Shang S, Chen L, We ZW *et al.* Dynamic shortest path monitoring in spatial networks. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 31(4): 637–648 July 2016. DOI 10.1007/s11390-016-1653-3

Dynamic Shortest Path Monitoring in Spatial Networks

Shuo Shang [1,2], Lisi Chen [3], Zhe-Wei Wei [4], Dan-Huai Guo [5,*], and Ji-Rong Wen [4], *Senior Member, IEEE*

[1] *State Key Laboratory of Software Development Environment, Beijing 100191, China*

[2] *Department of Computer Science, China University of Petroleum, Beijing 102249, China*

[3] *School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore*

[4] *Beijing Key Laboratory of Big-Data Management and Analysis Methods, Renmin University of China Beijing 100080, China*

[5] *Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China*

E-mail: jedi.shang@gmail.com; lchen012@e.ntu.edu.sg; zhewei@ruc.edu.cn; guodanhuai@cnic.cn jirong.wen@gmail.com

Received February 17, 2016; revised March 31, 2016.

Abstract With the increasing availability of real-time traffic information, dynamic spatial networks are pervasive nowadays and path planning in dynamic spatial networks becomes an important issue. In this light, we propose and investigate a novel problem of dynamically monitoring shortest paths in spatial networks (DSPM query). When a traveler aims to a destination, his/her shortest path to the destination may change due to two reasons: 1) the travel costs of some edges have been updated and 2) the traveler deviates from the pre-planned path. Our target is to accelerate the shortest path computing in dynamic spatial networks, and we believe that this study may be useful in many mobile applications, such as route planning and recommendation, car navigation and tracking, and location-based services in general. This problem is challenging due to two reasons: 1) how to maintain and reuse the existing computation results to accelerate the following computations, and 2) how to prune the search space effectively. To overcome these challenges, filter-and-refinement paradigm is adopted. We maintain an expansion tree and define a pair of upper and lower bounds to prune the search space. A series of optimization techniques are developed to accelerate the shortest path computing. The performance of the developed methods is studied in extensive experiments based on real spatial data.

Keywords shortest path, dynamic spatial network, spatial database, location-based service

1 Introduction

The continued proliferation of GPS-equipped mobile devices[1] (e.g., vehicle navigation systems and smart phones) and the proliferation of online map-based services (e.g., Google Maps①, Bing Maps②, and MapQuest③) enable people to acquire their current geographic positions in real time and to retrieve spatial information relevant to their travel. In the meantime, with the increasing availability of real-time traffic information, dynamic road networks are pervasive and path planning in dynamic spatial networks becomes an important issue. In this light, we propose and investigate a novel problem of efficiently monitoring shortest paths in dynamic spatial networks. When a traveler targets at a destination, his/her shortest path to the destina-

Regular Paper

Special Section on Data Management and Data Mining 2016

This work is partially supported by the National Natural Science Foundation of China under Grant Nos. 61402532 and 41371386, the Science Foundation of China University of Petroleum-Beijing under Grant No. 2462013YJRC031, the Excellent Talents of Beijing Program under Grant No. 2013D009051000003, Beijing Nova Program, and the Open Research Fund Program of Shenzhen Key Laboratory of Spatial Smart Sensing and Services (Shenzhen University).

*Corresponding Author

①http://maps.google.com/, April 2016.

②http://www.bing.com/maps/, April 2016.

③http://www.mapquest.com, April 2016.

tion may be continuously changing due to two reasons: 1) the travel cost of some edges has changed (e.g., at peak hours, 7:00 am∼9:00 am or 17:00 pm∼19:00 pm, the travel costs may increase) and 2) the traveler deviates from the pre-planned path. Our target is to accelerate the shortest path computing in dynamic spatial networks, and we believe that this study may be useful in many mobile applications, such as route planning and recommendation, car navigation and tracking, and location-based services in general.

Fig.1 shows two cases of shortest path update: 1) some edges' travel costs have changed and 2) the moving object deviates from the pre-planned path. Here, p_1, p_2, ..., p_8 are vertices in a spatial network. A moving object o's current position is p_1, and p_2 is the destination. Path $r_1 = \langle p_1, p_4, p_3, p_2 \rangle$ is a pre-planned shortest path from p_1 to p_2. In case 1), an edge $\langle p_1, p_4 \rangle$ is overcrowded and its travel cost increases.

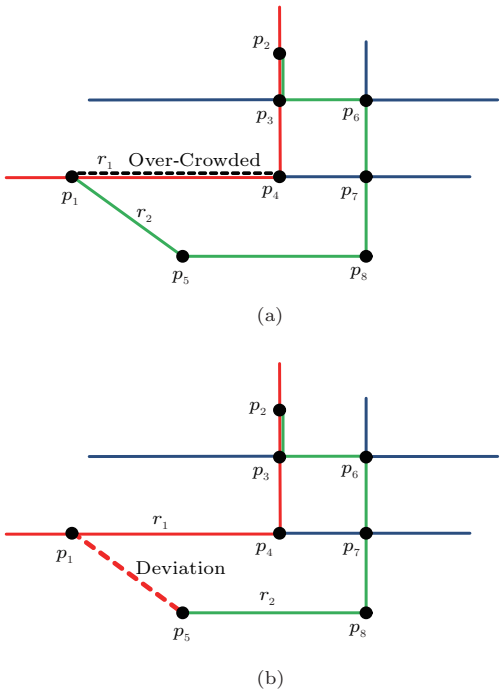

Fig.1. Two cases of shortest path update. (a) Travel-cost change. (b) Deviation.

Hence, the shortest path r_1 expires and is replaced by a new shortest path $r_2 = \langle p_1, p_5, p_8, p_7, p_6, p_3, p_2 \rangle$. In case 2), a moving object o deviates from the pre-planned path r_1 and it arrives at p_5. Compared with going back to r_1, path $r_2 = \langle p_5, p_8, p_7, p_6, p_3, p_2 \rangle$ has less travel cost and r_2 is the new shortest path from o's current position to p_2. Our target is to accelerate the new shortest path computing in the aforementioned two cases.

