International Journal of Innovative
Computing, Information and Control
Volume **13**, Number **6**, December **2017**

ICIC International ©2017 ISSN 1349-4198
pp. 1813–1828

DIRECTION-BASED SPATIAL-TEXTUAL SKYLINE

Zijun Chen[1,2], Shasha Guo[1,2] and Wenyuan Liu[1,2]

[1]College of Information Science and Engineering
[2]The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province
Yanshan University
No. 438, West Hebei Ave., Qinhuangdao 066004, P. R. China
{ zjchen; wyliu }@ysu.edu.cn; guoshashaxing@163.com

Received April 2017; revised August 2017

Abstract. *Existing location-based services mostly recommend nearest objects to the user by considering their spatial distance and textual relevance. However, an object not only has a spatial distance to the user, but also has a direction from it to the user. In this paper, we study the direction-based fuzzy preference spatial-textual skyline query. Given a user location, a threshold theta and multiple keywords, skyline and p-skyline objects located at different directions are returned to user. Identifying of the results depends on three aspects: spatial distance, textual similarity and direction. Two functions are introduced to compute the textual relevance and spatial proximity, respectively. A query algorithm is proposed, in which an effective pruning strategy is put forward to improve the query efficiency. In the end, empirical results show that our proposed method is effective and efficient.*

Keywords: Location-based service, Skyline query, Textual relevance, Direction

1. **Introduction.** In recent years, location-based service is widely used [1]. With the development of mobile network, the number of points of interests (POIs) with text information is gradually increasing. These POIs contain both text information and spatial information. For example, in TripAdvisor and Foursquare, people can share experiences of visiting some places.

In general, an object which is spatially close and textually relevant is a good choice for a user [2]. However, in some cases, we will not get a satisfactory result if the spatial distance and textual relevance are only considered. For instance, a user wants to buy stationery before returning home from work and inputs a keyword "notebook". The system returns the user desired results based on the distance between the user's query location and the point of interest containing the keyword "notebook". If the returned result is opposite to the user's home, the user will not be satisfied with the result.

Example 1.1. *Now, we consider an example as shown in Figure 1. A user has just returned from a business trip, but she is wondering whether to go to the barbershop, supermarket or office before going back home. Besides, she would like to have a meal at a restaurant. If she wants to eat noodles, she would express her needs by entering keywords. The recommendation system returns four best restaurants based on spatial distance and textual relevance, restaurant 1 to restaurant 4. Which restaurant is the best one for her? It depends on the direction she will take. If she would like to go to the barbershop, restaurant 1 which is the best, is not a good choice because restaurant 2 is on the way to the barbershop. Therefore, in order to return the best results for users, the query system should present the best restaurant for each direction. In the example, the query result might be restaurant 1 to restaurant 4.*

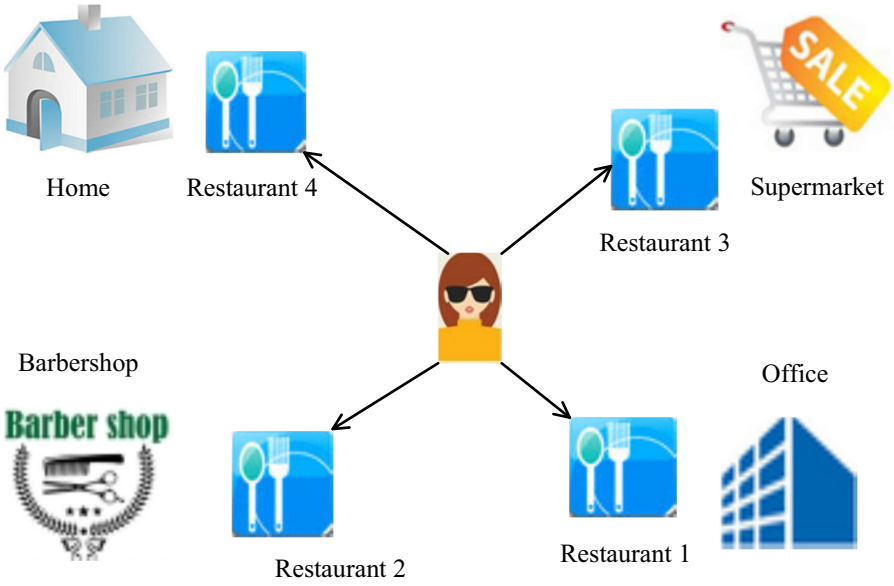


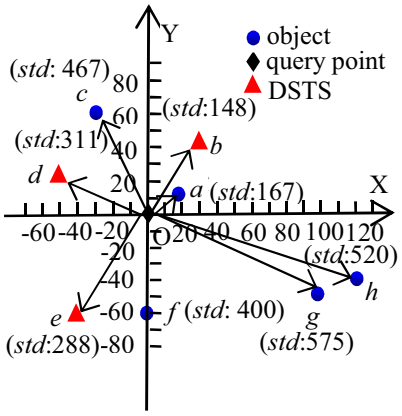
FIGURE 1. Motivating example

In view of this, in this paper, we propose a novel spatial skyline query, namely direction-based spatial-textual skyline query (DSTS query). The query not only compares spatial distance and textual proximity, but also considers direction. And it returns optimal objects according to spatial information and text information from different directions.

Existing methods [3, 4, 5] mostly consider exact matches for query keywords and do not support fuzzy queries. However, because of restrictions of mobile terminal on the size of the screen, the user may enter an error. For example, the user could input keyword "delicaous" in place of "delicious" by mistake, and cannot get satisfactory results. For this situation, approximate string search [6, 7] plays an important role. B. Yao et al. [8] proposed approximate string search in spatial databases, which measured textual relevance between a query point and objects by edit distance. Since fuzzy query algorithm needs to set the threshold in advance, when edit distance is less than or equal to the threshold, we think that related keywords match approximately. Nevertheless, in order to get exact query results, a drawback in this method is that threshold should vary with the length of keywords. J. Hu et al. [9] presented a function to quantify the textual relevance of two objects, but this function only supported single keyword query. Hence, we propose a new function to measure the textual relevance between objects and the query point. There is no need to set a threshold in advance by considering edit distance and the weight of the keyword.

In the following, we will give an example to explain the query. Given a query point q, there is a plane coordinate system originating from q. Then the angle of the object o is defined as the angle swept by x-axis counterclockwise rotation to qo. The angle of object o is denoted by $\beta(q, o)$, and the Euclidean distance between object o and q is denoted by $d(q, o)$.

