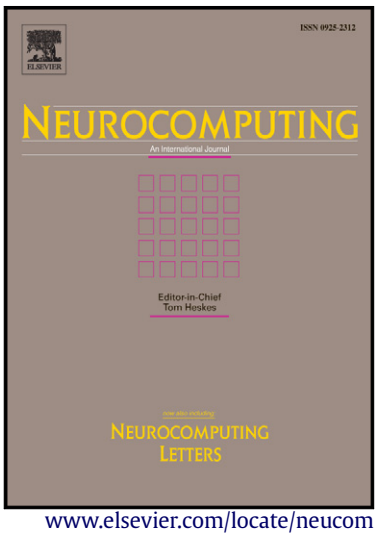


www.elsevier.com/locate/neucom

Finding Regions of Interest using Location Based Social Media

Shuo Shang, Danhuai Guo, Jiajun Liu, Kai Zheng, Ji-Rong Wen

PII: S0925-2312(15)01128-5
DOI: http://dx.doi.org/10.1016/j.neucom.2015.06.086
Reference: NEUCOM15904

To appear in: *Neurocomputing*

Received date: 24 March 2015
Revised date: 18 May 2015
Accepted date: 3 June 2015

Cite this article as: Shuo Shang, Danhuai Guo, Jiajun Liu, Kai Zheng, Ji-Rong Wen, Finding Regions of Interest using Location Based Social Media, *Neurocomputing*, http://dx.doi.org/10.1016/j.neucom.2015.06.086

Finding Regions of Interest using Location Based Social Media

Shuo Shang[a], Danhuai Guo[b], Jiajun Liu[c], Kai Zheng[d], and Ji-Rong Wen[e]

[a] *China University of Petroleum, Beijing, China*

[b] *CNIC, Chinese Academy of Sciences, Beijing, China*

[c] *CSIRO, Brisbane, Australia*

[d] *The University of Queensland, Brisbane, Australia*

[e] *Renmin University of China, Beijing, China*

Email:jedi.shang@gmail.com

Abstract

The discovery of regions of interest in city groups is increasingly important in recent years. In this light, we propose and investigate a novel problem called Region Discovery query (RD query) that finds regions of interest with respect to a user's current geographic location. Given a set of spatial objects O and a query location q, if a circular region ω is with high spatial-object density and is spatially close to q, it is returned by the query and is recommended to users. This type of query can bring significant benefit to users in many useful applications such as trip planning and region recommendation. The RD query faces a big challenge: how to prune the search space in the spatial and density domains. To overcome the challenge and process the RD query efficiently, we propose a novel collaboration search method and we define a pair of bounds to prune the search space effectively. The performance of the RD query is studied by extensive experiments on real and synthetic spatial data.

Keywords: Region of Interest, Distance, Density, Location based Social Media

1. Introduction

With the rapid development of GPS-equipped mobile device (e.g., smart phones, car navigation systems, and PDAs) and online map services (e.g., Google Maps[1] and Bing Maps[2]), people can easily acquire their current geographic location in real time and can retrieve spatial information relevant to their trips[6]. In this light, we propose and study a novel query called Region

[1] http://maps.google.com/

[2] http://www.bing.com/maps/

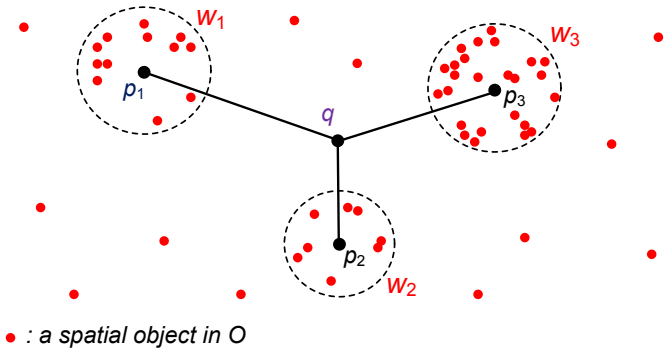


Figure 1: An example of region discovery query

Discovery query (RD query) that identifies regions of interest with respect to a user's current geographic location. Spatial objects can be geo-tagged tweets and micro-blog posts from location-based social media, such as Twitter[3], Weibo[4], and Foursquare[5][3, 27].

