bring important benefits to users in popular applications such as trip planning and location recommendation. Efficient computation of the PNC query faces two challenges: how to prune the search space during query processing, and how to identify clusters with high density effectively. To address these challenges, a novel collective search algorithm is developed. Conceptually, the search process is conducted in the spatial and density domains concurrently. In the spatial domain, network expansion is adopted, and a set of vertices are selected from the query route as expansion centers. In the density domain, clusters are sorted according to their density distributions and they are scanned from the maximum to the minimum. A pair of upper and lower bounds are defined to prune the search space in the two domains globally. The performance of the PNC query is studied in extensive experiments based on real and synthetic spatial data.

Index Terms—Path nearby cluster, efficiency, optimization, spatial networks, spatiotemporal databases

1 Introduction

The continued proliferation of GPS-equipped mobile devices [44] (e.g., vehicle navigation systems and smart phones) and the proliferation of online map-based services (e.g., Google Maps,[1] Bing Maps,[2] and MapQuest[3]) enable people to acquire their current geographic positions in real time and to retrieve spatial information relevant to their trips. In this paper, we aim to provide fundamental geographic functionality that is relevant to a range of services. Specifically, we propose and investigate a novel query, the path nearby cluster (PNC) query, that identifies clusters of spatial objects with respect to a user-specified travel route. Spatial objects can be points of interest, e.g., supermarkets, restaurants, and fashion shops, or they can be geo-tagged photos, micro-blog posts, and tweets from location-based social media, such as Flickr,[4] Weibo,[5] Twitter,[6] and Foursquare.[7]

Given a set of spatial objects O, a cluster is a subset of objects in O that contains at least a threshold number of objects and such that the objects fall within a circular region with a radius that does not exceed a threshold radius. Given a query route q, the PNC query intuitively returns a cluster that is both dense and close to q. For example, when planning a travel route in an unfamiliar city (e.g., when traveling overseas), travelers may wish to know about potential regions of interest (e.g., sightseeing places, commercial districts, dining area) along their route. Intuitively, clusters with high spatial-object density (e.g., POIs, geo-tagged photos, or geo-tagged tweets) are assumed to be more attractive to travelers than low-density clusters. Also, a cluster located close to their route is more accessible and thus more attractive than a further-away cluster. The PNC query aims to find potential regions of interest along the travel route and recommend them to travelers. We believe that this query can benefit users in many popular applications such as trip planning and location (region) recommendation.

To the best of our knowledge, this is the first work that investigates the path nearby cluster query in spatial

[1] http://maps.google.com/
[2] http://www.bing.com/maps/
[3] http://www.mapquest.com
[4] http://www.flickr.com/
[5] http://weibo.com/
[6] https://twitter.com/
[7] https://foursquare.com/

- *S. Shang is with the State Key Laboratory of Petroleum Resources and Prospecting, and the Department of Computer Science, China University of Petroleum, Beijing, P.R. China, and the Key Laboratory of Data Engineering and Knowledge Engineering MOE, Renmin University of China, P.R. China. E-mail: jedi.shang@gmail.com.*
- *K. Zheng is with the School of Information Technology & Electrical Engineering, The University of Queensland, Brisbane, Australia. E-mail: kevinz@itee.uq.edu.au.*
- *C.S. Jensen and B. Yang are with the Department of Computer Science, Aalborg University, DK-9220 Aalborg East, Denmark. E-mail: {csj, byang}@cs.aau.dk.*
- *P. Kalnis is with King Abdullah University of Science and Technology, Saudi Arabia. E-mail: panos.kalnis@kaust.edu.sa.*
- *G. Li is with State Key Laboratory of Petroleum Resources and Prospecting, and the Department of Computer Science, China University of Petroleum, Beijing, P.R. China. E-mail: ligh@cup.edu.cn.*
- *J.-R. Wen is with the Key Laboratory of Data Engineering and Knowledge Engineering MOE, Renmin University of China, P.R. China. E-mail: jirong.wen@gmail.com.*

Manuscript received 20 July 2014; revised 5 Dec. 2014; accepted 12 Dec. 2014. Date of publication 17 Dec. 2014; date of current version 27 Apr. 2015.
Recommended for acceptance by G. Yu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2014.2382583

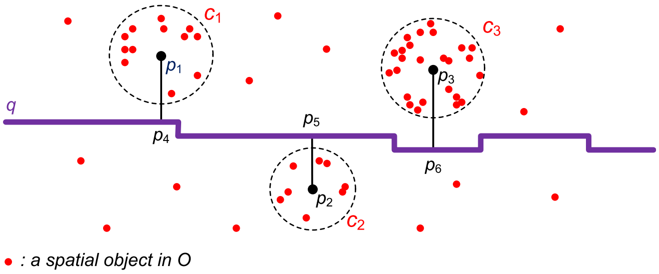


Fig. 1. An example of path nearby cluster query.

networks while taking spatial-object density into account. Previous studies (e.g., the Path Nearest Neighbor (NN) query [6]) use spatial distance as the sole factor when computing results and assume that a query route is a network shortest path. In contrast, the PNC query takes both spatial proximity and density into account and supports arbitrary query routes. A linear combination method (e.g., [5], [8]) is adopted to combine the spatial and density domains, and a distance-and-density evaluation score is defined correspondingly; thus, the PNC query is also different from skyline queries (e.g., [3], [24]).

An example PNC query is shown in Fig. 1. Here, q is a query route, and spatial objects are distributed in the spatial network. Vertices p_1, p_2, and p_3 are center points of clusters c_1, c_2, and c_3, respectively. In the query route q, vertices p_4, p_5, and p_6 are the closest vertices to p_1, p_2, and p_3, respectively. The distance between a cluster and a query route is the shortest spatial network distance between the cluster center and the route (e.g., $dist(c_2, q) = dist(p_2, p_5)$). If only the spatial distance is considered (e.g., as in related work [6]), c_2 is the cluster closest to q. However, when considering also the density of spatial objects, c_2 is less attractive than c_3 due to its sparser spatial-object distribution. Specifically although c_3 is not as good as c_2 according to spatial distance, we consider c_3 as the best choice for location (region) recommendation when taking both spatial distance and cluster density into account.

The proposed PNC query is applied in spatial networks, since in a large number of practical scenarios, users (e.g., pedestrians and vehicles) move in such networks (e.g., roads and railways) rather than in a euclidean space. Here, a spatial network is modeled by a connected and undirected graph G, and a query route is a path in G. Users can specify a query route on a digital map, or they can consult travel histories of other users and select a travel route accordingly. For each vertex p in G, we maintain the number of spatial objects that have p as their nearest vertex as an attribute of p. When formulating a cluster c, two system-defined thresholds are considered. The one is the cluster size threshold $\tau.s$, which defines the minimum cardinality of a cluster (i.e., a cluster must contain at least $\tau.s$ spatial objects). The other is the cluster radius threshold $\tau.r$, which defines the maximum spatial extent of a cluster ($c.r \leq \tau.r$, where $c.r$ is the radius of a cluster c, i.e., the largest network distance from the cluster center to the cluster boundary). These two thresholds define the minimum requirements of clusters that qualify for a PNC query.

To enable efficient processing of the PNC query, density based pre-computation is conducted. For each vertex p in G, we pre-compute the number of spatial objects that are covered by a circular region defined by $(p, \tau.r)$, where p is the center and $\tau.r$ is the radius. These counts are useful in pruning the search space during query processing. Based on pre-computation, we develop two algorithms to process the PNC query efficiently. Spatial-first cluster search (SF for short) is a straightforward approach that uses network expansion (i.e., Dijkstra's expansion [10]) to explore the spatial network and uses the two end vertices of a query route as expansion centers. A pair of distance-based upper and lower bounds are proposed to prune the search space. For the clusters in the explored region, we compute exact densities and distances to the query route. By combining these results, the best cluster in terms of density and location relative to the query route is found. The main drawback of SF is that the search space can be large. Density-based bounds are not effective at pruning the search space in both the spatial and density domains, which results in the algorithm having to process a large number of clusters. As a result, the PNC query can often not be answered in interactive time with SF.

To the best of our knowledge, there is no existing approach that can compute the PNC query efficiently. We note that PNC finds clusters with the highest density in the density domain, while existing online clustering methods (e.g., DBScan [11]) formulate clusters based on an estimated density distribution (i.e., they find clusters whose densities are higher than the estimated density); thus, they are not suitable for PNC query processing.

To achieve better performance than does SF, an adaptive collective cluster search algorithm is developed. Conceptually, the search process is conducted in the spatial and density domains concurrently. In the spatial domain, network expansion is adopted to explore the spatial network, and we carefully select a set of vertices from the query route as expansion centers. In the density domain, clusters are sorted according to density, and they are considered from maximum to minimum. A pair of upper and lower bounds are defined for both distance and density to prune the search space in the two domains globally. Compared to SF, the collective algorithm has smaller search space and avoids devoting unnecessary search efforts to clusters that cannot be the query results.

To sum up, the main contributions are as follows:

- We define a new path nearby cluster query according to a proposed spatial-and-density ranking function. It provides new spatial functionality and holds the potential to benefit users of popular mobile applications such as trip planning and location recommendation.
- We propose a set of new metrics to evaluate the distance-and-density evaluation score of clusters.
- We develop an adaptive collective search algorithm to process the PNC query efficiently with the support of upper and lower bounds.
- We conduct extensive experiments on real and synthetic data to investigate the performance of the proposed algorithms.

The rest of the paper is organized as follows. Section 2 introduces spatial networks, query routes, and the distance metrics used in the paper; and it also gives problem

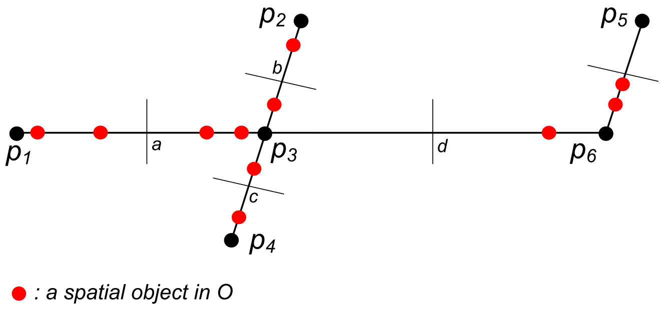


Fig. 2. An example of spatial object pre-processing.

definitions. A baseline method is introduced in Section 3, and collective PNC query processing is covered in Section 4, which is followed by a coverage of experimental results in Section 5. Related work is covered in Section 6, and conclusions are drawn in Section 7.