Example 1.2. *In Figure 2, there are eight objects with the query point q as original point $O = (0, 0)$. We get vectors $\vec{a}, \ldots, \vec{h}$ originating from O. In this example, we set threshold $\theta = \pi/3$. Two vectors are defined in the same direction, if their included angle is smaller than θ. As can be seen from Figure 2, the objects in the same direction with a are b, g and h. Note that b is the object with the smallest spatial-textual distance ($std(q, .)$) that is calculated based on spatial distance and textual proximity. It means that b dominates*



o_i	$d(q, o_i)$	$std(q, o_i)$	$\beta(q, o_i)$
a	20	167	30°
b	52	148	56°
d	56	311	156°
f	60	400	270°
c	70	467	120°
e	72	288	236°
g	115	575	335°
h	130	520	350°

FIGURE 2. An example of DSTS query ($\theta = \pi/3$)

other three objects in the same direction and is a direction-based spatial-textual skyline object (DSTS object). In a similar way, we can know that d and e are both DSTS objects.

We find that there is no skyline object in some angles. In order to return results in more directions, we introduce the concept of p-skyline.

The contributions are summarized as follows.

- We propose a novel query, namely direction-based spatial-textual skyline query. In the meantime, we introduce the concept of p-skyline that is helpful for some practical applications.
- On the basis of the direction-based spatial skyline query, text information is considered, and the fuzzy query is supported.
- We present a query algorithm and an efficient pruning strategy to improve the query speed.
- We conduct experiments on real and synthetic datasets, and results show that our method is effective.

The rest of this paper is organized as follows. In Section 2, we introduce related work. In Section 3, we give the problem statement. Section 4 presents query algorithm. Section 5 reports experimental results and Section 6 summarizes this paper.

2. **Related Work.** S. Börzsönyi et al. [10] first introduced the skyline operator into relational database systems and proposed two main algorithms: Block-Nested-Loops (BNL) and Divide and Conquer (DC). Since then, the skyline operator has attracted extensive research in the database [11, 12, 13].

M. Sharifzadeh and C. Shahabi [14] introduced the concept of spatial skyline query. A spatial skyline set is defined according to multiple derived spatial attributes, each of which is the object's distance to a query point. An object dominates another object if it is better than or equal to another object in their derived spatial attributes, and it is better in at least one derived spatial attribute. For the sake of improving computational efficiency, a pre-computed indexing technique based on Voronoi graph is used. However, the spatial skyline query disregards users' non-spatial preferences. In general, non-spatial preferences which represent users' specific demand are described by textual relevance. Therefore, textually relevant preference query [15] plays an important role. In order to make up for the weakness of the spatial skyline query, J. Shi et al. [16] introduced text into the spatial skyline query and presented textually relevant spatial skylines. They

introduced three models: Derived Dimension Augmentation (DDA), Keyword Boolean Filtering First (KBFF), and Spatio-Textual Dominance (STD), and conducted a detailed analysis of the efficient STD model, and put forward efficient query algorithms.

Above these methods, they consider spatial distance, but do not take account of other spatial attributes. Direction, an important spatial attribute, was first applied into spatial skyline by X. Guo et al. [17]. And they only consider direction and spatial distance, but do not consider textual relevance. Y. Ishikawa et al. [18] extended the work in [17] to the road network and conducted the corresponding study. On the basis of [17], E. El-Dawy et al. [19] applied static attributes (e.g., price, rank) into direction-based skyline. Although it takes account of static attributes, it is different from text information. Therefore, as far as we know, no one has combined the three aspects of spatial distance, text information and direction. In the past, many skyline queries mostly consider exact matches for textual relevance, and few people apply fuzzy matches. In this paper, we allow fuzzy matches, and introduce a new function to quantify the text relevance.

In conclusion, spatial distance, textual relevance and direction are not taken into account at the same time in the above work. Thus, in this paper, we propose direction-based spatial-textual skyline which considers the three aspects.

3. **Problem Statement.** Given a dataset D, an object $o \in D$ is defined as $o =< l, T >$, where $o.l$ is a spatial point with longitude and latitude and $o.T = \{t_1, t_2, \ldots, t_{|o.T|}\}$ is a set of keywords. Each keyword $t \in o.T$ has a weight $w_t(o)$ which represents the proportion of t in the set of keywords. In this paper, we use the TF-IDF model in the field of information retrieval to calculate the weight of a keyword. It can be estimated by Equation (1).

$$w_t(o) = tf(t, o.T) \cdot idf(t, D) \tag{1}$$

In Equation (1), $tf(t, o.T)$ is the frequency of occurrence of t in $o.T$, namely $\frac{count(t,o.T)}{|o.T|}$, $idf(t, D)$ is the logarithmic ratio between the size of D and the number of objects containing the keyword t, and is calculated by $\lg \frac{|D|}{count(t,D)}$.

Example 3.1. *In Figure 3, dataset* $D = \{o_1, o_2, o_3, o_4, o_5, o_6\}$. *We take object* o_1 *as an example, and it contains two keywords* $\{cozy, friendly\}$. *The two keywords' weight can be estimated by Equation (1),* $w_{cozy}(o_1) = \frac{1}{2} \cdot \lg\left(\frac{6}{1}\right) = 0.389$, $w_{friendly}(o_1) = \frac{1}{2} \cdot \lg\left(\frac{6}{2}\right) = 0.239$. *Table 1 represents weights of all keywords in* D.

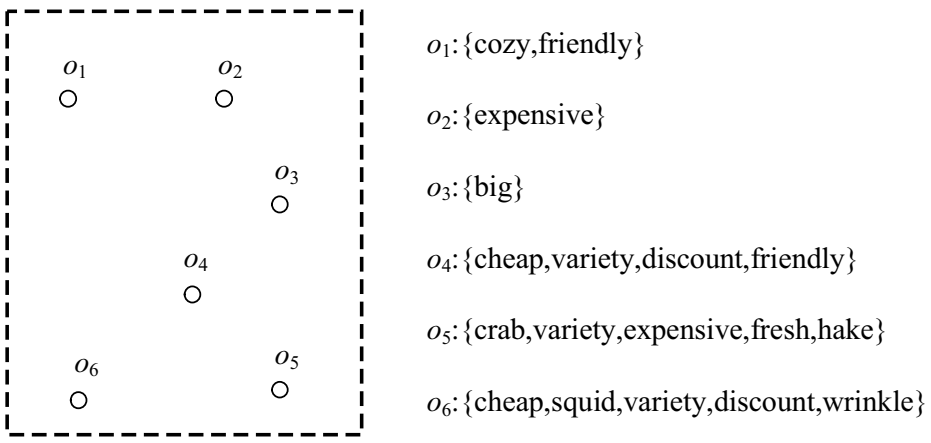


FIGURE 3. An example of TD-IDF model

TABLE 1. The weights of all keywords

	o_1	o_2	o_3	o_4	o_5	o_6
cozy	0.389	0	0	0	0	0
friendly	0.239	0	0	0.119	0	0
expensive	0	0.477	0	0	0.095	0
big	0	0	0.778	0	0	0
cheap	0	0	0	0.119	0	0.095
variety	0	0	0	0.075	0.060	0.060
discount	0	0	0	0.119	0	0.095
crab	0	0	0	0	0.156	0
fresh	0	0	0	0	0.156	0
hake	0	0	0	0	0.156	0
squid	0	0	0	0	0	0.156
wrinkle	0	0	0	0	0	0.156