A region ω is a circular area defined by a center point $\omega.c$ and a radius $\omega.r$. We define two thresholds to identify qualified regions: (1) a size threshold $\tau.s$, i.e., a region should contains at least a threshold number of spatial objects; (2) a radius threshold $\tau.r$, i.e., the radius of the region should not exceed a radius threshold. Given a query location q, if a region ω is with dense spatial-object distribution and is spatially close to q, the region ω is returned by the query and is recommended to users. The region discovery query (RD query) is useful in many popular mobile applications such as trip planning and location recommendation. For example, when traveling overseas, travelers may wish to know about regions of interest (e.g., commercial districts and dining areas) around him/her. Intuitively, regions with high spatial-object density (e.g., geo-tagged tweets, micro-blog posts, and points of interest) are assumed to be more attractive to users. Also, a region located close to a user's current location is more attractive than a far-away region.

To the best of our knowledge, this is the first work that study the region recommendation problem while taking both spatial distance and spatial-object density into account. Previous studies (e.g., nearest neighbor query [10, 9, 11, 23]) only use spatial distance as the sole factor when computing the query results. In contrast, the RD query takes both spatial distance and density distribution into account. A linear combination method [18, 17] is adopted to combine the spatial and density domains.

[3]https://twitter.com/
[4]http://weibo.com/
[5]https://foursquare.com/

An example of the RD query is shown in Figure 1. Here, q is a query point, and vertices p_1, p_2, and p_3 are the center points of regions ω_1, ω_2, and ω_3, respectively. The distance between a region and a query point is defined by the shortest network distance between the region center point and the query point (e.g., $dist(\omega_2, q) = dist(p_2, q)$). If considering the spatial distance only (e.g., the same as the nearest neighbor query [9]), ω_2 is the region closest to q. However, when considering both spatial distance and the density of spatial objects, ω_2 is less attractive than ω_3 because of its sparser spatial-object distribution. Although ω_3 is not as good as ω_2 according to spatial distance, we still consider ω_3 as the best choice for region recommendation when taking both spatial distance and spatial-object density into account.

The RD query is applied in spatial networks, since in a large number of practical scenarios, travelers move in spatial networks (e.g., road networks) rather than in a Euclidean space[25, 26, 19, 21, 20]. To enable efficient processing of the RD query, for each vertex p, 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 point and $\tau.r$ is the radius. These counts are useful in pruning the search space during query processing. Based on the pre-computation results, we develop an adaptive collaboration algorithm to compute the RD query efficiently. The search process is conducted in the spatial and density domains concurrently. A pair of upper and lower bounds are defined to prune the search space.

To the best of our knowledge, there is no existing approach that can compute the RD query efficiently. To sum up, the main contributions of this work are as follows:

- We define a novel region discovery (RD) query, and it is useful in many mobile applications such as trip planning and location recommendation.
- We propose a set of new metrics to evaluate the distance-and-density score of regions.
- We develop an adaptive collaboration algorithm to compute the RD query efficiently.
- We conduct extensive experiments on real and synthetic data to study the performance of RD query.

The rest of the paper is organized as follows. Related work is covered in Section 2. Section 3 introduces spatial networks and the distance metrics used in the paper; and it also gives problem definitions. The collaboration search method is covered in Section 4, which is followed by the experimental results in Section 5. Conclusions are drawn in Section 6.

2. Related Work

Spatial queries in advanced traveler information system continue to proliferate in recent years. Nearest Neighbor(NN) query is considered as an important issue in such kind of applications. This kind of query aims to retrieve the closest neighbor to a query point from a set of given objects. Based

on different constraint conditions, NN query processing can be classified into three categories, such that in Euclidean spaces (e.g. [14, 7]), in spatial networks (e.g. [10, 9, 11, 23, 22]), and in higher dimensional spaces (e.g. [8, 2]).