2 Preliminaries

2.1 Network Modeling and Preprocessing

A spatial network is modeled as 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 $v_i \in V$ represents a road intersection or an end 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 travel time [12] or fuel consumption [14], [40], which may be obtained by mining historic traffic data. Given two vertices p_a and p_b in a spatial network, the network shortest path between them (i.e., a sequence of edges linking p_a and p_b where the accumulated weight is minimal) is denoted by $SP(p_a, p_b)$, and its length is denoted by $sd(p_a, p_b)$. When weights model aspects such as travel time and fuel consumption, the lower bound of network distance is not necessarily the corresponding euclidean distance; thus spatial indexes such as the R-tree [15] are not effective.

Spatial objects may not be located on edges or vertices in a spatial network. We assume that all spatial objects have already been mapped to the spatial network (on edges or vertices) according to some map-matching algorithm [2], [4], [13], [22], [23], [37]. Then, each spatial object is attached to its nearest vertex. For each vertex $p \in G.V$, the number of spatial objects that are attached to p is maintained as an attribute of p, denoted by $p.g$. A vertex and its attached spatial objects make up the minimum unit in spatial-object density computations, and thus we do not need to access individual spatial objects during PNC query processing.

An example of this pre-processing is shown in Fig. 2. Here, p_1,$p_2, \ldots, p_6$ are vertices in $G.V$, and spatial objects are mapped to the edges and vertices. Each spatial object is attached to its nearest vertex according to network distance. We see that four spatial objects are attached to p_3; thus $p_3.g = 4$. The region of the minimum unit of vertex p is also called its control region, denoted by $p.cr$. In Fig. 2, the control region of vertex p_3 intersects its four adjacent edges and that it is defined by (a, p_3), (b, p_3), (c, p_3), and (d, p_3), where a, b, c, and d are the centers of edges (p_1, p_3), (p_2, p_3), (p_3, p_4), and (p_3, p_6), respectively. The size of p_3's control region is called its control area, denoted by $p_3.ca$, and $p_3.ca$ is computed as follows

$$p_3.ca = \frac{1}{2} \sum_{e \in p_3.adj} W(e). \tag{1}$$

Here, $p_3.adj$ is a set of adjacent edges of vertex p_3, and $W(e)$ is the weight of edge e. As an example, in Fig. 2, the value of $p_3.ca$ is calculated by

$$p_3.cr = \frac{1}{2}(W(p_1, p_3) + W(p_2, p_3) + W(p_3, p_4) + W(p_3, p_6)).$$

2.2 Evaluation Functions and Problem Definition

We define a query route as follows.

Definition (Query Route). *A query route is a sequence of vertices* $\langle p_1, p_2, \ldots, p_n \rangle$, *where* p_i *and* p_{i+1} *are adjacent vertices in* G, $i = 1, 2, \ldots, n-1$.

Given a query route q and a vertex p in a spatial network, the distance $d(p, q)$ between them is defined as

$$d(p, q) = \min_{p_i \in q}\{sd(p, p_i)\}, \tag{2}$$

where p_i is a vertex belonging to q.

Given a set of spatial objects O, a cluster is a subset of objects in O that contains at least a threshold $\tau.s$ number of objects and such that the objects fall within a circular region with a radius that does not exceed a threshold radius $\tau.r$. The circular region that is defined by two parameters, a cluster center $c.m$ and a cluster radius $c.r$, where $c.m$ is a vertex in $G.V$ and $c.r$ is the distance from $c.m$ to the cluster boundary. A cluster c has an associated subgraph $c._G$, which contains the vertices $c._V$ and edges $c._E$ from G that are in the circular region.

The density $c.\rho$ of cluster c is defined as

$$c.\rho = \frac{\sum_{p \in c._V} p.g}{\sum_{e \in c._E} W(e)}, \tag{3}$$

where p is a vertex in $c._V$ and $p.g$ is the number of spatial objects that are attached to p; and e is an edge in $c._E$ and $W(e)$ is its weight.

Given a cluster c and a query route q, a distance-evaluation function $E_s(c, q)$ and a density-evaluation function $E_d(c)$ are defined in Equations (4) and (5). A Sigmoid function [25] is adopted here to normalize the values of $E_s(c, q)$ and $E_d(c)$ to the range $[0, 1]$

$$E_s(c, q) = \frac{2}{1 + e^{-d(c.m, q)}} - 1, \tag{4}$$

$$E_d(c) = \begin{cases} \frac{2}{1+e^{c.\rho}} - 1 & \text{if} \quad c.r \leq \tau.r \wedge c.s \geq \tau.s, \\ 1 & \text{if} \quad c.r > \tau.r \vee c.s < \tau.s. \end{cases} \tag{5}$$

Here, $\tau.r$ and $\tau.s$ are cluster radius and size thresholds, respectively.

TABLE 1
A List of Notions

Notion	Description
$G.V$	the set of vertices in graph G
$G.E$	the set of edges in graph G
$W(e)$	the weight of edge e
$p.g$	the number of spatial objects attached to p
$p.cr$	the control region of p
$c.m$	the center of cluster c
$c.\rho$	the spatial object density of cluster c
$\tau.r$	the cluster radius threshold
$\tau.s$	the cluster size threshold
$sd(p_a, p_b)$	network distance between vertices p_a and p_b
$d(p, q)$	network distance between vertex p and trajectory q
$E_s()$,$E_d()$,$E_{sd}()$	distance, density, and distance-and-density evaluation functions
UB, LB	global upper and lower bounds

By combining Equations (4) and (5), the distance-and-density evaluation function $E_{sd}(c, \tau)$ is defined as:

$$E_{sd}(c, \tau) = \lambda \cdot E_s(c, \tau) + (1 - \lambda) \cdot E_d(c), \tag{6}$$

where parameter $\lambda \in [0, 1]$ is used to adjust the relative importance of the density and the distance. We allow users to adjust the parameter λ at the query time.

Table 1 offers an overview of the notation used in the paper.

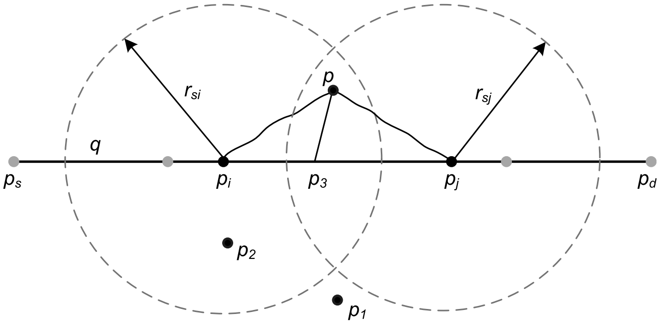
Problem Definition. *Given a set of spatial objects O, a query route q, a cluster size threshold $\tau.s$ and a radius threshold $\tau.r$, and an importance parameter λ, the path nearby cluster query finds the cluster $c \subseteq O$ with $c.s \geq \tau.s$ and $c.r \leq \tau.r$ such that $\forall c' \subseteq O(c' \neq c \wedge c'.s \geq \tau.s \wedge c'.r \leq \tau.r \Rightarrow E_{sd}(c, q) \leq E_{sd}(c', q))$.*

3 Baseline Method

3.1 Basic Idea

To enhance the performance of path nearby cluster query processing, a density based pre-computation technique is applied. For each vertex $p \in G.V$, we pre-compute the number of spatial objects that are covered by the circular region defined by $(p, \tau.r)$, where p is the center point and $\tau.r$ is the radius. Region $(p, \tau.r)$ is the maximum range of a cluster centered at p, and the number of spatial objects covered by $(p, \tau.r)$ is denoted by $p.max$. These counts are useful in estimating upper and lower bounds of density-evaluation scores (Equation (5)) and in pruning the search space during PNC query processing. It is possible to generalize the use of one radius and count to the use of multiple radiuses and counts. Then an application has a choice of possible parameter settings for the radius each time a user invokes the functionality through the application. For simplicity, we assume a single radius and corresponding count in the sequel.

Spatial-First (denoted by SF for short) is a straightforward approach to computing the PNC query, which includes two steps. First, it explores the spatial network and finds clusters located close to the query route and estimates their density-evaluation scores. Second, it uses a pair of distance-based upper and lower bounds to constrain a cluster candidate set and refines each candidate by computing its exact density and distance to the query route. By integrating these results, the cluster with the minimum distance-and-density evaluation score is found.



vertices p_i and p_j are expansion centers

Fig. 3. An example of spatial-first cluster search.

3.2 Upper and Lower Bounds

Consider the example in Fig. 3. Path q is a query route, and vertices p_i and p_j are adjacent expansion centers in the query route. In SF, we use the source vertex p_s and the destination vertex p_d of the query route q as expansion centers ($p_i = p_s$ and $p_j = p_d$, as in PNN query processing [6]). To explore the spatial network and find clusters close to the query route (i.e., with the smaller values of $d(p, q)$, refer to Equation (2)), Dijkstra's expansion [10] is adopted. From each expansion center, network expansion is performed using Dijkstra's algorithm, and their expansion speeds are the same. The explored circular regions are shown in Fig. 3, where the radiuses are the shortest network distances from expansion centers p_i and p_j to the corresponding expansion boundaries, respectively, denoted as rs_i and rs_j. Here, $rs_i = rs_j$ since the expansion speeds are the same.