Definition 3.1. (*Same Direction*) *Given objects o_i and o_j, we say that o_i and o_j are in the same direction from q if the included angle $\lambda_{i,j}$ satisfies $0 \leq \lambda_{i,j} < \theta$ $(0 < \theta \leq \pi/2)$. $\lambda_{i,j}$ is defined as follows:*

$$\lambda_{i,j} = \arccos \frac{\vec{o_i} \cdot \vec{o_j}}{|\vec{o_i}| \cdot |\vec{o_j}|}$$

In order to give the definition of spatial-textual dominance, we first introduce the definition of spatial-textual distance by Equation (2).

$$std(q, o) = \frac{d(q, o)}{Q_T(q, o)} \tag{2}$$

In Equation (2), $Q_T(q, o)$ represents the textual relevance between q and o, and $d(q, o)$ represents the Euclidean distance between q and o. For simplicity, we assume that spatial distances from objects to the query point are all different from each other.

$$Q_T(q, o) = \sum_{i=1}^{n} \frac{w_{t_i'}(o)}{w_{\max}} \times \left(1 - \frac{ed(\psi_i, t_i')}{d_i}\right) \times \frac{1}{n} \tag{3}$$

In Equation (3), the keyword set of object o is $o.T = \{t_1, t_2, \ldots, t_{|o.T|}\}$, the keyword set of query point q is $q.T = \{\psi_1, \psi_2, \ldots, \psi_n\}$, $t_i' = \arg\min_{t \in o.T} ed(\psi_i, t)$, $w_{\max}$ is the global maximum weight, and its purpose is to normalize the weight, $ed(\psi_i, t)$ is the edit distance between t and ψ_i, and d_i represents the length of ψ_i.

Definition 3.2. (Direction-Based Spatial-Textual Dominance) *For the given query point q, if two objects o_i and o_j are in the same direction, and (1) $std(q, o_i) < std(q, o_j)$, or (2) $std(q, o_i) = std(q, o_j)$ and $d(q, o_i) < d(q, o_j)$, we say that o_i dominates o_j based on direction and spatial-textual distance, denoted by $o_i \prec_{DST} o_j$.*

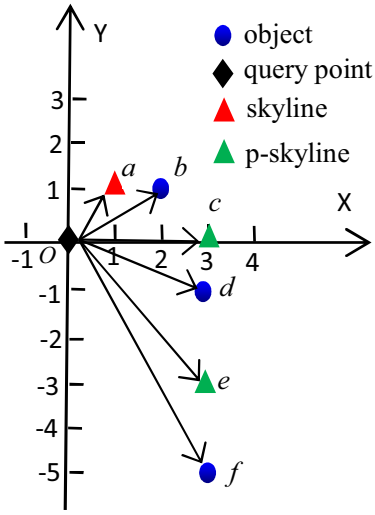
Definition 3.3. (Direction-Based Spatial-Textual Skyline) *Direction-based spatial-textual skyline (skyline) is the set of objects in D which cannot be dominated by any other object in D. That is, o is in the skyline iff we have $\forall o' \in D$, $o' \neq o$, $o' \not\prec_{DST} o$.*

Specially, due to the particularity of spatial-textual distance, there is no skyline object in some direction. In practical application, for the sake of better meeting user's demand, in the following, we give the definition of direction-based spatial-textual pseudo-skyline.

Definition 3.4. (Direction-Based Spatial-Textual Pseudo-Skyline) *Direction-based spatial-textual pseudo-skyline (p-skyline) is the set of objects in D which are not in skyline, and cannot be dominated by objects in skyline or p-skyline.*

Assume that S is a skyline, then PS is a p-skyline iff $\forall o \in PS$, $\exists o' \in D - S - PS$, s.t., $o' \prec_{DST} o$. According to the definition, we can see that $PS \bigcap S = \Phi$.

Example 3.2. *In Figure 4, there are six objects around the query point $O = (0, 0)$. There are six vectors $\vec{a}, \ldots, \vec{f}$ originating from O. In the example, two vectors are in the same direction, if their included angle is smaller than $\theta = \pi/6$. In Figure 4, we know that a is in the same direction with b. Since all objects have the same keywords and a is nearer to the query point than b, b is dominated by a. By the similar method, we can know that c, d, e and f are all dominated. So there is only one object a in skyline. Obviously, the result is not satisfied with the user who would want to obtain satisfactory result in any angle. The definition of p-skyline solves the above problem. In the example, c and e are in p-skyline. Finally, p-skyline objects and skyline objects are returned to the user all together.*



o_i	$d(q, o_i)$	Keywords	$\beta(q, o_i)$
a	1.5	cozy,friendly	45°
b	2.2	cozy,friendly	26.6°
c	3.0	cozy,friendly	0°
d	3.2	cozy,friendly	341.6°
e	4.2	cozy,friendly	315°
f	5.8	cozy,friendly	301°

FIGURE 4. An example of p-skyline ($\theta = \pi/6$)

Definition 3.5. (Direction-Based Spatial-Textual Skyline Query) *A direction-based spatial-textual skyline query (DSTSQ) returns the union of skyline and p-skyline of D.*

4. Query Algorithm.

4.1. **IR-Tree.** In IR-Tree [20], each leaf node contains a list of $(id, location)$, where id refers to the identifier of an object o in dataset D, and $location$ refers to coordinates of o. Each leaf node contains a pointer to an inverted file whose keyword set is the union of all the keywords of objects in the node. For each keyword t, there is a list of $(o, w_t(o))$, where o is an object containing t and $w_t(o)$ is the weight of t.

Each non-leaf node contains numerous entries which are made up of $(cp, rectangle)$ where cp is a pointer to a child node of the node, and rectangle is the minimum bounding rectangle (MBR) of all rectangles containing the child node. Each non-leaf node contains a pointer to an inverted file whose keyword set is the union of all child nodes of the node. For each keyword t, a list of $(e, w_t(e))$ is created, where e is a child node containing t and $w_t(e) = \max_{e' \in CHILD(e)} w_t(e')$.

Figure 5 illustrates an IR-Tree on 9 spatial objects. Table 2 and Table 3 show the contents of the inverted files of non-leaf nodes and leaf nodes, respectively. For example, in Table 2, the weight of the keyword c in child R_3 of non-leaf node R_6 is 0.4, which is the maximal weight of the keyword in the two objects o_3 and o_8 in R_3. In Table 3, the weight of the keyword d in child o_7 of leaf node R_4 is 0.1.