The proposed dynamic shortest path monitoring (DSPM) query is applied in spatial networks, since in a large number of practical scenarios, objects move in such networks (e.g., roads, railways, rivers) rather than in a Euclidean space. A straightforward approach of the DSPM query is recomputing the shortest path from a moving object's current position to the destination using network expansion method (e.g., Dijkstra's algorithm[2] or A* algorithm[3]). However, especially in a large spatial network, recomputing shortest paths using network expansion method is time-consuming, if the network is updated frequently or the moving object does not follow the pre-planned path. Existing network expansion methods are lack of effective pruning techniques to prune the search space. Such high computation cost may prevent the DSPM query from being answered efficiently. To the best of our knowledge, there is no existing method that can process the DSPM query efficiently. Existing studies of dynamic path planning[4-5] are mainly based on pre-defined traffic models (e.g., time-dependent road networks), and instant traffic conditions and moving object deviations are not taken into account.

To overcome the weakness of the baseline method, we propose a novel two-phase search algorithm to compute the DSPM query efficiently. Initially, we compute the shortest path from the source to the destination using network expansion method. We maintain the expansion tree and compute the upper and lower bounds of the shortest path distances (from the vertices in the tree to the destination). If a moving object travels along with the shortest path and there is no update (travel cost change) in the expansion tree, it is not necessary to recompute the shortest path. Otherwise, we have to recompute a new shortest path from the moving object's current location to the destination. We adopt filter-and-refinement paradigm and we use the upper and lower bounds to prune the search space. We can avoid devoting unnecessary search effort to paths unlikely to be the optimal choice and further enhance the query efficiency.

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

• We study a novel problem of monitoring shortest path in dynamic spatial networks (DSPM query). It provides new features for advanced spatiotemporal information systems, and benefits users in many popular mobile applications such as route planning and recommendation, car navigation and tracking, and location-based services in general.

• We define a pair of upper and lower bounds of shortest path distances to prune the search space effectively.

• We develop two efficient algorithms to compute the DSPM query in two cases, and the filter-and-refinement paradigm is adopted.

• We conduct extensive experiments to investigate the performance of the developed algorithms on real spatial data.

The rest of the paper is organized as follows. Section 2 introduces dynamic spatial networks used in this paper as well as the problem definition. The dynamic shortest path monitoring (DSPM) query processing is introduced in Section 3, which is followed by the experimental results in Section 4. This paper is concluded in Section 6 after discussions on related work in Section 5.

2 Preliminaries

2.1 Spatial Networks

A dynamic 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 the travel between vertices v_i and v_j. Function $F{:}V \cup E \rightarrow Geometries$ records the 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 edge e represents the corresponding road segment's travel cost or some other relevant properties such as its travel time[6], which may be obtained from instant traffic data, and hence $w(e)$ is a dynamic value.

In this paper, $w(e)$ represents the travel time along e and we define that

$$w(e) = \frac{e.dist}{e.speed},$$

where $e.dist$ is the network length of edge e and $e.speed$ is the instant travel speed along e. Suppose p_a and p_b are two end vertices of e, $E_{\text{dist}}(p_a, p_b)$ is the Euclidean distance between p_a and p_b and we have $E_{\text{dist}}(p_a, p_b) \leqslant e.dist$. Supposing $S_{\max}$ is the maximum speed, we have that

$$w(e) \geqslant \frac{E_{\text{dist}}(p_a, p_b)}{S_{\max}}.$$

2.2 Problem Definitions

Shortest Path. Given two vertices p_a and p_b in a dynamic 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 travel time is denoted by $sd(p_a, p_b)$.

DSPM Query. Given a moving object o, a destination d, and a pre-defined shortest path P, if 1) $\exists e \in G.E, w(e)$ changes, or 2) $o \notin P$, the DSPM query recomputes $SP(o, d)$.

3 Query Processing

Initially, we compute the shortest path from a source s to a destination d according to the A* algorithm[3]. Network expansion is expanded from s, and s is the root of an expansion tree. We put s into a priority heap H, at each time, select vertex p with the minimum distance label from H for network expansion, and then put the adjacent vertices of p into H, until the destination is reached. For vertex p, its distance label is defined by

$$p.dist = sd(s, p) + cost(p),$$

where $sd(s, p)$ is the network distance between s and p, and $cost(p) = E_{\text{dist}}(p, d)/S_{\max}$ is the heuristic part, and we have that $E_{\text{dist}}(p, d)/S_{\max} \leqslant sd(p, d)$.

Two cases may trigger the shortest path update: 1) the travel cost of some edges has changed and 2) a moving object deviates from the pre-planned path $SP(s, d)$. In Subsection 3.1, we introduce the definition and computation of shortest path distance upper bound and how we can use the upper bound to accelerate shortest path computation in the first case. In Subsection 3.2, we introduce the definition and computation of shortest path distance lower bound and how we can use both the upper and the lower bounds to accelerate shortest path computation in the second case.

3.1 Travel Cost Change

When the shortest path $SP(s, d)$ is planed, we maintain the expansion tree. An example is shown in Fig.2, where $s = p_1$ is the root of the expansion tree, and $d = p_9$ is the destination.

Lemma 1. *If the expansion tree T is not updated, the shortest path is not updated.*

Proof. Suppose p is a vertex which is not belonging to the expansion tree ($p \notin T$). Since network expansion

always selects the vertex with the minimum distance label for expansion, for a vertex $p_i \in T$, we have that

$$sd(p_i, p) + cost(p) > cost(p_i).$$

Otherwise, we will have

$$\begin{aligned} & sd(s, p_i) + sd(p_i, p) + cost(p) < sd(s, p_i) + cost(p_i) \\ \Rightarrow\ & sd(s, p) + cost(p) < sd(s, p_i) + cost(p_i), \end{aligned}$$

which conflicts with $p \notin T$. □

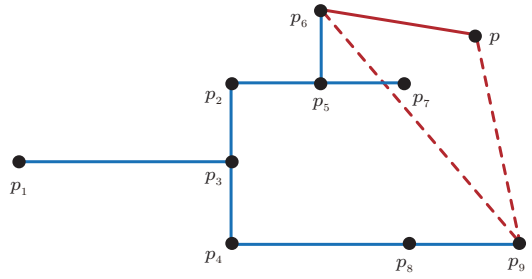

Fig.2. Expansion tree.