As a variant of NN queries, Continuous Nearest Neighbor queries (CNN) [1, 13, 24, 15, 16] report the kNN results continuously while the user is moving along a path. This type of queries aims to find the split points on the query path where an update of the kNN is required, and thus to avoid unnecessary re-computation. In [13], Mouratidis et al. investigate the CNN monitoring problem in a road network, in which the query point moves freely and the data objects' positions are also changing dynamically. The basic idea of [13] is to maintain a spanning tree originated from the query point and to grow or discard branches of the spanning tree according to the data objects and query point's movements.

To the best of our knowledge, there is no existing approach that can compute the RD query efficiently. Previous studies only use spatial distance as the sole factor when computing the query results, and the density of spatial objects is not taken into account. In contrast, the RD query takes both spatial distance and density distribution into account. A linear combination method [18, 17] is adopted to combine the spatial and density domains.

3. Preliminaries

3.1. Network Modeling and Preprocessing

In this work, spatial networks are modeled by connected and undirected planar graphs $G(V, E)$, where V is the set of vertices and E is the set of edges. A weight can be assigned to each edge to represent its length or application specific factors such as traveling time obtained from historical traffic data [5]. Given two points a and b in road networks, the network distance between them is the length of their shortest network path (i.e., a sequence of edges linking a and b where the accumulated weight is minimal). The data points are distributed along roads and if a data point is not located at a road intersection, we treat the data point as a vertex and further divide the edge that it lies on into two edges. Thus, we assume that all data points are in vertices for the sake of clarity. We assume that each spatial object (e.g., geo-tagged tweets, geo-tagged photos) 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 RD query processing.

3.2. Problem Definition

A region ω is a circular area defined by a center point $\omega.c$ and a radius $\omega.r$. We define two thresholds to identify qualified regions: (1) a size threshold $\tau.s$, i.e., a region should contains at least a threshold number of spatial objects; (2) a radius threshold $\tau.r$, i.e., the radius of the region should not exceed a

radius threshold. A region ω has an associated subgraph $\omega.G$, which contains the vertices $\omega.V$ and edges $\omega.E$ from G that are in the circular region.

The density $\omega.\rho$ of region ω is defined as

$$\omega.\rho = \frac{\sum_{p \in \omega.V} p.g}{\sum_{e \in \omega.E} W(e)}, \tag{1}$$

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

Given a region ω and a query point q, a distance function $E_s(\omega, q)$ and a density function $E_d(\omega)$ are defined in Equations 2 and 3, respectively. We use Sigmoid function [12] to normalize the values of $E_s(\omega, q)$ and $E_d(\omega)$ to $[0, 1]$.

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

$$E_d(\omega) = \begin{cases} \frac{2}{1+e^{\omega.\rho}} - 1 & \text{if } \omega.r \leq \tau.r \wedge \omega.s \geq \tau.s \\ 1 & \text{if } \omega.r > \tau.r \vee \omega.s < \tau.s \end{cases} \tag{3}$$

Here, $\omega.c$ is the center point of region ω, and $sd(\omega.c, q)$ is the shortest path distance between $\omega.c$ and q, and $\tau.r$ and $\tau.s$ are region radius and size thresholds, respectively.

By combining Equations 2 and 3, the distance-and-density function $E_{sd}(c, \tau)$ is defined by

$$E_{sd}(\omega, q) = \alpha \cdot E_s(\omega, q) + (1 - \alpha) \cdot E_d(\omega), \tag{4}$$

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

Problem Definition
Given a set of spatial objects O, a query point q, a region size threshold $\tau.s$ and a region radius threshold $\tau.r$, and an importance parameter α, the region discovery (RD) query finds the region ω with the minimum value of $E_{sd}(\omega, q)$ and $\omega.s \geq \tau.s$ and $\omega.r \leq \tau.r$, such that $\forall \omega'(\omega' \neq \omega \wedge \omega'.s \geq \tau.s \wedge \omega'.r \leq \tau.r \Rightarrow E_{sd}(\omega, q) \leq E_{sd}(\omega', q))$.

4. Query Processing

To enhance the performance of RD query processing, a density based precomputation technique is applied at first. For each vertex $p \in G.V$, we precompute 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.