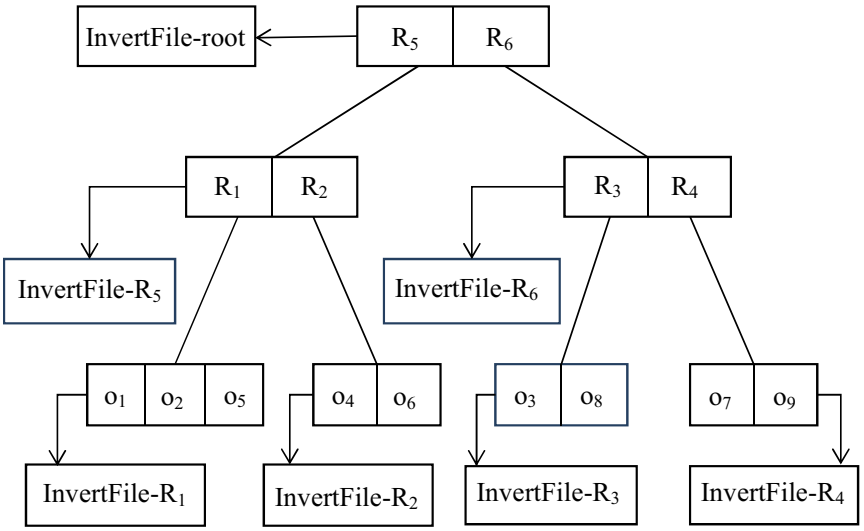


FIGURE 5. IR-Tree

TABLE 2. Content of inverted files of the non-leaf nodes

	$InvertFile-root$	$InvertFile-R_5$	$InvertFile-R_6$
a	$(R_5,0.7)$, $(R_6,0.3)$	$(R_1,0.7)$, $(R_2,0.4)$	$(R_3,0.1)$, $(R_4,0.3)$
b	$(R_5,0.5)$, $(R_6,0.3)$	$(R_1,0.5)$, $(R_2,0.4)$	$(R_3,0.3)$
c	$(R_5,0.5)$, $(R_6,0.7)$	$(R_1,0.5)$, $(R_2,0.4)$	$(R_3,0.4)$, $(R_4,0.7)$
d	$(R_5,0.1)$, $(R_6,0.1)$	$(R_1,0.1)$	$(R_3,0.1)$, $(R_4,0.1)$

TABLE 3. Content of inverted files of the leaf nodes

	$InvertFile-R_1$	$InvertFile-R_2$	$InvertFile-R_3$	$InvertFile-R_4$
a	$(o_1,0.5)$, $(o_2,0.7)$	$(o_4,0.4)$	$(o_3,0.1)$	$(o_9,0.3)$
b	$(o_5,0.5)$	$(o_6,0.4)$	$(o_3,0.1)$, $(o_8,0.3)$	
c	$(o_1,0.5)$, $(o_5,0.5)$	$(o_4,0.4)$, $(o_6,0.3)$	$(o_3,0.4)$, $(o_8,0.3)$	$(o_7,0.7)$, $(o_9,0.3)$
d	$(o_2,0.1)$		$(o_3,0.1)$	$(o_7,0.1)$

Definition 4.1. (MinDist Distance [21]) *In Euclidean space of dimension n, the minimum distance between a point p and MBR $R(s,t)$ is denoted by $MinDist(p,R(s,t))$, which is defined as follows:*

$$MinDist(p,R)=\sum_{i=1}^{n}|p_i-r_i|,\quad r_i=\begin{cases}s_i, & if\ p_i<s_i\\ t_i, & if\ p_i>t_i\\ p_i, & otherwise\end{cases} \tag{4}$$

For the purpose of proposing pruning rules later, some definitions and theorems are given in the following.

Textual Relevance of Node:

$$Q_T(q,e) = \sum_{i=1}^{n} \frac{w_{t'_i}(e)}{w_{\max}} \times \left(1 - \frac{ed(\psi_i, t'_i)}{d_i}\right) \times \frac{1}{n} \tag{5}$$

In Equation (5), $e.T$ is the keyword set of node e, where $e.T = \{t_1, t_2, \ldots, t_{|e.T|}\}$. The keyword set of query point q is denoted by $q.T$, where $q.T = \{\psi_1, \psi_2, \ldots, \psi_n\}$ and $t'_i = \arg\min_{t \in e.T} ed(\psi_i, t)$.

Spatial Distance of Node:

$$d(q,e) = MinDist(q,e) \tag{6}$$

Spatial-Textual Distance of Node:

$$std(q,e) = \frac{d(q,e)}{Q_T(q,e)} \tag{7}$$

Theorem 4.1. *Given a query point q, a node e and any child e' of e, we have $Q_T(q,e') \leq Q_T(q,e)$.*

Proof: In IR-Tree, we know that the weight of the keyword t in e is the maximum weight of t in all children of e. Therefore, we can get $w_t(e') \leq w_t(e)$. For any keyword ψ_i of q, assume that $t_i = \arg\min_{t \in e.T} ed(\psi_i, t)$, and $t'_i = \arg\min_{t \in e'.T} ed(\psi_i, t)$. Since $e.T = \bigcup_{e' \in CHILD(e)} e'.T$, $ed(\psi_i, t_i) \leq ed(\psi_i, t'_i)$. Based on Equation (5), we prove the theorem. □

Theorem 4.2. *Given a query point q, a node e and any child e' of e, we have $std(q,e') \geq std(q,e)$.*

Proof: According to Definition 4.1, we have $MinDist(q,e') \geq MinDist(q,e)$. If e' is an object, we have $MinDist(q,e') = d(q,e')$. Based on Theorem 4.1, we have $Q_T(q,e') \leq Q_T(q,e)$. Then, $std(q,e') = \frac{d(q,e')}{Q_T(q,e')} = \frac{MinDist(q,e')}{Q_T(q,e')} \geq \frac{MinDist(q,e)}{Q_T(q,e')} \geq \frac{MinDist(q,e)}{Q_T(q,e)} = std(q,e)$. So we prove the theorem. □

4.2. **Pruning method.** In IR-Tree query algorithm, with the help of priority queue, we can make the object or node with the smallest $std(q,\cdot)$ to be firstly dequeued. If there is a tie, the object or node with the smaller $d(q,\cdot)$ should be firstly dequeued.

Definition 4.2. (*Angle Range*) *Given a query point q, assuming that set S exists n ($n > 1$) objects. We sort the objects in S by the order of the angle in ascending order to get a sequence $(o_1, o_2, \ldots, o_n)$. Then, the entire plane is divided into n parts by half-line $qo_1, qo_2, \ldots, qo_n$, and we call each part an angle range. $Range(S, o_i)$ is defined as an angle range swept by qo_i ($1 \leq i < n$), q as the center, counterclockwise rotation to the position of qo_{i+1}. Accordingly, the swept angle is called the division angle of $Range(S, o_i)$. If o is in $Range(S, o_i)$, the predecessor and successor of o in S are o_i and o_{i+1}, respectively. $Range(S, o_n)$ is defined as an angle range swept by qo_n, q as the center, counterclockwise rotation to the position of qo_1. Accordingly, the swept angle is called the division angle of $Range(S, o_n)$. If o is in $Range(S, o_n)$, the predecessor and successor of o in S are o_n and o_1, respectively. In S, the predecessor and successor of o are called adjacent object of o and denoted by o_S^- and o_S^+, respectively. The division angle of $Range(S, o_i)$ ($1 \leq i \leq n$) is denoted by $A(S, o_i)$.*

In the following, we will propose a pruning rule in Theorem 4.3.