To prune the search space in the spatial domain, a pair of upper and lower bounds are developed to estimate the minimum network distance from a vertex p to the query route q. An example is shown in Fig. 3, where vertex p has been scanned by the network expansions from expansion centers p_i and p_j. Thus, the network distances $sd(p, p_i)$ and $sd(p, p_j)$ are known. The upper bound of $d(p, q)$ is estimated as

$$d(p, q).ub = \min\{sd(p, p_i), sd(p, p_j)\}. \tag{7}$$

We estimate the lower bound of $d(p, q)$ based on the triangle inequality of the shortest-path distance. The triangle inequality in spatial networks is represented as follows

$$sd(v_1, v_2) + sd(v_2, v_3) > sd(v_1, v_3),$$
$$sd(v_1, v_2) - sd(v_2, v_3) < sd(v_1, v_3).$$

Here, v_1, v_2, and v_3 are vertices in $G.V$, and v_2 is not on the shortest path from v_1 to v_3. Otherwise, we have that $sd(v_1, v_2) + sd(v_2, v_3) = sd(v_1, v_3)$.

In Fig. 3, we assume that $p_3 \in q$ is the vertex closest to p (i.e., $d(p, q) = sd(p, p_3)$). According to the triangle inequality, we have the following inequalities

$$sd(p, p_3) > sd(p_i, p) - sd(p_i, p_3),$$

$$sd(p, p_3) > sd(p_j, p) - sd(p_j, p_3). \tag{8}$$

The path between p_i and p_j along q is not necessary a shortest path (different from [6], which only supports the shortest path), and its distance is denoted as $d(p_i, p_j)$. Thus, $d(p_i, p_3) \geq sd(p_i, p_3)$ and $d(p_j, p_3) \geq sd(p_j, p_3)$. Substituting them into Equation (8), the lower bound of $d(p, q)$ becomes:

$$d(p, q).lb = \frac{sd(p_i, p) + sd(p_j, p) - d(p_i, p_j)}{2}. \tag{9}$$

Cluster c is a cluster centered at vertex p (i.e., $c.m = p$). By substituting Equations (7) and (9) into Equation (4), the upper and lower bounds of the distance-evaluation score $S_{dist}(c, q)$ are as follows

$$\begin{aligned} & d(p, q).ub = \min\{sd(p, p_i), sd(p, p_j)\} \geq d(p, q) \\ & \Rightarrow E_s(c, q).ub = \frac{2}{1 + e^{-\min\{sd(p,p_i), sd(p,p_j)\}}} - 1 \end{aligned} \tag{10}$$

$$\begin{aligned} & d(p, q).lb = \frac{sd(p_i, p) + sd(p_j, p) - d(p_i, p_j)}{2} \leq d(p, q) \\ & \Rightarrow E_s(c, q).lb = \frac{2}{1 + e^{-(sd(p_i,p) + sd(p_j,p) - d(p_i,p_j))/2}} - 1. \end{aligned} \tag{11}$$

Here, c is a cluster centered at vertex p, and p has been scanned by network expansions from p_i and p_j.

On the other hand, if a vertex is partly scanned in the spatial domain (i.e., it is scanned by only one network expansion, e.g., vertex p_2 in Fig. 3), we have that $sd(p_j, p_2) > rs_j$ (where rs_j is the network distance from p_j to the corresponding expansion boundary), since Dijkstra's algorithm always selects the vertex with the minimum distance for expansion. Then, we use the value of rs_j to replace that of $sd(p_j, p)$ in Equation (9). The lower bound of $d(p_2, q)$ is computed as follows

$$d(p_2, q).lb = \frac{sd(p_i, p_2) + rs_j - d(p_i, p_j)}{2}. \tag{12}$$

Here, p_2 is a partly scanned vertex in the spatial domain. Network expansions from different expansion centers are at the same speed, thus we have $rs = rs_i = rs_j$.

Cluster c' is a cluster centered at vertex p_2 (i.e., $c'.m = p_2$). By substituting Equation (12) into Equation (4), the lower bound of $E_s(c', q)$ is computed as

$$E_s(c', q).lb = \frac{2}{1 + e^{-(sd(p_i,p_2) + rs - d(p_i,p_j))/2}} - 1, \tag{13}$$

where c' is a cluster whose center vertex $c.m$ is partly scanned in the spatial domain (e.g., $c.m = p_2$). Among all partly scanned vertices in the spatial domain, we define a global lower bound LB as

$$LB = \min\{E_s(c', q).lb\}, \tag{14}$$

where LB is a dynamic value and is continuously updated during PNC query processing.

To reduce the computation and storage load, we only compute and maintain the lower bounds of partly scanned vertices in the spatial domain. Unscanned vertices (i.e., vertices outside the scanned region, e.g., vertex p_1 in Fig. 3) must not have distance upper bounds that exceed those of partly scanned vertices. For instance, in Fig. 3, vertex p_2 is a partly scanned vertex, and p_1 is an unscanned vertex in the spatial domain. We use the value of rs to replace that of $sd(p_i, p)$ and $sd(p_j.p)$ in Equation (9). The lower bound of $d(p_1, q)$ is estimated by

$$d(p_1, q).lb = \frac{2rs - d(p_i, p_j)}{2} = rs - \frac{d(p_i, p_j)}{2}.$$

Because $\frac{sd(p_i,p_2)+rs_j-d(p_i,p_j)}{2} < \frac{2rs-d(p_i,p_j)}{2}$, we have that $d(p_2, q).lb < d(p_1, q).lb$, where clusters c_1 and c_2 are centered at vertices p_1 and p_2, respectively. By substituting the values of $d(p_2, q).lb$ and $d(p_1, q).lb$ into Equation (4), we have that $E_s(c_2, q).lb < E_s(c_1, q).lb$.

Once a vertex p is scanned by the network expansions from both p_i and p_j (e.g., p in Fig. 3), we compute the upper and lower bounds of its distance to the query route q according to Equations (10) and (11), respectively. Then, we estimate the spatial-object density for the clusters centered at vertex p. The number of spatial objects covered by the circular region $(p, \tau.r)$ is pre-computed, denoted by $p.max$ (refer to Section 3.1). If the value of $p.max$ is less than the cluster size threshold $\tau.s$, the clusters centered at p can be pruned safely. Otherwise, the density for clusters centered at p is estimated as follows

$$c_h.\rho = \max_{c.m=p \wedge c.r<\tau.r \wedge c.s>\tau.s} \{c.\rho\} \geq \frac{p.max}{\sum_{\forall e \in (p,\tau.r)} e.w}.$$

Here, c is a cluster centered at p (i.e., $c.m = p$), and c_h is the cluster with the highest density among all qualifying clusters centered at p. According to Equation (5), the upper bound of the density-evaluation score is computed as follows

$$E_d(c_h) < \frac{2}{1 + e^{(p.n/\sum e.w)}} - 1 = E_d(c_h).ub. \tag{15}$$

By combining the upper bounds of the distance-evaluation score $E_s(c, q).ub$ and the density-evaluation score $E_d(c_h).ub$, the upper bound of the distance-and-density evaluation score $E_{sd}(c, q).ub$ is computed as

$$E_{sd}(c_h, q).ub = \lambda \cdot E_s(c, q).ub + (1 - \lambda) \cdot E_d(c_h).ub, \tag{16}$$

where c_h is the cluster with the highest density centered at p.

Among all vertices that have been scanned by the network expansions from both p_i and p_j, we define a global upper bound UB as follows

$$UB = \min\{E_{sd}(c_h, q).ub\}, \tag{17}$$

where the center of cluster c_h is a vertex that has been scanned by the network expansions from both p_i and p_j. Similar to LB, UB changes dynamically during PNC query processing.

3.3 Filter and Refinement

Once the value of LB exceeds that of UB, the network expansion in the spatial domain terminates, and the vertices outside the scanned region can be pruned safely (i.e., the clusters centered at such outside vertices cannot be the cluster with the minimum distance-and-density evaluation score). Each qualifying vertex p (i.e., $p.max \geq \tau.s$) in the scanned region is put into the candidate set, and then we refine each candidate vertex p in two steps: (1) computing its exact network distance to the query route q, and (2) finding the cluster with the highest density among all clusters centered at p. By integrating these results, the cluster with the highest distance-and-density evaluation score is found.

Step 1. We use Dijkstra's algorithm [10] for network expansion to compute the network distance $d(p, q)$ between a vertex p and a query route q. Dijkstra's algorithm always selects the vertex with the smallest weight for expansion, so the first vertex $v \in q$ scanned by the network expansion from p is the vertex closest to p, and $d(p, q) = sd(p, v)$.

Step 2. To find the cluster with the highest density among all clusters centered at p, we expand from p according to Dijkstra's algorithm. The subgraph in the scanned region is expanded step by step, and at each step, we compute and record its current density as

$$c.\rho = \frac{\sum_{p \in c.V} p.g}{\sum_{e \in c.E} e.w}, \tag{18}$$

where $c.\rho$ is the density of the region scanned so far, p is a vertex in the scanned region, and $\sum p.g$ is the number of spatial objects covered by the scanned region. Once the radius of the scanned region exceeds the cluster-radius threshold $\tau.r$, network expansion terminates, and the cluster with the highest density is returned.

3.4 Analysis

Correctness. The PNC query retrieves the cluster with the minimum distance-and-density evaluation score. The SF algorithm follows the "filter-and-refinement" paradigm. We define a global upper bound UB and a global lower bound LB (refer to Equations (14) and (17)) of the distance-and-density evaluation score to prune the search space. When the value of LB exceeds that of UB, the search terminates. Clusters outside the scanned regions have lower bounds that exceed the upper bounds of the clusters inside the scanned region. Thus, they cannot be the cluster with the minimum distance-and-density score, and they can be pruned safely. Then, we refine the clusters inside the scanned region by computing their exact distance-and-density evaluation scores, which yields the result. Because (1) the clusters pruned in the filtering cannot be a solution and because (2) the computation in the refinement is accurate, the SF algorithm computes the PNC query correctly.

Time complexity. In the filtering, we perform network expansion according to Dijkstra's algorithm [10] to constrain a candidate set, and the time complexity of the filtering is $O(|V|log(|V|) + |E|)$, where $|V|$ is the number of vertices in G, and $|E|$ is the number of edges in G. In the refinement, for each candidate, we compute its distance and density scores using Dijkstra's algorithm. In the worst case, the number of candidates is equal to the number of vertices $|V|$, and the time complexity of the refinement process is then $O(|V|^2 log(|V|) + |V||E|)$. The time complexity of the SF algorithm is then $O(|V|log(|V|) + |E|) + O(|V|^2 log(|V|) + |V| |E|) = O(|V|^2 log(|V|) + |V||E|)$.