The collaboration search algorithm is conducted in the spatial and density domains concurrently. In the spatial domain, we use network expansion to explore the spatial network. In the density domain, regions are sorted according to their density, and they are scanned from the highest density to the lowest density. We define a pair of upper and lower bounds to prune the search space in Section 4.1. The collaboration search algorithm is detailed in Section 4.2.

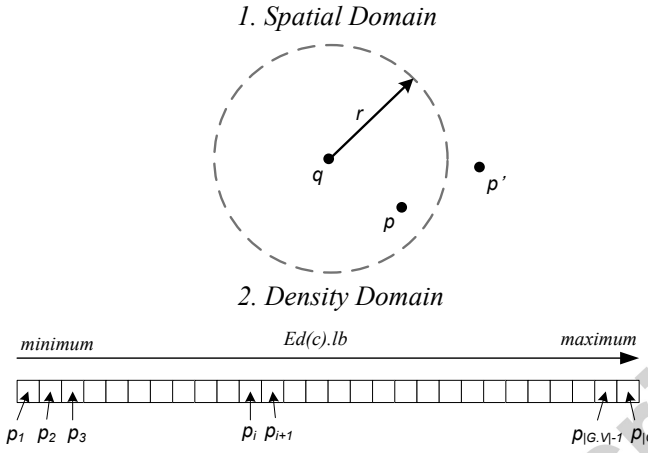


Figure 2: An example of the collaboration search

4.1. Basic Idea

Consider the example in Figure 2. In the spatial domain, point q is a query point. To explore the spatial network and find regions close to the query point (i.e., with the smaller values of $sd(p,q)$, and p is the center point of ω), Dijkstra's expansion [4] is adopted. From q, network expansion is performed using Dijkstra's algorithm. The explored region is a circular area (as shown in Figure 2), where the radius is the shortest network distance from q to the expansion boundary, denoted as r.

For the vertices inside the scanned region, such as p in Figure 2, we have its exact network distance to q (i.e., $sd(p,q)$). On the other hand, for the vertices outside the scanned region, such as p' in Figure 2, we estimate the lower bound of $sd(p',q)$ by

$$sd(p',q).lb = r, \tag{5}$$

where r is the radius of the scanned region. As Dijkstra's algorithm always selects the vertex with the minimum distance label for network expansion; thus for any p' outside the scanned region, we have $sd(p',q) > r = sd(p',q).lb$.

By substituting Equation 5 into Equation 2, the lower bound of $E_s(\omega,q)$ is estimated by

$$\begin{cases} E_s(\omega,q) = \frac{2}{1+e^{-sd(\omega.c,q)}} - 1 \\ r < sd(\omega.c,q) \end{cases}$$

$$\frac{2}{1+e^{-sd(\omega.c,q)}} - 1 > \frac{2}{1+e^{-r}} - 1 = E_s(\omega,q).lb, \tag{6}$$

where $\omega.c$ is outside the scanned region (e.g., p' in Figure 2).

In the density domain, we establish a heap H to maintain the lower bounds of the density score $E_d(\omega).lb$. The items in H are sorted from the minimum

to the maximum (i.e., from the highest density to the lowest density). Given a region ω, the upper bound of its density is estimated by

$$\omega.\rho.ub = \frac{p.m}{p.a} \geq \omega.\rho, \tag{7}$$

where vertex p is the center point of region ω ($\omega.c = p$), $p.m$ is the number of spatial objects covered by a circular region $(p, \tau.r)$, and $p.a$ is the control area of vertex p (i.e., the area occupied by the spatial objects that are attached to p). The value of $p.m$ is the maximum number of spatial objects that region ω may have, while the value of $p.a$ is the minimum area that region ω may occupy. Thus, a density upper bound can be obtained by dividing $p.a$ by $p.m$. For each vertex $p \in G.V$, the values of $p.m$ and $p.a$ are pre-computed. By substituting Equation 7 into Equation 3, the lower bound of the density score $E_d(\omega).lb$ is computed by

$$E_d(\omega) \geq \frac{2}{1 + e^{p.m/p.a}} - 1 = E_d(\omega).lb. \tag{8}$$

For each vertex $p \in G.V$, the value of $E_d(\omega).lb$ (where $\omega.c = p$) 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 (from the highest density to the lowest density). Intuitively, a region with a smaller value of $E_d(\omega).lb$ has a higher possibility to be the region with the minimum density score.