Theorem 4.3. *In the process of DSTS query, we will get the result set incrementally. The current result set and the final result set of DSTS query are denoted by R and R^F, respectively. Given an object or node p that will be inserted into the priority queue. Assume that p is in an angle range $Range(R, o_i)$ and $A(R, o_i) < 2\theta$, then p need not be inserted in the priority queue, and this will not affect the correctness of R^F.*

Proof: Since p is in $Range(R, o_i)$ and $A(R, o_i) < 2\theta$, we can get $\exists o \in R$, s.t., $o \prec_{DST} p$ (p is an object) or $\forall o' \in p.MBR$, $\exists o \in R$, s.t., $o \prec_{DST} o'$ (p is a node). So, p will not be in R^F. Next, we should prove that we can get R^F without the help of p.

(1) p is an object.

For an object o, if $p \not\prec_{DST} o$, we can determine whether o is in R^F without the help of p.

If an object o is dequeued from the priority queue before p's parent node, we can get $p \not\prec_{DST} o$ apparently. So we should consider the object o dequeued from the priority queue after p's parent node. If $std(q, o) \leq std(q, p)$, then we have $p \not\prec_{DST} o$. So we should only consider the object o which satisfies $std(q, o) > std(q, p)$.

According to the relationship of o and $Range(R, o_i)$, there are two cases.

Case 1 (o is in $Range(R, o_i)$): In case 1, we can get $\exists o' \in R$, s.t., $o' \prec_{DST} o$ and therefore o is not in R^F. So we can determine whether o is in R^F without the help of p. For instance, as shown in Figure 6, $R = \{a, b, c, d\}$. Assume that $A(R, a) < 2\theta$, p and o are both in $Range(R, a)$, and $std(q, o) > std(q, p)$. Then, we get $a \prec_{DST} o$ or $b \prec_{DST} o$.

Case 2 (o is out of $Range(R, o_i)$): If o is in the same direction with p, then we will have $p_R^- \prec_{DST} o$ or $p_R^+ \prec_{DST} o$. We get that o is not in R^F and therefore pruning p completely has no impact on o. If o is not in the same direction with p, then $p \not\prec_{DST} o$. So pruning p completely has no influence on o. For example, as shown in Figure 7, $R = \{a, b, c, d\}$. Assume that $A(R, a) < 2\theta$, p is in $Range(R, a)$, o is out of $Range(R, a)$, and $std(q, o) > std(q, p)$. If o is in the same direction with p, according to Figure 7, we know that $a \prec_{DST} o$.

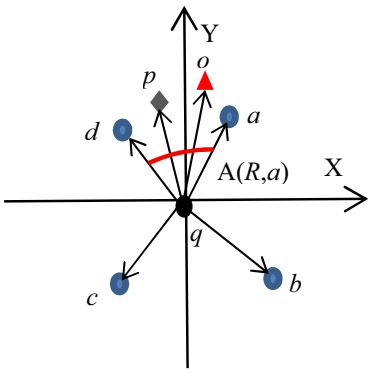


FIGURE 6. The first case

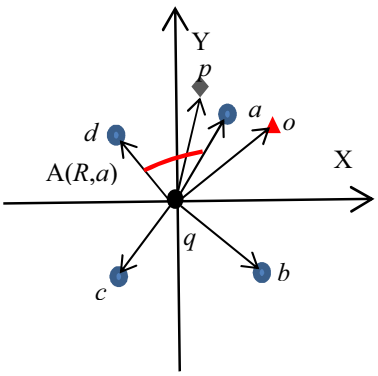


FIGURE 7. The second case

(2) p is a node.

Since $Range(R, o_i)$ contains p, it contains all objects in $p.MBR$. Therefore, we can prove that pruning p will not affect the correctness of R^F in a way which is similar to that of (1). So we prove the theorem. □

4.3. **Algorithm description.** Algorithm 1 describes details of query processing. Algorithm 2 illustrates how to calculate the number of division angles which are not smaller than 2θ. Algorithm 3 represents how to identify the type of o.

Theorem 4.4. *Given the current result set R, while all division angles obtained by R are smaller than 2θ, we can terminate the query algorithm.*

Proof: After finding the current result set R, we get any object o dequeued from the priority queue. While all division angles obtained by R are smaller than 2θ, there must be an object o' in R which is in the same direction with o. Apparently, o is dequeued from the priority queue after o'. According to the creation of the priority queue, we know that $o' \prec_{DST} o$. Therefore, R is the final result set, and the query algorithm could be terminated correctly. □

Theorem 4.5. *When an object o is dequeued from the priority queue, the type of o could be identified by R and L, where R is the current result set and L is the current set of visited objects.*

Proof: According to the relationship of o and the adjacent objects of o in L, there are four cases in the following.

Case 1: While o is not in the same direction with o_L^- and o_L^+, o is the object in skyline (see Figure 8(a)).

Case 2: While o is not in the same direction with o_L^- and is in the same direction with o_L^+, we can know that $o_L^+ \prec_{DST} o$. If $o_L^+ \in R$, o is not a result; if $o_L^+ \notin R$ and o is not in the same direction with the adjacent objects of o in R, o is an object in p-skyline; otherwise, o is not a result (see Figure 8(b)).

Case 3: While o is in the same direction with o_L^- and is not in the same direction with o_L^+, it is similar to that of Case 2 to identify the type of o (see Figure 8(c)).