Space complexity. In the spatial domain, we maintain two expansion trees to maintain the distance labels, and the space complexity of the spatial domain is $O(|V|)$. In the density domain, no additional information is maintained. Thus, the SF algorithm needs $O(|V|)$ additional space to maintain the related information, excluding the $O(|V| + |E|)$ space occupied by the spatial network.

3.5 Algorithm

The Spatial-First method is detailed in Algorithm 1. The query inputs are a graph $G(V, E)$, a set of spatial objects O, and a query route q, while the query output is the cluster c with the minimum value of $E_{sd}(c, q)$. The two end vertices of the query route are set to be expansion centers (line 2), and network expansion is performed from each center in turn using Dijkstra's algorithm [10], which always selects the vertex with the smallest weight for expansion (line 4). For each newly scanned vertex p in the spatial domain, we compute the upper bound of distance-and-density evaluation score $E_{sd}(c_h, q).ub$ according to Equation (16), where c_h is the cluster with the highest density among all qualifying clusters centered at p (lines 5-7). If the value of $E_{sd}(c_h, q).ub$ is less than that of global upper bound UB, the value of UB is updated to that of $E_{sd}(c_h, q).ub$ (lines 8-9). Then, we compute the value of $E_s(c_h, q).lb$ according to Equation (13). If the value of $E_s(c_h, q).lb$ is less than that of global lower bound LB, the value of LB is updated to that of $E_s(c_h, q).lb$ (lines 10-12). If the value of LB exceeds that of UB, the search process in the spatial domain terminates.

Algorithm 1. Spatial-First Cluster Search

```
   Data: graph $G(V, E)$, spatial-object set $O$, query route $q$
   Result: cluster $c$ with the minimum value of $E_{sd}(c, q)$
 1 $LB \leftarrow 0; UB \leftarrow +\infty; CS \leftarrow null; minE_{sd} \leftarrow 1;$
 2 $p_i \leftarrow q.s; p_j \leftarrow q.d;$
 3 while true do
 4     for each expansion center $p_i \in q$ do
 5         $p \leftarrow expand(p_i);$
 6         if $c_h.m = p$ then
 7             compute $E_{sd}(c_h, q).ub;$
 8             if $E_{sd}(c_h, q).ub < UB$ then
 9                 $UB \leftarrow E_{sd}(c, q).ub;$
10             compute $E_s(c, q).lb;$
11             if $E_s(c_h, q).lb < LB$ then
12                 $LB \leftarrow E_s(c_h, q).lb;$
13             if $LB > UB$ then
14                 break;
15         if $(p, \tau.r).s \geq \tau.s$ then
16             $CS.add(p);$
17 for each vertex $p \in CS$ do
18     compute $E_s(c_h, q)$, $E_d(c_h)$, and $E_{sd}(c_h, q);$
19     if $E_{sd}(c_h, q) < minE_{sd}$ then
20         $minE_{sd} \leftarrow E_{sd}(c_h, q);$
21 return the cluster $c$ with $minE_{sd};$
```

If the number of spatial objects covered by circular region $(p, \tau.r)$ (this number is pre-computed as explained in Section 3.1) is less than the cluster size threshold $\tau.s$, there cannot exist any clusters centered at p that meet the size and radius thresholds at the same time; thus vertex p can be pruned. Otherwise, vertex p is put into the candidate set CS (lines 15-16). For each vertex in the candidate set CS, we compute the exact value of $E_{sd}(c_h, q)$ $(c_h.m = p)$ according to the procedure introduced in Section 3.3. By integrating these results, the cluster with the minimum value of $E_{sd}(c_h, q)$ is found (lines 17-21).

4 Collective Cluster Search

The main drawback of spatial-first is that the loose upper and lower bounds are unable to constrain the search space effectively. Density-based bounds are not effective at pruning the search space in both the spatial and density domains, which results in a large number of clusters having to be considered. Thus, the PNC query can generally not be answered in interactive time with SF.

This motivates the development of an adaptive collective search algorithm that conducts the search process in the spatial and density domains concurrently. In the spatial domain, network expansion is adopted to explore the spatial network, and a set of vertices on the query route are selected as expansion centers. In the density domain, clusters are sorted according to their density, and they are scanned from the maximum to the minimum. A pair of upper and lower bounds are defined to prune the search space in the two domains globally. Section 4.1 develops the upper and lower bounds, and the expansion-center selection strategy is given in Section 4.2. Compared to SF, the collective search algorithm reduces the search space and avoids devoting unnecessary search efforts to clusters that cannot be query results. We analyze the correctness, the time and space complexity of the collective algorithm in Section 4.3. The collective algorithm is detailed in Section 4.4, and the proposed techniques are extended to support a top-k PNC query in Section 4.5.

4.1 Upper and Lower Bounds

The collective cluster search is conducted in the spatial and density domains concurrently. In the spatial domain, we select a set of vertices from the query route as expansion centers, and we explore the spatial network using Dijkstra's algorithm [10] (as in SF). The expansion-center selection strategy is detailed in Section 4.2. In the density domain, we establish a heap H to maintain the lower bounds of the density-evaluation score $E_d(c).lb$. The items in H are sorted from the minimum to the maximum. Given a cluster c, the upper bound of its density is estimated as

$$c.\rho.ub = \frac{p.max}{p.ca} \geq c.\rho, \tag{19}$$

where vertex p is the center point of cluster c (i.e., $c.m = p$), $p.max$ is the number of spatial objects covered by a circular region $(p, \tau.r)$, and $p.ca$ is the control area of vertex p (see Equation (1)). The value of $p.max$ is the maximum number of spatial objects that cluster c may have, while the value of $p.ca$ is the minimum area that cluster c may occupy. Thus, a density upper bound can be obtained by dividing $p.ca$ by $p.max$. For each vertex $p \in G.V$, the values of $p.max$ and $p.ca$ are pre-computed (refer to Sections 2.1 and 3.1). By substituting Equation (19) into Equation (5), the lower bound of the density-evaluation score $E_d(c).lb$ is computed as follows

$$E_d(c) \geq \frac{2}{1 + e^{p.max/p.ca}} - 1 = E_d(c).lb. \tag{20}$$

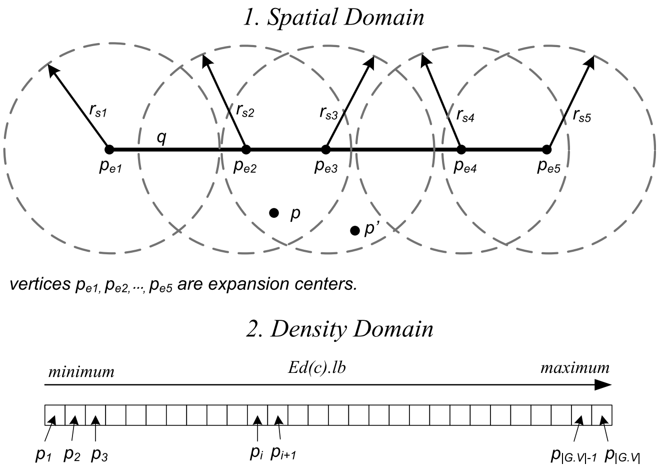


Fig. 4. An example of collective cluster search.

For each vertex $p \in G.V$, the value of $E_d(c).lb$ (where $c.m = p$) is pre-computed and inserted into a heap H. Thus the size of H is equal to that of $G.V$. The search process in the density domain is performed from the minimum to the maximum. Intuitively, a cluster with a smaller value of $E_d(c).lb$ has a higher possibility to be the cluster with the minimum density-evaluation score.

An example is shown in Fig. 4, where q is a query route, and vertices $p_{e1}, p_{e2}, \ldots, p_{e5} \in q$ are expansion centers. Network expansions occur from these according to Dijkstra's algorithm to explore the spatial network. Once a vertex is reached from any two adjacent expansion centers (e.g., vertex p in Fig. 4), we compute the upper bound of its distance-evaluation score $E_s(c, q).ub$ according to Equation (10). That is

$$E_s(c, q).ub = \frac{2}{1 + e^{-\min\{sd(p,p_i), sd(p,p_j)\}}} - 1,$$

where cluster c is centered at p, and p_i and p_j are two adjacent expansion centers. Such vertices are called "fully scanned" vertices. For partly scanned vertices (i.e., vertices reached from only one network expansion, e.g., vertex p' in Fig. 4), the lower bound of the distance-evaluation score $E_s(c', q).lb$ is estimated as (refer to Equation (13))

$$E'_s(c', q).lb = \frac{2}{1 + e^{-(sd(p_{e3},p') + rs - d(p_{e3},p_{e4}))/2}} - 1,$$

where cluster c' is centered at p', and rs is the radius of a circular scanned region. Since network expansions from different expansion centers occur at the same speed, we have that $rs = rs_1 = rs_2 = \cdots = rs_5$.

In the density domain, the items in heap H are sorted from minimum to maximum. Thus, we have that $E_d(c_1).lb < E_d(c_2).lb < \cdots < E_d(c_{|G.V|-1}).lb < E_d(c_{|G.V|}).lb$, where clusters $c_1, c_2, \ldots, c_{|G.V|}$ are centered at vertices $p_1, p_2, \ldots, p_{|G.V|}$, respectively (refer to Fig. 4). The search process is performed from the minimum to the maximum, step by step. In each step, the search pointer i is increased by 1. For example, in Fig. 4, clusters $c_1, c_2, \ldots, c_i$ have already been scanned (where $c_1, c_2, \ldots, c_i$ are centered at $p_1, p_2, \ldots, p_i$, respectively), and the rest are unscanned. For each scanned cluster, we compute the upper bound of its density-evaluation score $E_d(c).ub$ (refer to Equation (15)). For unscanned clusters (e.g., c_{i+1}, where $c_{i+1}.m = p_{i+1}$), the lower bound of density-evaluation score is estimated by

$$E'_d(c).lb = E_d(c_i).lb, \tag{21}$$

where cluster c is in $\{c_{i+1}, c_{i+2}, \ldots, c_{|G.V|}\}$ and i is the search-progress pointer.