Once a region is fully scanned in both the spatial and density domains (e.g., ω is scanned in the density domain, and $\omega.c$ is scanned in the spatial domain), we compute its exact spatial-density score according to Equation 4. Among all scanned regions, we define a global upper bound UB as

$$UB = \min_{\omega \in R_s} \{E_{sd}(\omega, q)\}, \tag{9}$$

where R_s is a set of fully scanned regions.

If a region ω is unscanned in the density domain, the lower bound of its density score $E'_d(\omega).lb$ is defined by

$$E'_d(\omega).lb = E_d(\omega_i).lb, \tag{10}$$

where i is an iterator. The regions are sorted by their density lower bounds, and they are scanned from the minimum to the maximum. There are total i regions are scanned. Thus, for an unscanned region ω, we can use the value of $E_d(\omega_i).lb$ to estimate its density lower bound.

By combining Equations 7, 8 and 10, we define a lower bound for spatial-density score $E_{sd}(\omega, q)$.

$$E_{sd}(c, q).lb = \begin{cases} \alpha \cdot E_s(\omega, q) + (1 - \alpha) \cdot E'_d(\omega).lb & \text{if } C_1 \\ \alpha \cdot E_s(\omega, q).lb + (1 - \alpha) \cdot E_d(\omega).lb & \text{if } C_2 \\ \alpha \cdot E_s(\omega, q).lb + (1 - \alpha) \cdot E'_d(\omega).lb & \text{if } C_3 \end{cases} \tag{11}$$

C_1: the center point of region ω is scanned in the spatial domain and unscanned in the density domain.

C_2: the center point of region ω is unscanned in the spatial domain and is scanned in the density domain.
C_3: the center point of region ω is unscanned in both the spatial and density domains.

For partly scanned regions (i.e., a region ω is scanned in the spatial domain or in the density domain) and non-scanned regions, we define a global lower bound LB to estimate their spatial-density scores.

$$LB = \min_{\omega \in R_{pu}} \{E_{sd}(\omega, q)\} \tag{12}$$

Here, R_{pu} is a set of unscanned and partly scanned regions.

If the value of LB exceeds that of UB, the collaboration search terminates. The global upper bound UB and the corresponding region are returned. Other regions can be pruned safely.

4.2. Algorithm

Data: $G(V, E)$, O, q
Result: UB and the corresponding ω

```
1  LB ← 0; UB ← +∞; i ← 1;
2  while true do
3      //in the spatial domain
4      p ← expand(q);
5      if ω.c = p then
6          compute E_s(ω, q) and all related parameters;
7      end
8      if p is scanned in the two domains then
9          compute E_sd(c, q);
10         update LB and UB;
11         if LB > UB then
12             break;
13         end
14     end
15     //in the density domain
16     if ω.c = p_i then
17         compute E'_d(ω).lb and all related parameters;
18         i ← i + 1;
19     end
20     if p is scanned in the two domains then
21         compute E_sd(c, q);
22         update LB and UB;
23         if LB > UB then
24             break;
25         end
26     end
27 end
28 return UB and the corresponding ω;
```

Algorithm 1: Collaboration Search Algorithm

The collaboration search algorithm is detailed in Algorithm 1. Initially, the default value of global lower bound LB is set to 0, and the default value of global upper bound UB is set to $+\infty$. The iterator i is set to 1. In the spatial domain, we explore the spatial network and find the regions spatially close to the query point q using Dijkstra's algorithm. For each newly scanned vertex p, we compute the value of $E_s(\omega, q)$ ($\omega.c = p$) and all related parameters. If p has been fully scanned , we compute its distance-and-density score $E_{sd}(c, q).lb$. Then, we update the values of UB and LB. If the value of LB exceeds that of UB, the search process terminates (lines 1–14). In the density domain, the search is performed from the minimum to the maximum. In each step, the iterator i is incremented by 1. For each newly scanned vertex p_i, we compute the lower bound of its density score $E'_d(\omega).lb$ and all related parameters. If vertex p has been fully scanned, we update the values of UB and LB and check whether they meet the the search-stop criteria. By integrating these results, the region with the minimum value of $E_{sd}(\omega, q)$ is found (lines 15–28).