For example, in Fig.2, $sd(p_6, p)+cost(p) > cost(p_6)$. If the travel cost of $sd(p_6, p)$ is updated to $sd'(p_6, p)$ (no matter $sd(p_6, p) > sd'(p_6, p)$ or $sd(p_6, p) < sd'(p_6, p)$), we have that $sd(p_6, p) \geqslant E_{\text{dist}}(p_6, p)/S_{\max}$. According to the triangular inequality, we have that

$$\begin{aligned} & \frac{E_{\text{dist}}(p_6, p)}{S_{\max}} + \frac{E_{\text{dist}}(p, p_9)}{S_{\max}} > \frac{E_{\text{dist}}(p_6, p_9)}{S_{\max}} \\ \Rightarrow\ & sd'(p_6, p) + \frac{E_{\text{dist}}(p, p_9)}{S_{\max}} > \frac{E_{\text{dist}}(p_6, p_9)}{S_{\max}} \\ \Rightarrow\ & sd'(p_6, p) + cost(p) > cost(p_6). \end{aligned}$$

Therefore, if the expansion tree is not updated, there does not exist a new shortest path via p to the destination.

In Fig.2, $SP(s, d) = \langle p_1, p_3, p_4, p_8, p_9 \rangle$ is the shortest path from $s = p_1$ to $d = p_9$, and we maintain $SP(s, d)$ as an accessible path from s to d. When the travel costs of some edges update, $SP(s, d)$ may not be the shortest path but its length $sd(s, d)$ can be used as the upper bound of the shortest path distance. If the network is updated frequently, we may have several accessible paths and we select the one with the minimum length as the upper bound of shortest path distance UB:

$$UB = \min\{sd_i(s, d)\}.$$

The search process of DSPM query processing in the first case is detailed in Algorithm 1. In Algorithm 1, the query inputs include a graph $G(V, E)$, a source s, a destination d, an existing shortest path $SP'(s, d)$ and expansion tree $T(s)$. Then, the travel costs of some edges are updated. If the expansion tree $T(s)$ is not updated, the shortest path is not updated and we return $SP'(s, d)$ (lines 1 and 2). Otherwise, we have to recompute the shortest path, and $SP'(s, d)$ is used as an accessible path and its length is used as the shortest path upper bound UB. Initially, the priority heap O_s is set to null and the distance labels of all vertices are set to $+\infty$. We put s into O_s (lines 3∼6). At each time, we select the vertex v with the minimum distance label from O_s for expansion, until the destination d is reached.

Algorithm 1. DSPM Query Processing in the First Case

```
   Data: G(V, E), s, d, SP'(s, d), and T(s)
   Result: shortest path SP(s, d)
1  if ∀e ∈ T(s), w(e) is not updated then
2      Return SP'(s, d);
3  UB = min{sd_i(s, d)};
4  O_s ← ∅;
5  ∀v ∈ G(V, E)(v.dist ← +∞);
6  O_s.push(s);
7  while O_s ≠ ∅ do
8      Select v ∈ O_s with the minimum distance label v.dist;
9      O_s.remove(v);
10     if v = d then
11         while v.pre ≠ null do
12             SP(s, d).push(v);
13             v ← v.pre;
14         return SP(s, d);
15     for each vertex n ∈ v.adjacentList do
16         if n.dist > sd(s, v) + w(v, n) + cost(n) then
17             if sd(s, v) + w(v, n) + cost(n) < UB then
18                 n.dist ← sd(s, v) + w(v, n) + cost(n);
19                 n.pre ← v;
20                 O_s.push(n);
```

Then, we compute and return the new shortest path $SP(s, d)$ (lines 7∼14). For each adjacent vertex n of vertex v, we compare its distance label $n.dist$ to $sd(s, v) + w(v, n) + cost(n)$. If $n.dist$ is greater than $sd(s, v) + w(v, n) + cost(n)$ and $sd(s, v) + w(v, n) + cost(n)$ is less than UB, we update the value of $n.dist$ and put n into O_s (lines 15∼20).

3.2 Deviation

Once a moving object deviates from the pre-planned path, we have to recompute the shortest path from the moving object's current position to the destination.

An example of moving object deviation is shown

in Fig.3, where $p, p_1, ..., p_9$ are vertices, $s = p_1$ and $d = p_9$, and $SP(s,d) = (p_1, p_3, p_4, p_8, p_9)$ is the pre-planned shortest path. Vertex $s = p_1$ is the root of the expansion tree. A moving object o deviates from $SP(s,d)$ and p is its current position.

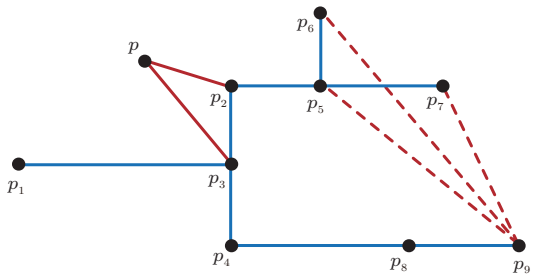

Fig.3. Deviation.

For each vertex p_i in the expansion tree, we compute a lower bound of the shortest path distance $sd(p_i, d)$. The computation procedure is from leaf nodes to the root. For example, in Fig.3, p_6, p_7 and p_9 are leaf nodes. For a leaf node p_j, its distance lower bound is defined by

$$sd(p_j, d).lb = cost(p_j) = E_{\text{dist}}(p_j, d)/S_{\max},$$

and we have that $sd(p_j, d).lb \leqslant sd(p_j, d)$.

For a non-leaf node p_k, its distance lower bound is defined by

$$sd(p_k, d).lb = \min_{\forall p_j (p_j.pre = p_k)} \{sd(p_j, d).lb + sd(p_k, p_j)\},$$

where p_k is the parent node of p_j in the expansion tree, and it is clear that $sd(p_k, d).lb \leqslant sd(p_k, d)$.