By combining the lower bound of the distance-evaluation score (Equations (9) and (13)) and the lower bound of the density-evaluation score (Equations (20) and (21)), the lower bound of the distance-density evaluation score $E_{sd}(c,q).lb$ is defined as follows

$$E_{sd}(c,q).lb = \begin{cases} \lambda \cdot E_s(c,q).lb + (1-\lambda) \cdot E'_d(c).lb & \text{if } C_1 \\ \lambda \cdot E'_s(c,q).lb + (1-\lambda) \cdot E_d(c).lb & \text{if } C_2 \\ \lambda \cdot E'_s(c,q).lb + (1-\lambda) \cdot E'_d(c).lb & \text{if } C_3. \end{cases} \tag{22}$$

C_1: the center point of cluster c is fully scanned in the spatial domain and unscanned in the density domain.
C_2: the center point of cluster c is partly scanned in the spatial domain and scanned in the density domain.
C_3: the center point of cluster c is partly scanned in the spatial domain and unscanned in the density domain.

Among all clusters that have not been fully scanned in both the spatial and density domains, we define a global lower bound LB as

$$LB = \min\{E_{sd}(c_h, q).lb\}, \tag{23}$$

where LB is continuously updated during PNC query processing.

If a cluster c has been fully scanned in both domains, we compute the upper bound of its distance-and-density evaluation score $E_{sd}(c,q).ub$ (refer to Equation (16)). Among all clusters that have been fully scanned in both the spatial and density domains, a global upper bound UB is defined as

$$UB = \min\{E_{sd}(c_h, q).ub\}. \tag{24}$$

Similar to LB, UB is updated during query processing.

The search-stop criteria in both the spatial and density domains is whether the value of LB exceeds that of UB ($LB > UB$). The clusters outside the scanned regions in the both domains can be pruned safely. Then, the cluster candidates will be further refined according to the two-step refinement strategy (refer to Section 3.3), and the cluster with the minimum distance-and-density evaluation score is returned.

4.2 Expansion Center Selection

In this section, we introduce the expansion-center-selection strategy used in the collective cluster search algorithm. The strategy aims to minimize the search space in the spatial domain during query processing.

Assume that vertices along the query route q are uniformly distributed in the spatial domain. In the extreme case where each vertex in q is an expansion center, the search space for each individual expansion center is minimized while the number of expansion centers is maximized. When any two adjacent sample points (usually close to each other) are selected as expansion centers, the search space overlap is substantial. The vertices (clusters) in the overlapping region will be read and processed unnecessarily, which adversely affects performance. Next, in the extreme case where only the two ends of q (i.e., the source and destination, as in PNN query processing [6]) are selected as expansion centers, the number of expansion centers is minimized, but the search space for each expansion center may be very large.

The optimal selection of expansion centers can be estimated using linear programming. Let $\langle p_1 = s, p_2, .., p_{n-1}, p_n = d \rangle$ be the vertices in q ordered from source s to destination d. Let A be an $n \times n$ matrix where $a_{ij} = 1$ if the ith and the jth vertices are adjacent expansion centers (i.e., no vertices in-between them are expansion centers) and $a_{ij} = 0$, otherwise. Our goal is to minimize the area of the total search space in the spatial domain:

$$\omega = \sum a_{ij} \frac{\pi}{4} (d_{ij} + 2\epsilon)^2, \tag{25}$$

subject to $i < j$, $\sum_{j=1}^{n} a_{0j} = 1$, $\sum_{i=1}^{n} a_{i0} = 1$, $\sum_{j=1}^{n} a_{ij} \leq 1$, $\sum_{i=1}^{n} a_{ij} \leq 1$, and $\sum_{j=1}^{n} a_{ij} = \sum_{k=1}^{n} a_{ki}$. Here, d_{ij} is the network distance between the ith and the jth vertices along the query route q. We use $\sum a_{ij} \frac{1}{4} \pi (d_{ij} + 2\epsilon)^2$ to estimate the area of the total search space in the spatial domain, and we use $\sum a_{ij}$ to estimate the number of expansion centers. The area of the search space for each individual expansion center is estimated by $\frac{1}{4}\pi(d_{ij} + 2\epsilon)^2$, and $r = (d_{ij}/2 + \epsilon)$ is the radius of the search space. Here, ϵ is a parameter that estimates the maximum distance between a candidate cluster c and a query route q. For a cluster c with $d(c,q) \geq 8$, its distance-evaluation score $E_s(c,q) = \frac{2}{1+e^{-d(c,q)}} > 0.999 \approx 1$ (refer to Equation (4)). As the query finds the cluster with the minimum distance-evaluation score in the spatial domain, it is almost impossible for c to be the query result, and it is very likely that the search in the spatial domain terminates here (when search radius $r = d_{ij}/2 + \epsilon = d_{ij}/2 + 8$). Thus, the value of ϵ is set to 8 in the experimental studies.

Considering the online processing scenario, the time cost of finding the optimal selection of expansion centers by solving the above objective function may not be practical. Thus, we simplify the objective functions (Equations (25)) by assuming that the gap between any two adjacent expansion centers is identical and the vertices in q are uniformly distributed. Our goal is now changed to finding the optimal number of expansion centers. Then we have:

$$\omega(x) = \frac{x \cdot \pi}{4} \left(\frac{q.l}{x-1} + 2\epsilon \right)^2, \tag{26}$$

where $q.l$ is the length of query route q in the spatial domain and x is the number of expansion centers. The value of x resulting in the minimum ω is obtained using the derivatives of the function in Equation (26)

$$\begin{aligned} \omega(x)' &= \frac{\partial \omega}{\partial x} = 0 \\ &\Rightarrow 2q.l^2(x-1)^{-3} + (q.l^2 + 4\epsilon \cdot q.l)(x-1)^{-2} + 4\epsilon = 0. \end{aligned} \tag{27}$$

The cubic equation in Equation (27) can be solved by applying the general formula of roots. Then x uniformly distributed vertices (including the source and destination) are selected in q as the expansion centers. The necessity of the vertex-selection strategy in the spatial domain depends on the values computed for x. If the number of vertices in the query route is no greater than the value of x (i.e., $|q| \leq x$), it is not necessary to conduct the expansion-center-selection strategy in the spatial domain. By following the aforementioned procedure, the expansion centers that define the minimum search spaces in the spatial domain is found.

4.3 Analysis

Correctness. Similar to the SF algorithm, the collective algorithm follows the "filter-and-refinement" paradigm. We define a global upper bound UB and a global lower bound LB (refer to Equations (23) and (24)) of the distance-and-density evaluation score to prune the search space. When we have that $LB > UB$, the search process terminates. It is clear that clusters outside the scanned region cannot be a solution, and they can be pruned safely. Then we refine the candidates inside the scanned region by computing their exact distance-and-density evaluation scores, and we find the result. Because (1) the clusters pruned in the filtering cannot be a solution and because (2) the computation in the refinement is accurate, the collective algorithm computes the correct solution to the PNC query.

Notice that the expansion-center selection strategy (refer to Section 4.2) only accelerates the query processing and does not affect the query result.

Time complexity. Similar to the SF algorithm, in the spatial domain, we use network expansion to constrain a candidate set, and then we use Dijkstra's algorithm [10] to refine the candidates. The time complexity for the spatial domain is $O(|V|^2 log(|V|) + |V||E|)$. In the density domain, clusters are sorted according to their density, and they are scanned from maximum to minimum. The time complexity in the density domain is $O(|V|)$. The time complexity of the collective algorithm is then $O(|V|^2 log(|V|) + |V||E|) + O(|V|) = O(|V|^2 log(|V|) + |V||E|)$.

Space complexity. In the spatial domain, we maintain m expansion trees (m is the number of expansion centers, and it is a constant), and the space complexity of the spatial domain is $O(|V|)$. In the density domain, we establish a heap to maintain the lower bounds of the density-evaluation scores, and its space complexity is $O(|V|)$. Thus, the collective algorithm needs $O(|V|)$ additional space, excluding the $O(|V| + |E|)$ space occupied by the spatial network.

4.4 Algorithm

The collective cluster search algorithm is shown in Algorithm 2. Initially, the default value of global lower bound LB (refer to Equation (23)) is set to 0, and the default value of global upper bound UB is set to $+\infty$. The candidate set CS is set to $null$, and the search pointer i in the density domain is set to 1 (line 1). We carefully select a set of expansion centers according to the expansion-center selection strategy introduced in Section 4.2 (line 2). In the spatial domain, we explore the spatial network and find the cluster located close to the query route according to Dijkstra's algorithm. For each newly scanned vertex p, we update the lower bound of its distance-evaluation score $E_s(c, q).lb$ (Equation (13)) and all related parameters. If p has been "fully scanned" (i.e., scanned by network expansions from two adjacent expansion centers), we compute the upper bound of its distance-evaluation score $E_s(c, q).ub$ (Equation (10)) (lines 4-10). If vertex p has been scanned in both the spatial and density domains, we compute the upper and lower bounds of its distance-and-density evaluation score $E_{sd}(c, q).ub$ and $E_{sd}(c, q).lb$ (Equations (16) and (22)). Then, we update the values of global upper and lower bounds UB and LB (Equations (17) and (23)). If the value of LB exceeds that of UB, the search process in the two domains terminates (lines 11-15). If the number of spatial objects covered by circular region $(p, \tau.r)$ (this number is pre-computed, as explained in Section 3.1) is less than the cluster size threshold $\tau.s$, no cluster centered at p exists that meets the size and radius thresholds; thus vertex p is pruned. Otherwise, vertex p is put into the candidate set CS (lines 16-17).