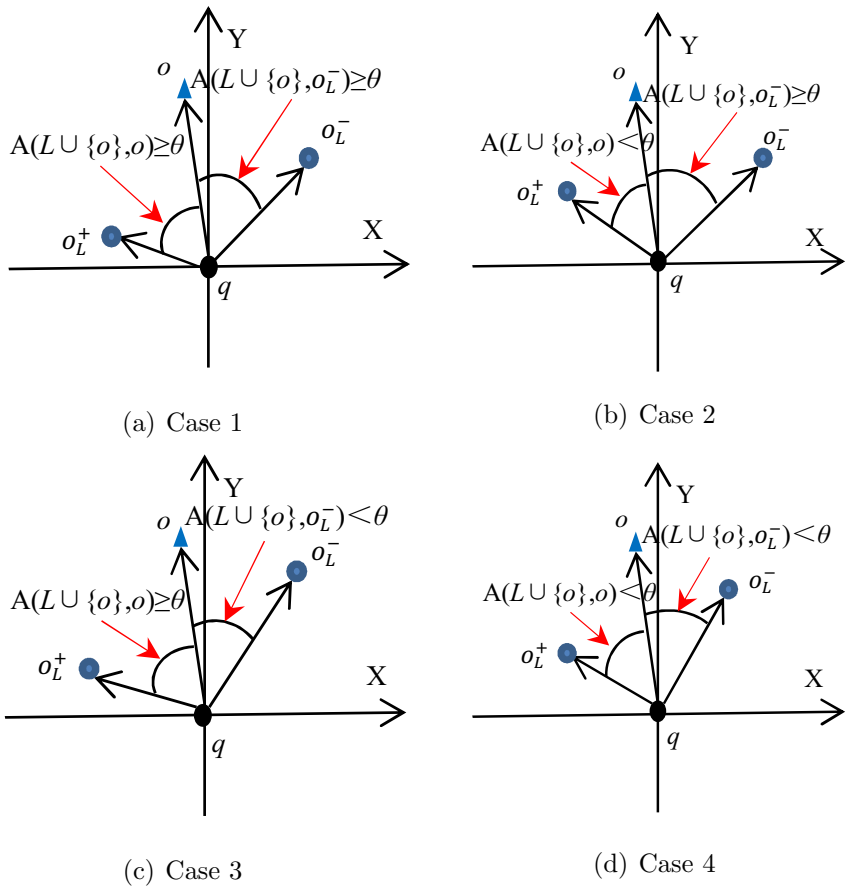


FIGURE 8. Four cases of indetifying object type

Case 4: While o is in the same direction with o_L^- and o_L^+, we can know that $o_L^- \prec_{DST} o$ and $o_L^+ \prec_{DST} o$. If $o_L^- \in R$ or $o_L^+ \in R$, o is not a result; if $o_L^- \notin R$, $o_L^+ \notin R$, and o is not in the same direction with the adjacent object of o in R, we can know that o is an object in p-skyline; otherwise, o is not a result (see Figure 8(d)).

So we prove the theorem. □

For Algorithm 1, in line 5, if all division angles obtained by R are smaller than 2θ, then $count = 0$, and we can terminate the query algorithm according to Theorem 4.4. That is, *count* is the number of division angles which are not smaller than 2θ. In lines 7-20, if e is an object, its type will be identified in the following steps. In lines 9-12, e is the first object dequeued from the priority queue, so it must be in skyline. ComputeCount() is invoked to update the value of *count*, and e is inserted into R and L, respectively. In lines 13-20, e is not the first object dequeued from the priority queue. ComputeType() is invoked to identify the type of e. In lines 21-26, if e is a node, the child of e could be pruned according to Theorem 4.3. If e is a non-leaf node, then p is a node; otherwise p is an object.

Algorithm 1 Query Algorithm (DSTSQ)

Input: query point q, threshold θ, IR-Tree *index*
Output: result set R

```
1: R = Φ, count = −1, L = Φ //count refers to a counter. L is a set of visited objects
2: Queue ← NewPriorityQueue()
3: Queue.Enqueue(index.RootNode, 0, 0) //There are two keys in the queue, which are std(q, .)
   and d(q, .), respectively
4: while not Queue.IsEmpty() do
5:     if count == 0 then
6:         return R
7:     e = Queue.Dequeue()
8:     if e refers to an object then
9:         if R == Φ then
10:            count=ComputeCount(R, e, count) //call Algorithm2
11:            R ← e
12:            L ← e
13:        else
14:            e.type=ComputeType(L, e, R)        //call Algorithm3
15:            L ← e
16:            if (e.type==1 or e.type==2) then //e is in skyline or p-skyline
17:                count=ComputeCount(R, e, count)
18:                R ← e
19:            else
20:                continue
21:    else   //e refers to a node
22:        for each child p of node e do
23:            if (p can be pruned according to Theorem 4.3) then
24:                continue
25:            else
26:                Queue.Enqueue(p, std(q, p), d(q, p))
27: return R
```

For Algorithm 2, it will be invoked when an object o is identified as a result, and will be added to R. Since o is a result, we know that $A(R, o_R^-) \geq 2\theta$. When R is empty, count is not used to record the number of division angles which are not smaller than 2θ. So in the beginning, we let $count = -1$. In lines 1-2, we can know that R is empty according

Algorithm 2 ComputeCount Algorithm

Input: current result set R, object o, counter *count*
Output: *count*

```
1: if (count == −1) then
2:     return 1
3: else   find the adjacent object (o_R^− and o_R^+) in R
4:     if (A(R ∪ {o}, o_R^−) ≥ 2θ ∧ A(R ∪ {o}, o) ≥ 2θ) then
5:         return ++count
6:     else
7:         if (A(R ∪ {o}, o_R^−) < 2θ ∧ A(R ∪ {o}, o) < 2θ) then
8:             return −−count
9:         else
10:            return count
```

Algorithm 3 ComputeType Algorithm

Input: set of visited objects L, object o, current result set R
Output: *type* of o

```
1: < o_L^−, o_L^+ > = L.getAdjacentObject(o)
2: < o_R^−, o_R^+ > = R.getAdjacentObject(o)
3: if (A(L ∪ {o}, o_L^−) ≥ θ ∧ A(L ∪ {o}, o) ≥ θ) then    //case 1
4:     return 1 //o is in skyline
5: if (A(L ∪ {o}, o_L^−) ≥ θ ∧ A(L ∪ {o}, o) < θ) then    //case 2
6:     if (o_L^+ ∈ R) then
7:         return 3 //o is not in result set
8:     else
9:         if (A(R ∪ {o}, o_R^−) ≥ θ ∧ A(R ∪ {o}, o) ≥ θ) then
10:            return 2 //o is in p-skyline
11:        else
12:            return 3
13: if (A(L ∪ {o}, o_L^−) < θ ∧ A(L ∪ {o}, o) ≥ θ) then    //case 3
14:    if o_L^− ∈ R then
15:        return 3
16:    else
17:        if (A(R ∪ {o}, o_R^−) ≥ θ ∧ A(R ∪ {o}, o) ≥ θ) then
18:            return 2
19:        else
20:            return 3
21: if (A(L ∪ {o}, o_L^−) < θ ∧ A(L ∪ {o}, o) < θ) then    //case 4
22:    if (o_L^− ∈ R or o_L^+ ∈ R) then
23:        return 3
24:    else
25:        if (A(R ∪ {o}, o_R^−) ≥ θ ∧ A(R ∪ {o}, o) ≥ θ) then
26:            return 2
27:        else
28:            return 3
```

to the value of *count*. We let *count* = 1, since o will be the first object added to R. In lines 4-5, if a division angle larger than or equal to 2θ is divided into two angles larger than or equal to 2θ, *count* is increased by 1. In lines 7-8, if a division angle larger than or equal to 2θ is divided into two angles less than 2θ, *count* is decreased by 1.

In Algorithm 3, we can identify the type of an object according to Theorem 4.5.

5. **Experimental Study.** All algorithms were implemented in Java, and the experiments were conducted on a 3.60GHz six-core machine running Windows 10 operating system with 8GB memory.