For each vertex in $SP(s,d)$, we compute and record its shortest path distance to d. To find the shortest path from a moving object's current position p to the destination d, network expansion is conducted from p. Once a vertex in $SP(s,d)$ is scanned (e.g., vertex p_3 in Fig.3), we have an accessible path from p to d. Path $P = \langle p, p_3, p_4, p_8, p_9 \rangle$ is an accessible path, and we can compute its length as $P.dist = sd(p, p_3) + sd(p_3, p_9)$. The lengths of accessible paths can be used to define the upper bound of shortest path distance UB.

$$UB = \min\{P_i.dist\},$$

where P_i is an accessible path.

On the other hand, if a vertex in the expansion tree T is scanned (e.g., vertex p_2 in Fig.3), we can estimate the shortest path distance from p to d via p_2 as

$$sd(p, p_2, d) \geqslant sd(p, p_2) + sd(p_2, d).lb$$

$$\Downarrow$$

$$sd(p, p_2, d).lb = sd(p, p_2) + sd(p_2, d).lb.$$

If the value of $sd(p, p_2, d).lb$ exceeds UB, there does not exist a shortest path from p to d via p_2; thus p_2 can be pruned safely.

The search process of DSPM query processing in the second case is detailed in Algorithm 2.

Algorithm 2. DSPM Query Processing in the Second Case

```
   Data: G(V,E), s, d, p, SP(s,d), and T(s)
   Result: shortest path SP(p,d)
1  O_s ← ∅;
2  ∀v ∈ G(V,E)(v.dist ← +∞);
3  UB ← +∞;
4  O_s.push(s);
5  while O_s ≠ ∅ do
6      Select v ∈ O_s with the minimum distance label v.dist;
7      O_s.remove(v);
8      if v = d then
9          while v.pre ≠ null do
10             SP(p,d).push(v);
11             v ← v.pre;
12         return SP(p,d);
13     if v ∈ SP(s,d) then
14         if UB > sd(p,v) + sd(v,d) then
15             UB = sd(p,v) + sd(v,d);
16     if v ∈ T(s) then
17         if sd(p,v) + sd(v,d).lb > UB then
18             Continue;
19     for each vertex n ∈ v.adjacentList do
20         if n.dist > sd(p,n) + w(v,n) + cost(n) then
21             if sd(p,n) + w(v,n) + cost(n) < UB then
22                 n.dist ← sd(p,n) + w(v,n) + cost(n);
23                 n.pre ← v;
24                 O_s.push(n);
```

In Algorithm 2, the query inputs include a graph $G(V,E)$, a source s, a destination d, a moving object's current position p, an existing shortest path $SP(s,d)$, and expansion tree $T(s)$. Initially, the priority heap O_s is set to null, the distance labels of all vertices are set to $+\infty$, and the upper bound UB is set to $+\infty$ (lines 1∼3). We put source s into O_s (line 4). At each time, we select vertex v with the minimum distance label from O_s for expansion, until destination d is reached. Then, we compute and return the new shortest path $SP(p,d)$ (lines 5∼12). If v is in the shortest path $SP(s,d)$, we

compare and update the value of UB (lines 13∼15). If v is in the expansion tree $T(s)$, we compute the lower bound of shortest path distance $sd(p,v,d).lb = sd(p,v) + sd(v,d).lb$. If $sd(p,v) + sd(v,d).lb > UB$, there does not exist a shortest path from p to d via v, and v can be pruned safely (lines 16∼18). For each adjacent vertex n of vertex v, we compare its distance label $n.dist$ with $sd(p,n) + w(v,n) + cost(n)$. If $n.dist$ is greater than $sd(p,v) + w(v,n) + cost(n)$ and $sd(p,v) + w(v,n) + cost(n)$ is less than UB, we update the value of $n.dist$ and put n into O_s (lines 19∼24).

4 Experiments

We conducted extensive experiments on real spatial datasets to demonstrate the performance of the proposed dynamic shortest path monitoring (DSPM) query. The two datasets used in our experiments were Beijing Road Network (BRN) and North America Road Network (NRN), which contain 28 342 vertexes and 175 812 vertexes respectively, stored in adjacency lists.

In the experiments, the graphs were memory resident when running A* algorithm[3], as the memory occupied by BRN or NRN was less than 20 MB. All algorithms were implemented in Java and run on a Windows 7 platform with an Intel i7-4770k processor (3.50 GHz) and 16 GB memory. Unless stated otherwise, 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 used as a metric since it describes the number of data accesses.

The parameter settings are detailed in Table 1. In both BRN and NRN, the shortest path length (the number of vertices in the shortest path) varies from 20 to 100, and 60 is the default value. In the first case (travel cost change), by default, 3% of all edges change their travel costs. In the second case (deviation), the deviation distance varies from 1 to 5, and 3 is the default value. The deviation distance is defined by the shortest path distance from the moving object's current position to the deviation point.

Table 1. Parameter Settings

	Shortest Path Length	Percentage of Travel Cost Change	Deviation Distance
BRN	20∼100 (default 60)	1%∼5% (default 3%)	1∼5 (default 3)
NRN	20∼100 (default 60)	1%∼5% (default 3%)	1∼5 (default 3)

4.1 Travel-Cost Change

Initially, we tested the performance of the DSPM algorithm of the first case (travel cost change) (DSPM-1 algorithm) in Fig.4. Intuitively, a longer shortest path means a larger search space. The shortest path length varies from 20 to 100 (the number of vertices in the shortest path). The CPU time and the number of visited vertices are expected to be higher for both A* algorithm and DSPM-1 algorithm. With the help of effective pruning techniques (refer to Algorithm 1), the DSPM-1 algorithm outperforms the A* algorithm by almost a factor of 3 (for both CPU time and the number of visited vertices). Moreover, the change of some edges' travel costs may weaken the pruning effectiveness of DSPM-1 algorithm; thus, a higher percentage of travel cost may lead to a higher computation cost (for both CPU time and the number of visited vertices). However, DSPM-1 algorithm still outperforms A* algorithm by a factor of 2 in terms of both CPU time and visited vertices.