Algorithm 2. Collective Cluster Search

```
   Data: graph $G(V, E)$, spatial-object set $O$, query route $q$
   Result: cluster $c$ with the minimum value of $E_{sd}(c, q)$
1  $LB \leftarrow 0; UB \leftarrow +\infty; CS \leftarrow null; minE_{sd} \leftarrow 1; i \leftarrow 1;$
2  select expansion centers in the query route;
3  while true do
4      //in the spatial domain
5      for each expansion center $p_e \in q$ do
6          $p \leftarrow expand(p_e);$
7          if $c.m = p$ then
8              if $p$ is fully scanned then
9                  compute $E_s(c, q).ub$;
10             update $E_s(c, q).lb$ and all related parameters;
11         if $p$ is scanned in the two domains then
12             compute $E_{sd}(c, q).lb$ and $E_{sd}(c, q).ub$;
13             update $LB$ and $UB$;
14             if $LB > UB$ then
15                 break;
16         if $(p, \tau.r).s \geq \tau.s$ then
17             $CS.add(p)$;
18     //in the density domain
19     if $c.m = p_i$ then
20         compute $E_d(c).ub$, $E_d(c).lb$, and all related parameters;
21         $i \leftarrow i + 1$;
22     if $p$ is scanned in the two domains then
23         compute $E_{sd}(c, q).lb$ and $E_{sd}(c, q).ub$;
24         update $LB$ and $UB$;
25         if $LB > UB$ then
26             break;
27     if $(p, \tau.r).s \geq \tau.s$ then
28         $CS.add(p)$;
29 for each vertex $p \in CS$ do
30     compute $E_{sd}(c_h, q)$;
31     if $E_{sd}(c_h, q) < minE_{sd}$ then
32         $minE_{sd} \leftarrow E_{sd}(c_h, q)$;
33 return the cluster $c$ with $minE_{sd}$;
```

In the density domain, the search is performed from the minimum to the maximum. In each step, the search pointer i is incremented by 1. For each newly scanned vertex p_i, we compute the upper and lower bounds of its density-evaluation score $E_d(c).ub$ and $E_d(c).lb$ (Equations (15) and (20)) and all related parameters (lines 18-21). If vertex p has been scanned in both the spatial and density domains, we compute the corresponding upper and lower bounds and check whether they meet the the search-stop criteria (lines 22-26). Also, we check whether the number of spatial objects covered by circular region $(p, \tau.r)$ meets the cluster size threshold $\tau.s$. If so, vertex p is put into CS (lines 27-28). For each vertex in the candidate set CS, we compute the exact value of $E_{sd}(c_h, q)$ $(c_h.m = p)$ according to procedure introduced in Section 3.3. By integrating these results, the cluster with the minimum value of $E_{sd}(c_h, q)$ is found (lines 29-33).

4.5 Top-k Extension

It is straightforward to extend the proposed techniques to support a top-k PNC query. Among all clusters fully scanned in the spatial and density domains, we define a global upper bound UB_k for the top-k PNC query as

$$UB_k = \max_{c \in C_k} \{E_{sd}(c, q).ub\}, \tag{28}$$

where C_f is the set of clusters that have been fully touched in both domains and $C_k = \{c_1, c_2, \ldots, c_k\}$ $(C_k \subseteq C_f)$, such that

$$\forall c \in C_k (\forall c' \in C_f \setminus C_k (E_{sd}(c, q).ub \leq E_{sd}(c', q).ub)).$$

The value of UB_k is dynamic and is updated continuously. The search process for the top-k PNC query is conducted by substituting Equation (28) into Algorithms 1 and 2. If LB exceeds UB_k, the search process terminates. Then, we refine the cluster candidates and find top-k clusters with the minimum spatial-density evaluation score.

5 Experimental Results

In this section, we conduct extensive experiments on real and synthetic spatial data sets to study the performance of the developed algorithms.

5.1 Settings

We use graphs extracted from two spatial networks, namely the Beijing Road Network (BRN) and the Oldenburg Road Network (ORN),[8] which contain 28,342 and 6,104 vertices, respectively. The graphs are stored using adjacency lists. For studies with BRN, we use a real geo-tagged object set collected from the micro-blogging service Weibo over within thirty days, which contains 600,000 spatial objects. Raw geo-tagged objects have latitude-longitude coordinates. They were mapped to the spatial network and attached to their nearest vertices. For each vertex p in BRN, we maintain the number of spatial objects that have p as their nearest vertex as an attribute of p. Thus, we do not need to access individual spatial objects during PNC query processing. For studies with ORN, synthetic data was used. For each vertex p' in ORN, we generate the number of spatial objects attached to it, and we maintain this number as one of its attributes. There are 150,000 generated spatial objects.

TABLE 2
Parameter Settings

	BRN	ORN
Spatial object count	600,000	150,000
Query route length	20-100/default 60	20-100/default 60
Cluster size threshold	200-1,000/default 600	100-500/default 300
Cluster radius threshold	1-3 km/default 2 km	1-3 km/default 2 km
λ	0.1-0.9/default 0.5	0.1-0.9/default 0.5
k	1-9/default 1	1-9/default 1

In the experiments, the graphs were memory resident when running Dijkstra's algorithm [10], as the memory occupied by BRN/ORN was less than 20 MB. All algorithms were implemented in Java and run on a Windows 7 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 query inputs. The main performance metrics are CPU time and the number of visited vertices. The number of visited vertices is selected as a metric since it describes the exact amount of data accesses.

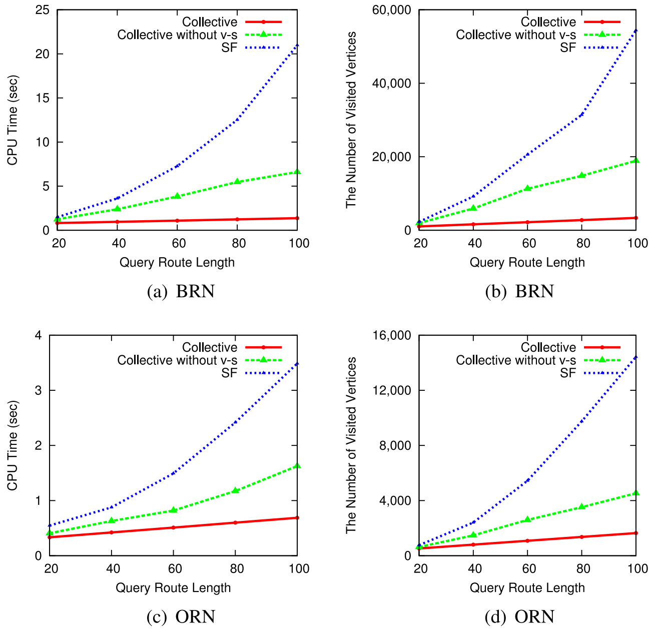
The parameter settings are listed in Table 2. By default, the lengths of query routes were set to 60 in both BRN and ORN, and each query route was randomly generated. The cluster size thresholds were set to 600 and 300 in BRN and ORN, respectively. The cluster radius threshold was set to 2 km, and λ was set to 0.5 for both BRN and ORN. The collective cluster search algorithm (Section 4) is denoted by "Collective", Spatial-first cluster search (Section 3) denoted by "SF", and collective cluster search algorithm without expansion-center selection strategy, denoted by "Collective without v-s".

5.2 Effect of Query Route Length $q.l$

First, we investigate the effect of query route length $q.l$ on the performance of the three algorithms with the default settings. Intuitively, a longer query route causes more expansion centers to be processed and has a larger search space. Thus, the CPU time and the number of visited vertices are expected to be higher for all three algorithms. However, from Fig. 5, it is clear that the CPU time and the number of visited vertices of SF search increase much faster than those of collective search. This is due to two reasons. First, SF search lacks effective upper and lower bounds to prune the search space. Second, the expansion-center selection strategy can further enhance the query efficiency. For instance, with the query route length $q.l = 100$, the collective algorithm outperforms SF search by almost a factor of 10 (for both CPU time and visited vertices); with the help of the expansion-center selection strategy (refer to Section 4.2), the performance is improved by approximately a factor of 3 in terms of both CPU time and visited vertices.

It is worth to note that (i) the number of visited vertices may be greater than the size of graph $|G.V|$ since a vertex may be visited several times by network expansions from

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



(a) BRN

(b) BRN

(c) ORN

(d) ORN

Fig. 5. Effect of query route length.

different expansion centers; (ii) the CPU time is not fully aligned with the number of visited vertices. To prune the search space, the collective cluster search algorithm needs more computational effort to maintain its bounds. In some cases, the increased computation cost may offset the benefits of the reduction in the number of visited vertices.

5.3 Effect of Cluster Size Threshold $\tau.s$

Fig. 6 presents the performance of the algorithms with varying cluster size threshold $\tau.s$. Since the total number of spatial objects is fixed, a higher cluster size threshold means fewer "qualified" clusters. The sparser the cluster (center point) distribution, the larger the required search space, and thus the performance of query processing may decrease. In Fig. 6, the CPU time and the number of visited vertices for all three algorithms increase as the cluster size threshold increases. From Fig. 6, it is clear that the CPU time and the number of visited vertices required by SF search are 5-10 times higher than those needed by the collective search algorithm. In addition, the expansion-center selection strategy can further enhance the efficiency by a factor of 2-4 in terms of both CPU time and the number of visited vertices.

5.4 Effect of Cluster Radius Threshold $\tau.r$

Next, we vary the cluster radius thresholds $\tau.r$. With a fixed value of cluster size threshold, a larger cluster radius threshold leads to more "qualified" clusters. Intuitively, the denser the cluster (center point) distribution, the smaller the required search space, and thus the queries are expected to be faster. In Fig. 7, the CPU time and the number of visited vertices for all three algorithms decrease as the cluster radius threshold increases. The collective search outperforms spatial-first search and collective search without the expansion-center selection strategy by factors of 5-10 and 2-4, respectively, in terms of both CPU time and the number of visited vertices.

5.5 Effect of λ

Parameter λ is used to adjust the relative importance of the distance and density. In the extreme case where $\lambda = 1$, the PNC query is conducted in the spatial domain only. And when $\lambda = 0$, density is the sole factor considered. Fig. 8 shows the performance of the three algorithms for different values of λ. For both collective search algorithms, it is clear that the search effort (CPU time and visited vertices) required in the spatial domain is higher than that required in the density domain. For the spatial-first algorithm, the increasing importance of the spatial domain improves the pruning effectiveness of its bounds; thus, the CPU time and number of vertices decrease with the increasing value of λ. In fact, the spatial-first algorithm uses the bounds in the spatial domain to prune the search space in both the spatial and density domains. With the increasing value of λ, the PNC query is becoming an increasingly spatial query, and the pruning effectiveness of its bounds is improved. When $\lambda = 1$, spatial-first search is equal to the collective search without the expansion-center selection strategy.