5.1. **Datasets.** In experiments, we use a real dataset and a synthetic dataset to study the performance of our proposals. The real dataset is TG which is obtained by https://www.cs.utah.edu/ lifeifei/, and the synthetic dataset is generated by random generator. We randomly assign 5-10 keywords to each object in the two datasets. The real dataset contains 18,200 objects, and the synthetic dataset contains 70,900 objects. We measure the average runtime and number of dominance tests over generated workloads of 100 queries for each parameter setting. The query location is randomly generated, and the keywords are randomly selected from the keywords of objects.

5.2. **Experimental results.** For simplicity, we denote the real dataset as R and the synthetic dataset as S. In order to test the effect of pruning strategy, we update the algorithm DSTSQ by removing lines 23-25 in Algorithm 1, denoted by DSTSQ′. We compare DSTSQ with DSTSQ′ in R and S, respectively.

(1) Effect of θ on runtime

Figures 10(a) and 10(b) show the effect of θ on runtime when the number of keywords for query point q is 1-3. We observe that as θ increases, the runtime decreases in general. The reason is that as θ increases, an object can dominate more objects and the termination condition of Theorem 4.4 is easy to meet.

(2) Effect of θ on dominance test

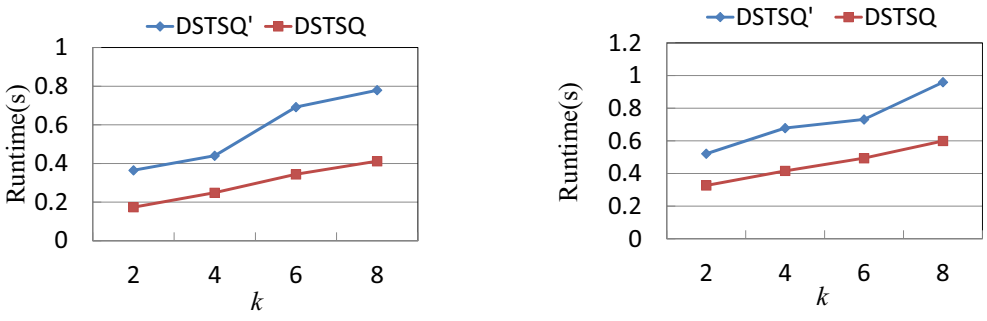
Figures 10(c) and 10(d) describe the effect of θ on dominance test when the number of keywords for query point q is 1-3. With θ increasing, the dominance test decreases in general. The reason is as follows. As θ increases, the size of result set decreases. Accordingly, the number of objects to be compared will be reduced, namely dominance test decreases.

(3) Effect of the number of keywords k on runtime

Figures 9(a) and 9(b) represent the effect of the number of keywords k on runtime when $\theta = \pi/6$. We find that as k increases, the runtime becomes longer. The reason is as follows. For each keyword in $q.T$, the list created for it in the inverted file should be visited. With k increasing, more lists in the inverted file should be visited, and the runtime of traversing inverted file becomes longer.

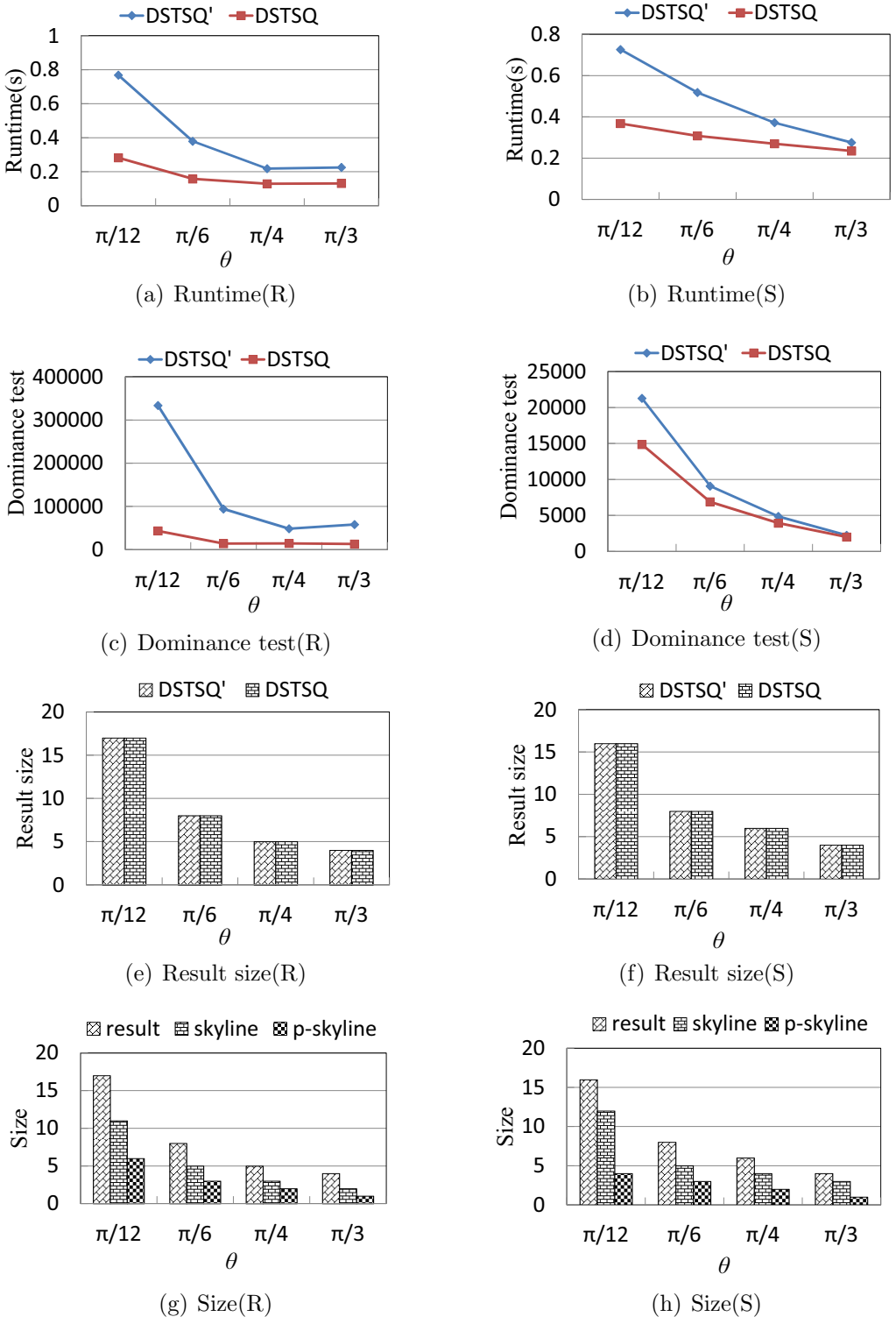
(4) Effect of θ on the result set

Figures 10(e) and 10(f) show the effect of θ on the result set when the number of keywords for query point q is 1-3. With the increase of θ, the size of the result set



(a) Runtime(R) (b) Runtime(S)

FIGURE 9. Effect of number of keywords k



(a) Runtime(R) (b) Runtime(S)

(c) Dominance test(R) (d) Dominance test(S)

(e) Result size(R) (f) Result size(S)

(g) Size(R) (h) Size(S)

FIGURE 10. Effect of threshold θ

decreases. Given a final result set R, the size of R equals the number of division angles obtained by R. According to Definition 3.5, the division angles obtained by R will not be less than θ, so the size of R will not be larger than $2\pi/\theta$. Therefore, with θ increasing, the size of result set might become smaller. Meanwhile, from the experiment, we know that for the same θ, the results of DSTSQ and DSTSQ′ are all the same, which verifies the correctness of the pruning rule in Theorem 4.3.