4.2 Deviation

Fig.5 presents the performance of the DSPM algorithm in the second case (deviation). Similarly, a longer shortest path length causes a higher computation cost for both A* algorithm and DSPM-1 algorithm. With the help of effective pruning techniques (refer to Algorithm 2), DSPM-2 algorithm outperforms A* algorithm by a factor of 2 (for both CPU time and the number of visited vertices). In addition, for the DSPM-2 algorithm, a longer deviation distance may cause a larger search space hence a higher computation cost. The deviation distance varies from 1 to 5. With the help of the expansion tree and the upper and lower bounds of shortest path distance (refer to Algorithm 2), the performance is improved by approximately a factor of 2 in terms of both CPU time and visited vertices.

4.3 Combination

Finally, we combined Algorithm 1 and Algorithm 2 to test the performance of the DSPM algorithm in both two cases, and the experimental results are shown in Fig.6. The shortest path distance varies from 20 to 100 (the number of vertices in the shortest path), and the deviation distance and the percentage of travel cost change are with their default values. As introduced before, a longer shortest path leads to a higher computation cost for both CPU time and the number of visited vertices. Compared with the baseline algorithm,

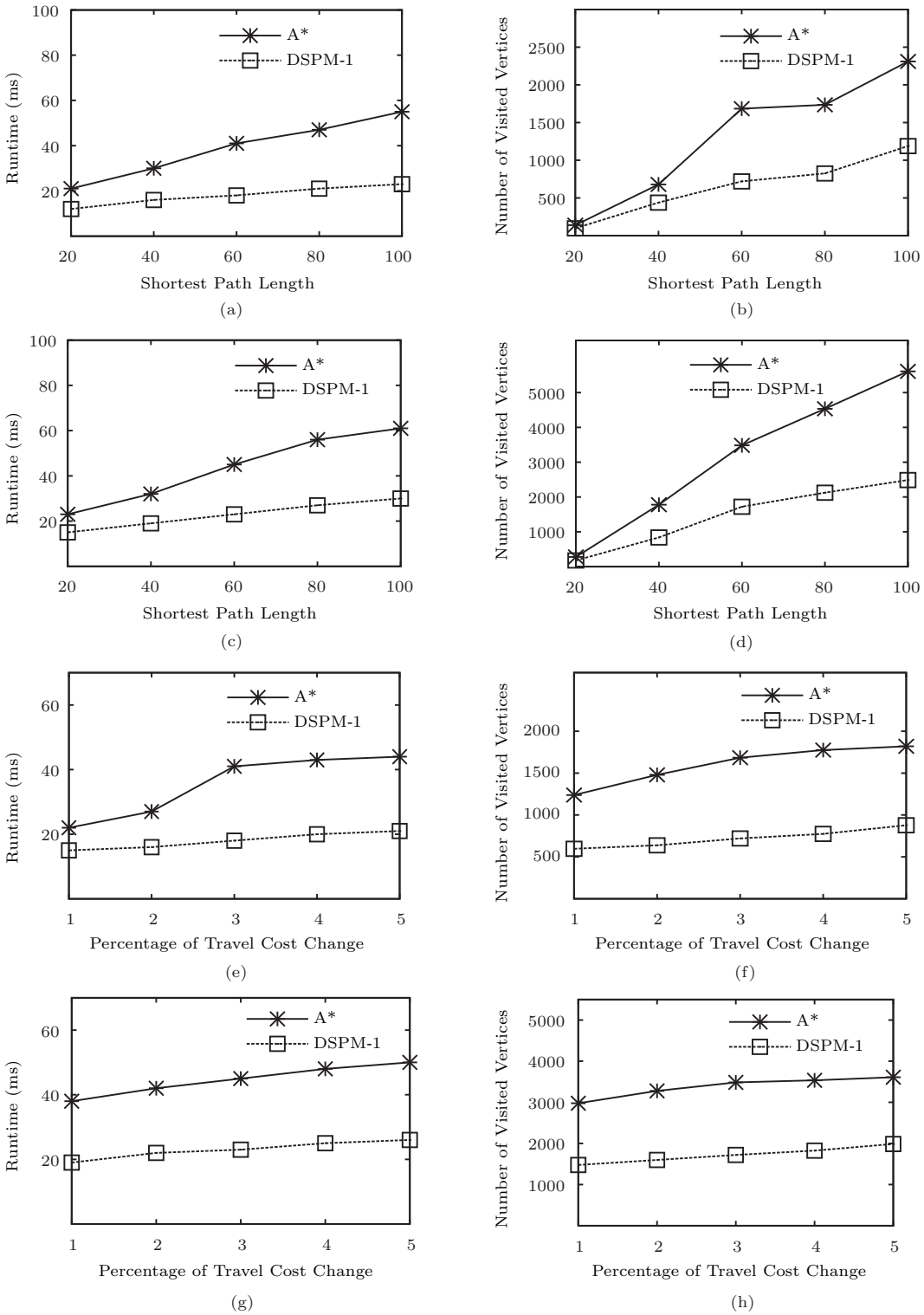

Fig.4. Performance of the DSPM algorithm in the first case (DSPM-1). (a) CPU-time with respect to the shortest path length of BRN. (b) Number of visited vertices with respect to the shortest path length of BRN. (c) CPU-time with respect to the shortest path length of NRN. (d) Number of visited vertices with respect to the shortest path length of NRN. (e) CPU-time with respect to travel cost change of BRN. (f) Number of visited vertices with respect to travel cost change of BRN. (g) CPU-time with respect to travel cost change of NRN. (h) Number of visited vertices with respect to travel cost change of NRN.