5.6 Effect of k

Fig. 9 shows the effect of varying parameter k on the performance of the three algorithms with the default settings. Intuitively, a larger value of k leads to a larger search space, and the CPU time and the number of visited vertices are expected to be higher for all three algorithms. The collective algorithm has a clear advantage over other two algorithms. It outperforms the SF algorithm and the "collective without v-s" algorithm by factors of 5-10 and 2-4, respectively, in terms of both CPU time and the number of visited vertices.

5.7 Scalability

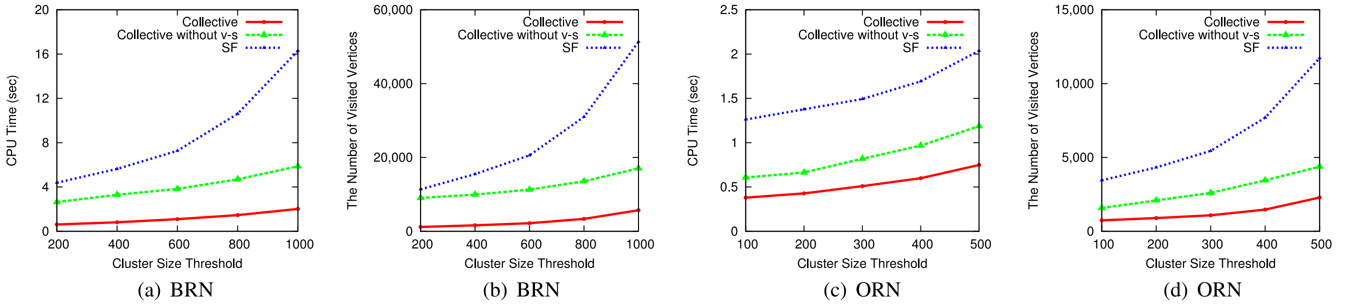
To study the scalability of the developed algorithms, we conduct experiments on the North America Road Network (NRN),[9] which contains 175,813 vertices and 179,179 edges. For each vertex p in NRN, we generate the number of spatial objects attached to it, and we maintain this number as one of its attributes. There are a total of 1,000,000 generated spatial objects. Fig. 10 shows the effect of query route length on the performance of the three algorithms with the default settings. It is clear that collective algorithm is capable of computing the PNC query on large spatial data sets in interactive time.

6 Related Work

Nearest Neighbor queries aim to find objects near a query location and constitute fundamental functionality in spatial data management. NN query processing may occur in different settings, including in euclidean spaces (e.g., [7], [16], [26], [27], [28], [36]), in spatial networks (e.g., [19], [20], [21], [26], [34]), in indoor spaces (e.g., [41]), and in higher dimensional spaces (e.g., [9], [18]).

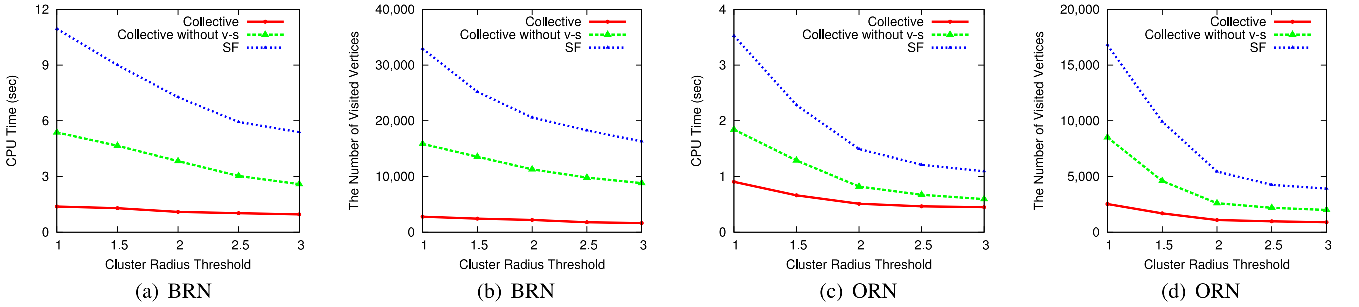
Existing trajectory queries (route queries) come in two general forms. In the trajectory-to-point category, queries aim to find spatial objects closest to a query route (trajectory) according to some distance metric. For example, the in-route nearest neighbor (IRNN) query (e.g., [35], [42]) is

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



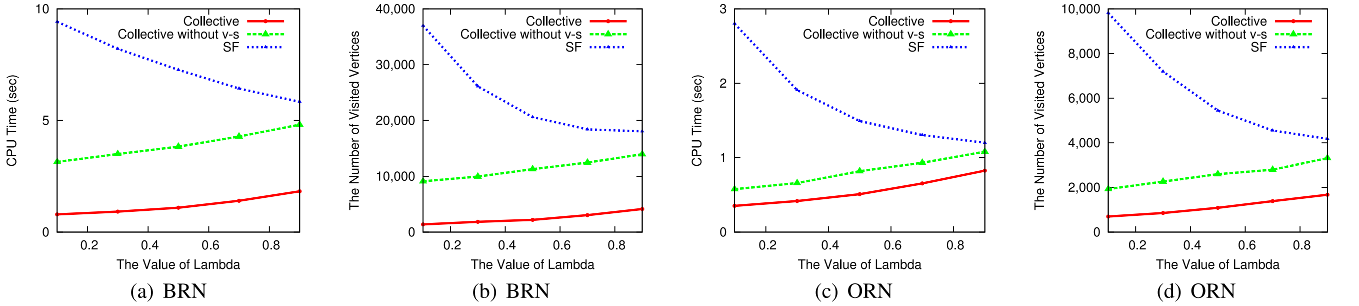
(a) BRN (b) BRN (c) ORN (d) ORN

Fig. 6. Effect of cluster size threshold.



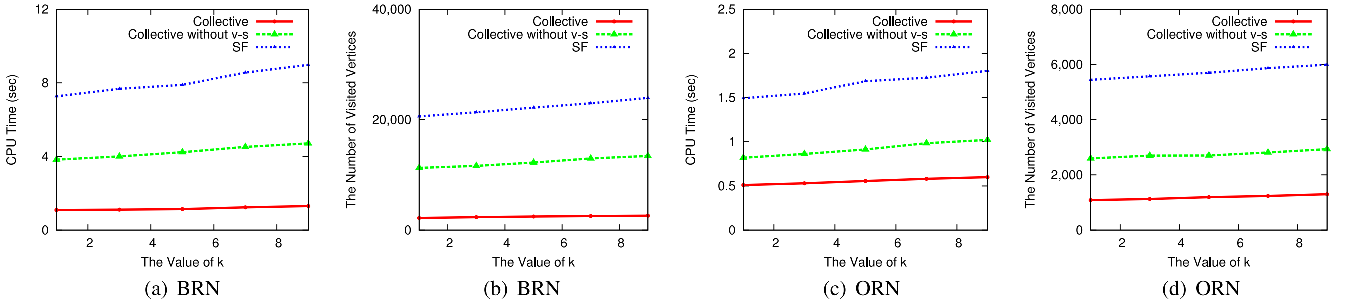
(a) BRN (b) BRN (c) ORN (d) ORN

Fig. 7. Effect of cluster radius threshold.



(a) BRN (b) BRN (c) ORN (d) ORN

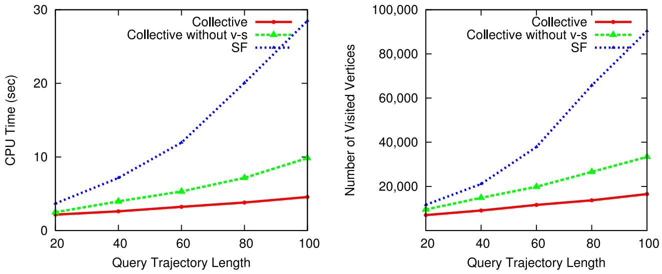
Fig. 8. Effect of λ.



(a) BRN (b) BRN (c) ORN (d) ORN

Fig. 9. Effect of k.

designed for travelers following a fixed route. The IRNN query finds the facility (e.g., a gas station) with the minimum detour distance from the fixed route, the assumption being that a traveler will return to the original route after visiting the nearest facility. The path nearest neighbor query (e.g., [6], [29], [32], [33]) is an extension of the IRNN query that maintains an up-to-date path nearest neighbor result as the user is moving along a predefined route. In the trajectory-to-trajectory category, queries aim to find trajectories with the highest similarity to a query trajectory (e.g., [30], [31], [45], [46]). A trajectory similarity function may contain spatial, temporal, and textual elements. For example, the UOTS query [30] and the OATSQ query [45] have spatial and textual elements, and the PTM query [31] has spatial and temporal elements. Both curve similarity and location proximity are taken into account in the spatial and temporal domains, and textual similarity is considered in the textual domain. There also exist other trajectory and path planning



(a) NRN

(b) NRN

Fig. 10. Scalability.

studies, including safest path planning [1], trajectory-based destination prediction [38], [39], trajectory-based travel time estimation and fastest path planning [43], and route recommendation under uncertainty [17].

Unlike the existing studies, the path nearby cluster query belongs to a trajectory-to-cluster category. It aims to find clusters with high spatial-object density and located near a query route. Existing solutions fail to solve the PNC query due to four reasons. First, PNN considers only spatial distance, while PNC balances the importance of spatial-object density and spatial distance. Second, PNN only supports network shortest paths as a query argument, while PNC can tackle arbitrary types of query routes. Third, in PNC, a linear combination method (e.g., [5], [8]) is used to combine the spatial and density elements, and a distance-and-density evaluation score is defined correspondingly; thus, PNC is also different from skyline queries (e.g., [3], [24], [40]), and existing skyline techniques cannot be used here. Fourth, PNC finds clusters with the highest density, while existing online clustering methods (e.g., DBScan [11]) formulate clusters based on an estimated density distribution (i.e., they find clusters whose densities are higher than the estimated density); thus, they are not suitable for PNC query processing.

7 Conclusions and Future Directions

We propose and investigate a novel problem, the path nearby cluster query, of finding path nearby clusters in spatial networks. This query is designed to discover regions of potential interest, and we believe that it is useful in scenarios such as trip planning and location recommendation. To compute the query efficiently, a collective cluster search algorithm was proposed. A pair of upper and lower bounds were developed to prune the search space effectively. Finally, the performance of PNC query processing was investigated by means of extensive experiments on real and synthetic spatial data.