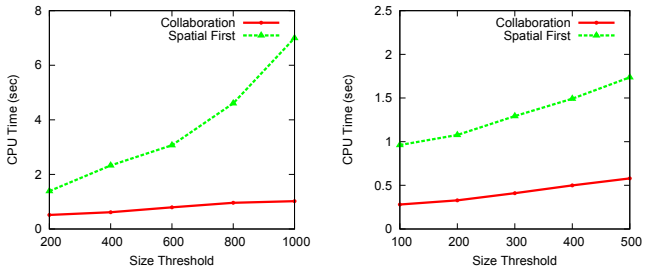
5. Experimental Results

In this section, we conducted extensive experiments on real and synthetic spatial data sets to study the performance of RD query processing. The data set used in our experiments were Beijing Road Network (BRN) and Oldenburg City Road Network (ORN)[6], which contain 28,342 vertices and 6,105 vertices respectively, stored as adjacency lists. In BRN, we use the real location based social media data (i.e., 100,000 points of interest). In ORN, the synthetic data were used (i.e., 10,000 points of interest). All algorithms were implemented in Java and tested on a Windows platform with Intel Core i7-3520M Processor (2.90GHz) and 8GB memory. All experimental results were averaged over 10 independent tests with different query inputs. For the purpose of comparison, a baseline method called spatial-first algorithm is also implemented. In the spatial-first algorithm, we search the regions spatially close to the query point q, and then we compute its corresponding density score. By combining the computation results, we find the region with the minimum spatial-density score.

5.1. Effect of $\tau.s$

Figure 3 presents the performance of the algorithms with varying size threshold $\tau.s$. Since the total number of spatial objects is fixed, a higher size threshold means fewer qualified regions. The sparser the region distribution, the larger the required search space, and thus the performance of query processing decreases. In Figure 3, the CPU time increases as the size threshold increases. It is clear that the CPU time required by the spatial-first search is 3–5 times higher than those needed by the collaboration search.

[6] www.cs.fsu.edu/ lifeifei/SpatialDataset.htm

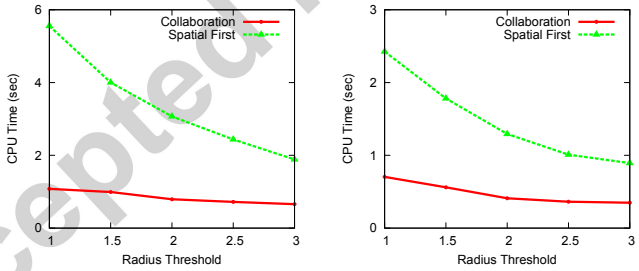


(a) BRN (b) ORN

Figure 3: Effect of $\tau.s$

5.2. *Effect of* $\tau.r$

Next, we vary the radius thresholds $\tau.r$. With a fixed value of size threshold, a larger radius threshold leads to more qualified regions. Intuitively, the denser the region distribution, the smaller the required search space, and thus the queries are expected to be faster. In Figure 4, the CPU time decreases as the radius threshold increases. The collaboration search outperforms the spatial-first search by factors of 3–5 in terms of CPU time.