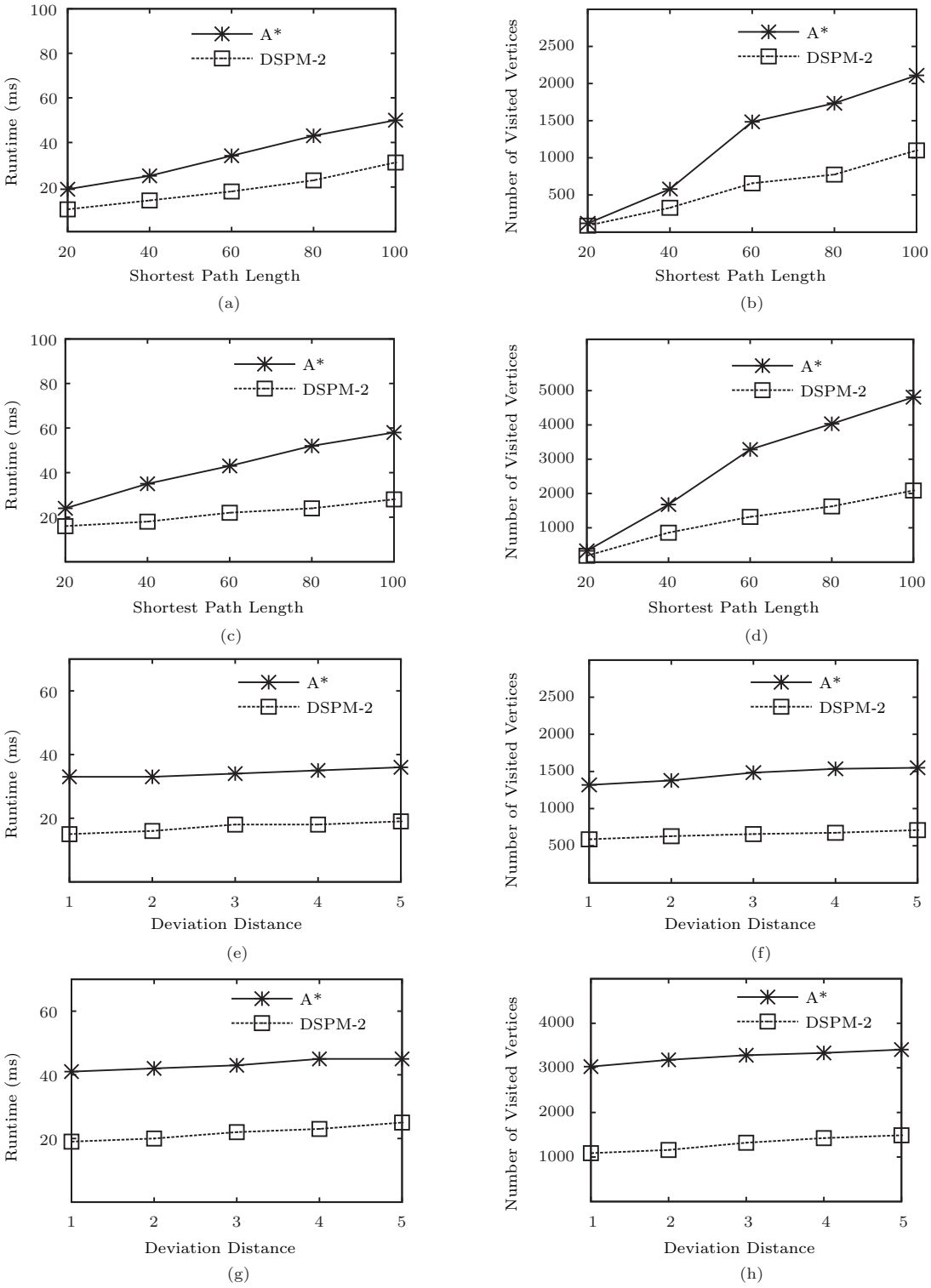

Fig.5. Performance of the DSPM algorithm in the second case (DSPM-2). (a) CPU-time with respect to the shortest path length of BRN. (b) Number of visited vertices with respect to the shortest path length of BRN. (c) CPU-time with respect to the shortest path length of NRN. (d) Number of visited vertices with respect to the shortest path length of NRN. (e) CPU-time with respect to deviation distance of BRN. (f) Number of visited vertices with respect to deviation distance of BRN. (g) CPU-time with respect to deviation distance of NRN. (h) Number of visited vertices with respect to deviation distance of NRN.

the DSPM algorithms of both cases still have a high query performance, and outperform the baseline algorithm by a factor of 2 in terms of both CPU time and visited vertices. It is worth to note that the CPU time is not fully aligned with the number of visited vertices. To prune the search space, the DSPM 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 Related Work

5.1 Path Planning Queries

Network-based path planning[7] is pervasive. Generally, path planning queries can be classified into two categories: path planning in static spatial networks, and path planning in dynamic spatial networks. In static spatial networks, Dijkstra's algorithm[2] and A* algorithm[3] are two general methods based on network expansion to compute the shortest path from a single source to a destination. In Dijkstra's algorithm, a network expansion is expanded from a source s. A priority heap H is adopted to maintain the unscanned vertexes. At each time, a vertex v with the minimum distance label is selected. It is removed from H and labeled as "scanned" vertex. Then, all unscanned neighbor vertices of v are put into H, until the destination d is reached and the shortest path from s to d is found. In the A* algorithm, the value of $sd(s,v)+d_E(v,d)$, where $d_E(v,d)$ is the Euclidean distance between v and d, is used as the distance label of vertex v, to estimate the network distance from s to d via v. The A* algorithm is a heuristic search method, and its performance is greater than that of Dijkstra's algorithm in general.

Group nearest neighbor[8], aggregate nearest neighbor[9], and collective travel planning[7] queries have multiple sources and a single destination. Group nearest neighbor and aggregate nearest neighbor queries assume that each traveler goes to the destination directly, and the collective travel planning query assumes that travelers go to the destination via k meeting points. They may go to the meeting point individually, and then go together to the destination by collective travel. The collective travel planning problem is Max-SNP hard.

Time-dependent road network[4,10-11] and probabilistic road network[5] are two representative dynamic spatial networks. A time-dependent road network can be established according to speed patterns from historical traffic information, which motivates the time-dependent shortest path query[4]. Time-dependent shortest path query is a variant of dynamic shortest path problem, which is designed to find the best departure time for users, to minimize the global traveling time from a source to a destination over a large road network, where the traffic conditions are dynamically changing from time to time. The challenge of this problem lies in the dynamic edge delay. In probabilistic road networks, each edge is assigned a set of probabilistic data to describe the traveling cost along this edge, and probabilistic shortest path queries[5] ask for 1) the fastest path constrained by a probability threshold, and 2) the path with the highest probability constrained by a travel time threshold. There also exist other dynamic spatial networks. For example, Shang *et al.*[10-12] used trajectory data to establish traffic-aware spatial networks, and then they planned the fastest paths based on a probabilistic threshold.