Two interesting directions for future research exist. First, it is of interest to study a continuous counterpart of the PNC query (continuous-PNC). Assume that a traveler is moving along a specified route and that the target is to monitor the path nearby clusters with the traveler's movement. The new challenge lies in finding a set of update locations along the query route. Second, it is of interest to consider an additional textual element for the PNC query, where spatial objects are associated with textual attributes. The PNC query then aims to find clusters of specified web objects, such as clusters of sightseeing places, close to a given travel route. Existing collective search algorithm can be extended to support the spatial, textual, and density domains, and the difficulty lies in how to schedule the search processes in these domains effectively.

Acknowledgments

This work is partly supported by the National Natural Science Foundation of China (NSFC. 61402532), the Science Foundation of China University of Petroleum-Beijing (No. 2462013YJRC031), the Excellent Talents of Beijing Program (No. 2013D009051000003), and by a grant from the Obel Family Foundation. Guohe Li is the corresponding author.

References

[1] S. Aljubayrin, J. Qi, C. S. Jensen, R. Zhang, Z. He, and Z. Wen, "The safest path via safe zones," in *Proc. 31th IEEE Int. Conf. Data Eng.*, 2015, pp. 1–12.

[2] H. Alt, A. Efrat, G. Rote, and C. Wenk, "Matching planar maps," in *Proc. 14th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2003, pp. 589–598.

[3] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, 2001, pp. 421–430.

[4] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *Proc. 31st Int. Conf. Very Large Data Bases*, 2005, pp. 853–864.

[5] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi, "Collective spatial keyword querying," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2011, pp. 373–384.

[6] Z. Chen, H. T. Shen, X. Zhou, and J. X. Yu, "Monitoring path nearest neighbor in road networks," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 591–602.

[7] H.-J. Cho and C.-W. Chung, "An efficient and scalable approach to CNN queries in a road network," in *Proc. 31st Int. Conf. Very Large Data Bases*, 2005, pp. 865–876.

[8] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial web objects," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 337–348, 2009.

[9] K. Deng, X. Zhou, H. T. Shen, K. Xu, and X. Lin, "Surface k-NN query processing," in *Proc. 22nd Int. Conf. Dta Eng.*, 2006, p. 78.

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

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231.

[12] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. Sondag, "Adaptive fastest path computation on a road network: A traffic mining approach," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 794–805.

[13] J. Greenfeld, "Matching GPS observations to locations on a digital map," in *Proc. 81th Annu. Meeting Transportation Res. Board*, 2002, pp. 1–13.

[14] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul, "Ecomark: Evaluating models of vehicular environmental impact," in *Proc. 20th Int. Conf. Adv. Geographic Inform. Syst.*, 2012, pp. 269–278.

[15] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1984, pp. 47–57.

[16] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Trans. Database Syst.*, vol. 24, no. 2, pp. 265–318, 1999.

[17] E. Horvitz and J. Krumm, "Some help on the way: opportunistic routing under uncertainty," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 371–380.

[18] H. Jagadish, B. Ooi, K.-L. Tan, C. Yu, and R. Zhang, "idistance: An adaptive b+-tree based indexing method for nearest neighbour search," *ACM Trans. Database Syst.*, vol. 30, no. 2, pp. 364–397, 2005.

[19] C. S. Jensen, J. Kolarvr, T. B. Pedersen, and I. Timko, "Nearest neighbor queries in road networks," in *Proc. 11th ACM Int. Symp. Adv. Geographic Inf. Syst.*, 2003, pp. 1–8.

[20] M. Kolahdouzan and C. Shahabi, "Voronoi-based k nearest neighbor search for spatial network databases," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2004, pp. 840–851.

[21] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, "On trip planning queries in spatial databases," in *Proc. 9th Int. Conf. Adv. Spatial Temporal Databases*, 2005, pp. 273–290.

[22] K. Liu, Y. Li, F. He, J. Xu, and Z. Ding, "Effective map-matching on the most simplified road network," in *Proc. 20th Int. Conf. Adv. Geographic Inform. Syst.*, 2012, pp. 609–612.

[23] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate GPS trajectories," in *Proc. 17th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inform. Syst.*, 2009, pp. 352–361.

[24] H. Lu, C. S. Jensen, and Z. Zhang, "Flexible and efficient resolution of skyline query size constraints," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 7, pp. 991–1005, Jul. 2011.

[25] T. M. Mitchell, "Artificial neural networks," in *Machine Learning*. New York, NY, USA: WCB/McGraw-Hill, 1997.

[26] K. Mouratidis, D. Papadias, and M. Hadjieleftheriou, "Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 634–645.

[27] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1995, pp. 71–79.

[28] C. Shahabi, M. R. Kolahdouzan, and M. Sharifzadeh, "A road network embedding technique for k-nearest neighbor search in moving object databases," in *Proc. 10th ACM Int. Symp. Adv. Geographic Inform. Syst.*, 2002, pp. 94–100.

[29] S. Shang, K. Deng, and K. Xie, "Best point detour query in road networks," in *Proc. 18th SIGSPATIAL Int. Conf. Adv. Geographic Inform. Syst.*, 2010, pp. 71–80.

[30] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis, "User oriented trajectory search for trip recommendation," in *Proc. 15th Int. Conf. Extending Database Technol.*, 2012, pp. 156–167.

[31] S. Shang, R. Ding, K. Zheng, C. S. Jensen, P. Kalnis, and X. Zhou, "Personalized trajectory matching in spatial networks," *VLDB J.*, vol. 23, no. 3, pp. 449–468, 2014.

[32] S. Shang, B. Yuan, K. Deng, K. Xie, K. Zheng, and X. Zhou, "PNN query processing on compressed trajectories," *GeoInformatica*, vol. 16, no. 3, pp. 467–496, 2012.

[33] S. Shang, B. Yuan, K. Deng, K. Xie, and X. Zhou, "Finding the most accessible locations: Reverse path nearest neighbor query in road networks," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inform. Syst.*, 2011, pp. 181–190.

[34] M. Sharifzadeh, M. Kolahdouzan, and C. Shahabi, "The optimal sequenced route query," *VLDB J.*, vol. 17, pp. 765–787, 2008.

[35] S. Shekhar and J. S. Yoo, "Processing in-route nearest neighbor queries: A comparison of alternative approaches," in *Proc. 11th ACM Int. Symp. Adv. Geographic Inform. Syst.*, 2003, pp. 9–16.

[36] Y. Tao, D. Papadias, and Q. Shen, "Continuous nearest neighbor search," in *Proc. 28th Int. Conf. Very Large Data Bases*, 2002, pp. 287–298.

[37] C. Wenk, R. Salas, and D. Pfoser, "Addressing the need for map-matching speed: Localizing global curve-matching algorithms," in *Proc. 18th Int. Conf. Sci. Statistical Database Manage.*, 2006, pp. 379–388.

[38] A. Y. Xue, J. Qi, X. Xie, R. Zhang, J. Huang, and Y. Li, "Solving the data sparsity problem in destination prediction," *VLDB J.*, pp. 1–25, online first, 2014.

[39] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction," in *Proc. IEEE Int. Conf. Data Eng.*, 2013, pp. 254–265.

[40] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 136–147.

[41] B. Yang, H. Lu, and C. S. Jensen, "Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space," in *Proc. 13th Int. Conf. Extending Database Technol.*, 2010, pp. 335–346.

[42] J. S. Yoo and S. Shekhar, "In-route nearest neighbor queries," *GeoInformatica*, vol. 9, pp. 117–137, 2005.

[43] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 316–324.

[44] P. Zarchan, *Global Positioning System Theory and Applications* in Progress in Astronautics and Aeronautics (Series), vol. 163. Reston, VA, USA: AIAA, 1996, pp. 1–781.

[45] K. Zheng, S. Shang, N. J. Yuan, and Y. Yang, "Towards efficient search for activity trajectories," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 230–241.

[46] Y. Zheng and X. Zhou, Eds., *Computing with Spatial Trajectories*. New York, NY, USA: Springer, 2011.

Shuo Shang received the PhD degree in computer science from the University of Queensland. He is a professor of computer science at the China University of Petroleum-Beijing. He was a research assistant professor in the Department of Computer Science, Aalborg University. His research interests include efficient query processing in spatiotemporal databases, trajectory search and mining, and location-based services. He has served as a session chair and an invited reviewer for many prestigious conferences and journals, including ICDE, *TKDE*, *The VLDB Journal*, *ACM TIST*, *GeoInformatica*, *KAIS*, *DKE*, and *IEICE Transactions*.

Kai Zheng received the PhD degree from The University of Queensland. He is an ARC DECRA fellow with The University of Queensland. His research interests include uncertain database, spatial-temporal query processing, and trajectory computing.

Christian S. Jensen is an 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. He has received several national and international awards for his research. He is an editor-in-chief of *The VLDB Journal*. He is a fellow of the ACM and IEEE, and he is a member of the Academia Europaea, the Royal Danish Academy of Sciences and Letters, and the Danish Academy of Technical Sciences.

Bin Yang received the PhD degree from Fudan University, China. He is an assistant professor at Aalborg University, Denmark. He was recently at Aarhus University, Denmark, for three years and at Max-Planck-Institut für Informatik, Germany, for one year. His research interests include data management and data analytics.

Panos Kalnis received the diploma in computer engineering from the Computer Engineering and Informatics Department, University of Patras, and the PhD degree from HKUST. He is an associate professor at KAUST. His research interests include Database outsourcing and Cloud Computing, Mobile Computing, and Spatiotemporal and High-dimensional Databases. He serves on associate editors of *TKDE*, and *The VLDB Journal*.

Guohe Li is a professor of computer science at the China University of Petroleum-Beijing. His research interests include database, data mining, and artificial intelligence.

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+ US patents in web search and related areas. He has published extensively on prestigious international conferences and journals. He is currently the associate editor of the *ACM Transactions on Information Systems (TOIS)*. He is a senior member of the IEEE.