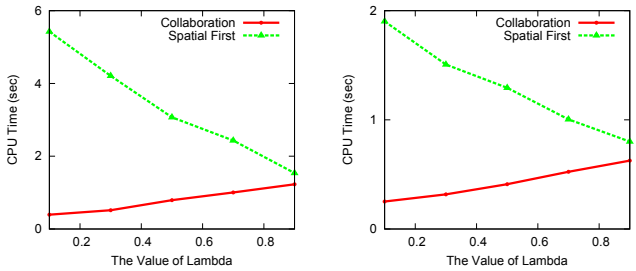


(a) BRN (b) ORN

Figure 4: Effect of $\tau.r$

5.3. *Effect of* α

Parameter α is used to adjust the relative importance of the spatial distance and density. In the extreme case where $\alpha = 1$, the query is conducted in the spatial domain only. And when $\alpha = 0$, density is the sole factor for the query. Figure 5 shows the performance of the algorithms for different values of α. For the collaboration algorithm, it is clear that the search effort required in the spatial domain is higher than that required in the density domain.



(a) BRN (b) ORN

Figure 5: Effect of α

6. Conclusion

In this paper, we propose and investigate a novel problem called Region Discovery (RD) query to find regions of interest with respect to a user's current geographic location. This type of query can bring significant benefits to users in many popular applications such as trip planning and location recommendation. To compute the query efficiently, a collaboration search algorithm was developed. A pair of upper and lower bounds were defined to prune the search space effectively. Finally, the performance of RD query was studied by extensive experiments on real and synthetic spatial data.

7. Acknowledgements

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), and the Excellent Talents of Beijing Program (No. 2013D009051000003).

8. Reference

[1] H.-J. Cho and C.-W. Chung. An efficient and scalable approach to cnn queries in a road network. In *VLDB*, pages 865–876, 2005.

[2] K. Deng, X. Zhou, H. T. Shen, K. Xu, and X. Lin. Surface k-nn query processing. In *ICDE*, page 78, 2006.

[3] L. Derczynski, B. Yang, and C. S. Jensen. Towards context-aware search and analysis on social media data. In *EDBT*, pages 137–142, 2013.

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

[5] 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.

[6] C. Guo, C. S. Jensen, and B. Yang. Towards total traffic awareness. *SIGMOD Record*, 43(3):18–23, 2014.

[7] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM TODS*, 24(2):265–318, 1999.

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

[9] C. S. Jensen, J. Kolarvr, T. B. Pedersen, and I. Timko. Nearest neighbor queries in road networks. In *ACM GIS*, pages 1–8, 2003.

[10] M. Kolahdouzan and C. Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *VLDB*, pages 840–851, 2004.

[11] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On trip planning queries in spatial databases. In *SSTD*, pages 273–290, 2005.

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

[13] K. Mouratidis, D. Papadias, and M. Hadjieleftheriou. Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring. In *SIGMOD*, pages 634–645, 2005.

[14] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *SIGMOD*, pages 71–79, 1995.

[15] C. Shahabi, M. R. Kolahdouzan, and M. Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. In *ACM GIS*, pages 94–100, 2002.

[16] S. Shang, K. Deng, and K. Xie. Best point detour query in road networks. In *ACM GIS*, pages 71–80, 2010.

[17] 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.

[18] 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.

[19] S. Shang, D. Guo, J. Liu, and K. Liu. Human mobility prediction and unobstructed route planning in public transport networks. In *MDM*, pages 43–48, 2014.

[20] S. Shang, J. Liu, K. Zheng, H. Lu, T. B. Pedersen, and J.-R. Wen. Planning unobstructed paths in traffic-aware spatial networks. *Geoinformatica*, online first:1–24, 2015.

[21] S. Shang, H. Lu, T. B. Pedersen, and X. Xie. Finding traffic-aware fastest paths in spatial networks. In *SSTD*, pages 128–145, 2013.

[22] S. Shang, B. Yuan, K. Deng, K. Xie, K. Zheng, and X. Zhou. Pnn query processing on compressed trajectories. *GeoInformatica*, 16(3):467–496, 2012.

[23] M. Sharifzadeh, M. Kolahdouzan, and C. Shahabi. The optimal sequenced route query. *VLDB Journal*, pages 765–787, 2008.

[24] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *VLDB*, pages 287–298, 2002.

[25] 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.

[26] B. Yang, C. Guo, Y. Ma, and C. S. Jensen. Toward personalized, context-aware routing. *VLDB J.*, 24(2):297–318, 2015.

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