In Figures 10(a)-10(f) and Figure 9, we find that the runtime of DSTSQ is less than that of DSTSQ′. At the same time, the dominance test of DSTSQ is less than that of DSTSQ′. In addition, the number of objects in S is more than that in R. Thus, the runtime in S is longer than that in R.

(5) Effect of θ on the size of skyline and p-skyline

Figures 10(g) and 10(h) describe the effect of θ on the size of skyline and p-skyline. We can see that the p-skyline is not empty under different value of θ. Since the size of result set will not be larger than $2\pi/\theta$, if the size of result set is less than $2\pi/\theta$, it shows that the user would not get useful result in some direction. However, if the objects in p-skyline are not returned, the user will not get satisfactory result in more directions. Therefore, we should return the objects in p-skyline to better meet the user's demand.

6. **Conclusions.** In this paper, we propose direction-based spatial-textual skyline query. On the basis of direction, we take account of spatial distance and textual relevance, and apply fuzzy query to the calculation of textual relevance. Since there is no skyline object in some direction, we propose the concept of p-skyline to better meet user's demand. The query is carried out to find skyline and p-skyline objects, and the union of skyline and p-skyline will be the result set of the query. We present an effective pruning strategy in the process of query. Objects or nodes that satisfy the pruning rule will not be inserted into the priority queue, such that there will be less objects to be visited. We have proved that while all division angles obtained by the result set are smaller than 2θ, the query algorithm can be terminated. Finally, extensive experimental evaluation on both real and synthetic datasets shows that the proposed algorithm is efficient and effective.

In future work, we intend to develop algorithms to support the continuous query which retrieves objects on the direction-based spatial-textual skyline. In addition, it is of interest to examine the processing of the skyline query over road networks.

REFERENCES

[1] A. Ponomarev, Recommending tourist locations based on data from photo sharing service: Method and algorithm, *Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology*, pp.272-278, 2016.

[2] I. D. Felipe, V. Hristidis and N. Rishe, Keyword search on spatial databases, *Proc. of the 24th International Conference on Data Engineering*, Cancún, México, pp.656-665, 2008.

[3] C. Zhang, Y. Zhang, W. Zhang et al., Inverted linear quadtree: Efficient top k spatial keyword search, *IEEE Trans. Knowledge and Data Engineering*, vol.28, no.7, pp.901-912, 2013.

[4] L. Chen, G. Cong and X. Cao, An efficient query indexing mechanism for filtering geo-textual data, *Proc. of the 2013 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, pp.749-760, 2013.

[5] D. Zhang, Y. M. Chee, A. Mondal et al., Keyword search in spatial databases: Towards searching by document, *Proc. of the 25th International Conference on Data Engineering*, Shanghai, China, pp.688-699, 2009.

[6] C. Li, J. Lu and Y. Lu, Efficient merging and filtering algorithms for approximate string searches, *Proc. of the 24th International Conference on Data Engineering*, Cancún, México, pp.257-266, 2008.

[7] K. Chakrabarti, S. Chaudhuri, V. Ganti et al., An efficient filter for approximate membership checking, *Proc. of the ACM SIGMOD International Conference on Management of Data*, Vancouver, BC, Canada, pp.805-818, 2008.

[8] B. Yao, F. Li, M. Hadjieleftheriou et al., Approximate string search in spatial databases, *Proc. of the 26th International Conference on Data Engineering*, Long Beach, CA, USA, pp.545-556, 2010.

[9] J. Hu, J. Fan and G. Li, Top-k fuzzy spatial keyword search, *Chinese Journal of Computers*, vol.35, no.11, pp.2237-3346, 2012 (in Chinese).

[10] S. Börzsönyi, D. Kossmann and K. Stocker, The skyline operator, *Proc. of the 17th International Conference on Data Engineering*, Heidelberg, Germany, pp.421-430, 2001.

[11] K. L. Tan, P. K. Eng and B. C. Ooi, Efficient progressive skyline computation, *Proc. of the 27th International Conference on Very Large Data Bases*, Roma, Italy, pp.301-310, 2001.
[12] J. Chomicki, P. Godfrey, J. Gryz et al., Skyline with presorting, *Proc. of the 19th International Conference on Data Engineering*, Bangalore, India, pp.717-719, 2003.
[13] D. Papadias, Y. Tao, G. Fu et al., Progressive skyline computation in database systems, *ACM Trans. Database Systems*, vol.30, no.1, pp.41-82, 2005.
[14] M. Sharifzadeh and C. Shahabi, The spatial skyline queries, *Proc. of the 32nd International Conference on Very Large Data Bases*, Seoul, Korea, pp.751-762, 2006.
[15] X. Kuang, P. Zhao, V. S. Sheng et al., TK-SK: Textual-restricted k spatial keyword query on road networks, *Databases Theory and Applications – The 26th Australasian Database Conference*, Melbourne, VIC, Australia, pp.167-179, 2015.
[16] J. Shi, D. Wu and N. Mamoulis, Textually relevant spatial skylines, *IEEE Trans. Knowledge and Data Engineering*, vol.28, no.1, pp.224-237, 2016.
[17] X. Guo, Y. Ishikawa and Y. Gao, Direction-based spatial skylines, *Proc. of the 9th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, Indianapolis, IN, USA, pp.73-80, 2010.
[18] X. Guo, B. Zheng, Y. Ishikawa et al., Direction-based surrounder queries for mobile recommendations, *The VLDB Journal*, vol.20, no.5, pp.743-766, 2011.
[19] E. El-Dawy, H. M. O. Mokhtar and A. El-Bastawissy, Directional skyline queries, *Proc. of the 3rd International Conference on Data and Knowledge Engineering*, Wuyishan, China, pp.15-28, 2012.
[20] G. Cong, C. S. Jensen and D. Wu, Efficient retrieval of the top-k most relevant spatial web objects, *Proc. of the VLDB Endowment*, vol.2, no.1, pp.337-348, 2009.
[21] N. Roussopoulos, S. Kelley and F. Vincent, Nearest neighbor queries, *Proc. of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, CA, USA, pp.71-79, 1995.