Despite the bulk of literatures on shortest path queries[2-5,10-11], none of existing work can address the proposed DSPM query due to two reasons: Dijkstra's algorithm and A* algorithm are based on static spatial networks, and time-dependent and probabilistic shortest[5,12] path queries are based on different traffic models from ours (their models are mainly based on pre-computation, historical traffic information, and fixed traffic models, and instant traffic information is not taken into account; instant traffic information, e.g., traffic accidents, is not predictable).

5.2 Trajectory-Based Travel Planning

The trajectory-based travel planning queries can be further divided into trajectory-to-object search and trajectory-to-trajectory search. In the trajectory-to-object search, queries aim to find objects spatially close to a query path according to some distance metrics. For example, the path nearest neighbor (PNN) query[13-14] maintains an up-to-date path nearest neighbor result as the user is moving along a predefined route. Moreover, the path nearby cluster query[15] further extends the PNN query to find the POI clusters spatially close to a given path. In trajectory-to-trajectory search, queries retrieve the trajectories that have similar curve and are spatially close to a query trajectory. Travelers can use the travel history of other travelers to guide their own trips. Zheng *et al.* studied this problem in Euclidean space[16], and Shang *et al.* studied the problem in spatial networks[17-18].

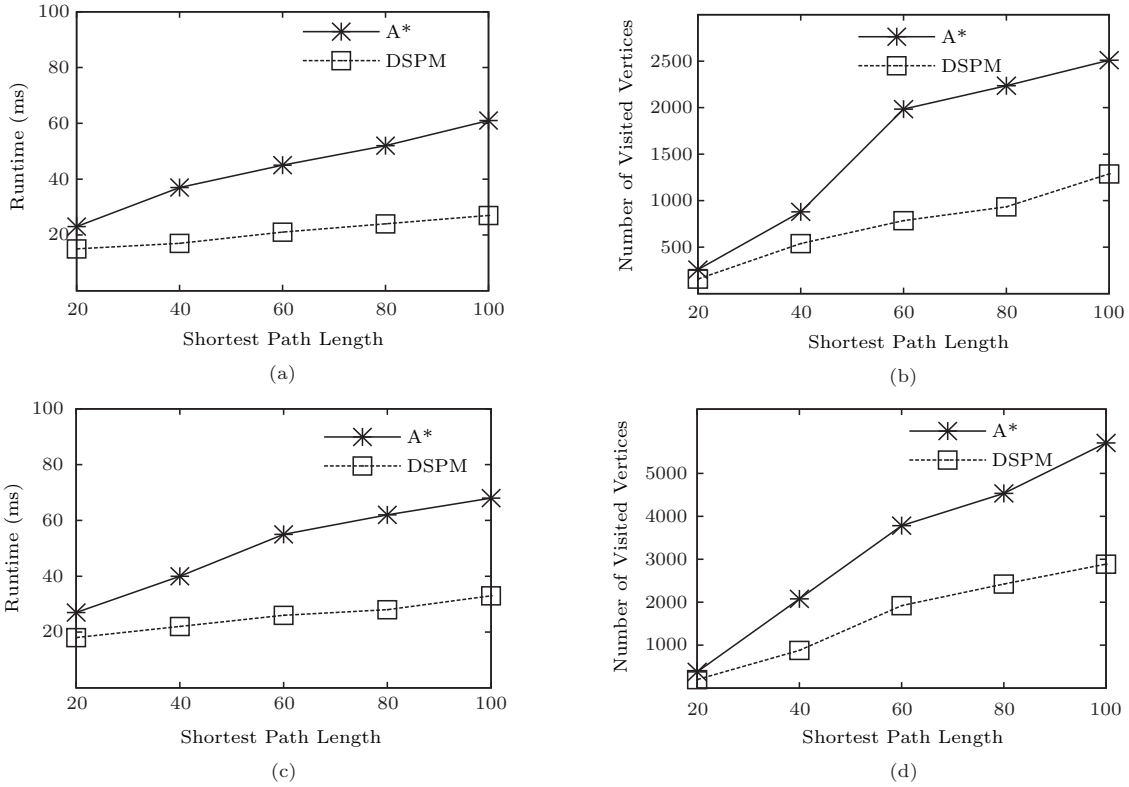

Fig.6. Performance for the DSPM algorithm (combined). (a) BRN (CPU-time). (b) BRN (visited vertices). (c) NRN (CPU-time). (d) NRN (visited vertices).

There also exist several interesting directions on path planning[19-20], spatial-social information processing[21-26], and network information processing[27-31], which may be considered in our future studies.

6 Conclusions

In this paper, we proposed and investigated a novel problem of planning shortest paths in dynamic spatial networks, and the change of some edges' travel cost and the deviation of moving objects are taken into account. Our target is to accelerate the shortest path computing in dynamic spatial networks, and we believe that this study may be useful in many mobile applications, such as route planning and recommendation, car navigation and tracking, and location-based services in general. We developed two efficient algorithms based on filter-and-refinement paradigm, and we devised a series of pruning techniques to prune the search space effectively. The performance of the developed methods was studied in extensive experiments based on real spatial data.

References

[1] Parkinson B, Spiker Jr J, Axelrad P, Enge P. Global positioning system: Theory and applications. In *Progress in Astronautics and Aeronautics 163*, Zarchan P(ed.), American Institute of Aeronautics and Astronautics, Inc., 1996.

[2] Dijkstra E W. A note on two problems in connection with graphs. *Numerische Mathematik*, 1959, 1(1): 269-271.

[3] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 1968, 4(2): 100-107.

[4] Ding B, Yu J X, Qin L. Finding time-dependent shortest paths over large graphs. In *Proc. the 11th EDBT*, March 2008, pp.205-216.

[5] Hua M, Pei J. Probabilistic path queries in road networks: Traffic uncertainty aware path selection. In *Proc. the 13th EDBT*, March 2010, pp.347-358.

[6] Yang B, Guo C, Jensen C S *et al.* Stochastic skyline route planning under time-varying uncertainty. In *Proc. the 30th IEEE International Conference on Data Engineering*, March 31-April 4, 2014, pp.136-147.

[7] Shang S, Chen L, Wei Z *et al.* Collective travel planning in spatial networks. *IEEE Trans. Knowl. Data Eng.*, 2016, 28(5): 1132-1146.

[8] Papadias D, Shen Q, Tao Y *et al.* Group nearest neighbor queries. In *Proc. the 20th ICDE*, March 30-April 2, 2004, pp.301-312.

[9] Papadias D, Tao Y, Mouratidis K *et al.* Aggregate nearest neighbor queries in spatial databases. *ACM Transactions on Database Systems*, 2005, 30(2): 529-576.

[10] Shang S, Lu H, Pedersen T B *et al.* Finding traffic-aware fastest paths in spatial networks. In *Proc. the 13th SSTD*, Aug. 2013, pp.128-145.

[11] Shang S, Lu H, Pedersen T B *et al.* Modeling of traffic-aware travel time in spatial networks. In *Proc. the 14th IEEE MDM*, June 2013, pp.247-250.

[12] Shang S, Liu J, Zheng K *et al.* Planning unobstructed paths in traffic-aware spatial networks. *GeoInformatica*, 2015, 19(4): 723-746.

[13] Chen Z, Shen H T, Zhou X. Monitoring path nearest neighbor in road networks. In *Proc. ACM SIGMOD*, June 2009, pp.591-602.

[14] Shang S, Yuan B, Deng K *et al.* PNN query processing on compressed trajectories. *GeoInformatica*, 2012, 16(3): 467-496.

[15] Shang S, Zheng K, Jensen C S *et al.* Discovery of path nearby clusters in spatial networks. *IEEE Transactions on Knowledge and Data Engineering*, 2015, 27(6): 1505-1518.

[16] Zheng K, Shang S, Yuan N J *et al.* Towards efficient search for activity trajectories. In *Proc. the 29th ICDE*, April 2013, pp.230-241.

[17] Shang S, Ding R, Yuan B *et al.* User oriented trajectory search for trip recommendation. In *Proc. the 15th EDBT*, March 2012, pp.156-167.

[18] Shang S, Ding R, Zheng K *et al.* Personalized trajectory matching in spatial networks. *The VLDB Journal*, 2014, 23(3): 449-468.

[19] Wang F, Zhu Z. Global path planning of wheeled robots using multi-objective memetic algorithms. In *Proc. the 14th IDEAL*, Oct. 2013, pp.437-444.

[20] Guo X, Zhang D, Wu K *et al.* MODLoc: Localizing multiple objects in dynamic indoor environment. *IEEE Transactions on Parallel and Distribution Systems*, 2014, 25(11): 2969-2980.

[21] Shang S, Xie K, Zheng K *et al.* VID join: Mapping trajectories to points of interest to support location-based services. *Journal of Computer Science and Technology*, 2015, 30(4): 725-744.

[22] Li B, Tan S, Wang M *et al.* Investigation on cost assignment in spatial image steganography. *IEEE Transactions on Information Forensics and Security*, 2014, 9(8): 1264-1277.

[23] Li B, Wang M, Li X *et al.* A strategy of clustering modification directions in spatial image steganography. *IEEE Transactions on Information Forensics and Security*, 2015, 10(9): 1905-1917

[24] Yang X S, Pei J, Sun W. Elastic image registration using hierarchical spatially based mean shift. *Comp. in Bio. and Med.*, 2013, 43(9): 1086-1097.

[25] Zhou F, Jiao J, Lei B Y. A linear threshold-hurdle model for product adoption prediction incorporating social network effects. *Inf. Sci.*, 2015, 307: 95-109.

[26] Wang J, Huang J Z, Guo J *et al.* Recommending high-utility search engine queries via a query-recommending model. *Neurocomputing*, 2015, 167(C): 195-208.

[27] Dai M, Sung C. Achieving high diversity and multiplexing gains in the asynchronous parallel relay network. *Trans. Emerging Telecommunications Technologies*, 2013, 24(2): 232-243.

[28] Zhang D, Lu K, Mao R. A precise RFID indoor localization system with sensor network assistance. *China Communications*, 2015, 12(4): 13-22.

[29] Huang X, Cheng H, Li R H *et al.* Top-k structural diversity search in large networks. *VLDB J.*, 2015, 24(3): 319-343.

[30] Wu R, Li C, Lu D. Power minimization with derivative constraints for high dynamic GPS interference suppression. *SCIENCE CHINA Information Sciences*, 2012, 55(4): 857-866.

[31] Zhao Q, Liew S, Zhang S, Yu Y. Distance-based location management utilizing initial position for mobile communication networks. *IEEE Trans. Mob. Comput.*, 2016, 15(1): 107-120.

Shuo Shang is a professor of computer science at China University of Petroleum, Beijing. He was a research assistant professor at the Department of Computer Science, Aalborg University, Denmark. He received his B.S. degree from Peking University, Beijing, in 2008, and Ph.D. degree from The University of Queensland, Australia, in 2012 respectively, both in computer science. His research interests include efficient query processing in spatio-temporal databases, spatial trajectory computing, and location-based social media. He has served on program committees and as session chairs for several database conferences and as invited reviewer for several database journals, including ICDE, TKDE, The VLDB Journal, ACM TIST, GeoInformatica, KAIS, DKE, and IEICE Transactions.

Lisi Chen is a Ph.D. candidate with DANTE Group at Nanyang Technological University, Singapore. His research interests include geo-textual data management, spatial keyword query evaluation, and location-based social networks.

Zhe-Wei Wei is an associate professor at Renmin University of China. He obtained his Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology in 2012. His research interests include streaming algorithms and data structures.

Dan-Huai Guo is an associate professor of computer science at Computer Network Information Center, Chinese Academy of Sciences, Beijing. His research interests include spatial database, data mining, and machine learning.

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 more than 50 U.S. patents in Web search and related areas. He has published extensively on prestigious international conferences and journals. He is currently the associate editor of ACM Transactions on Information Systems (TOIS). He is a senior member of IEEE.