

# Retrieving Text-based Surrounding Objects in Spatial Databases

Bojie Shen, Md. Saiful Islam, David Taniar and Junhu Wang

**Abstract** Retrieval of textually relevant non-dominated surrounding data objects has many potential applications in spatial databases such as textually relevant nearby point-of-interest retrieval surrounding a user. This paper presents a novel query, called textually-relevant direction-based spatial skyline (TDSS), for retrieving textually relevant non-dominated surrounding data objects in spatial databases. The paper also presents efficient algorithms for processing TDSS queries in spatial databases by designing novel data pruning techniques using keyword inverted index and R-Tree data indexing scheme. The effectiveness and efficiency of the proposed algorithms are demonstrated by conducting extensive experiments.

## 1 Introduction

The retrieval of textually relevant non-dominated surrounding data objects has potential applications in spatial databases. For example, consider a user who is looking for nearby restaurants surrounding her (green point as shown in Fig. 1) that provide “Shushi”. A textually-relevant direction-based spatial skyline (TDSS) query can return a number of restaurants surrounding her by trading off the direction and distance  $\{r_2, r_3, r_4, r_5, r_6\}$  as shown in Fig.1 as well as matching her preferred food item. Though  $\{r_1, r_8\}$  and  $r_7$  also provide “Shushi”, they are in the same directions

---

Bojie Shen  
Monash University, Melbourne, Australia e-mail: bshe21@student.monash.edu

Md. Saiful Islam  
Griffith University, Griffith University, Gold Coast, Australia e-mail: saiful.islam@griffith.edu.au

David Taniar  
Monash University, Melbourne, Australia e-mail: david.taniar@monash.edu

Junhu Wang  
Griffith University, Griffith University, Gold Coast, Australia e-mail: j.wang@griffith.edu.au

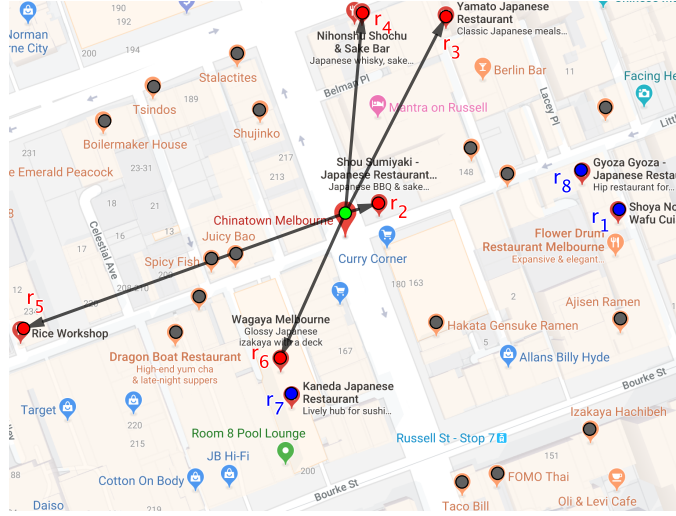


Fig. 1: An application of TDSS queries for retrieving surrounding objects (results produced by our query model)

as  $r_2$  and  $r_6$ , respectively and are also far from the user in comparison to  $r_2$  and  $r_6$ , respectively. There exists plenty of works on the retrieval of textually relevant objects in spatial databases ([2, 3, 6, 20, 22, 25] for survey). Unfortunately, none of these works incorporates surroundingness in the retrieval of textually relevant data objects from spatial databases.

The first work on direction based spatial skyline query (DSQ) for retrieving surrounding data objects in spatial databases is proposed by Guo et al. [7, 8]. The DSQ query retrieves all data objects that are closest to the given query point and that are not dominated by other data objects in their directions w.r.t. query point. The authors propose that a data object  $p_i$  should dominate another data object  $p_j$  w.r.t. the query object  $q$  if (i) both  $p_i$  and  $p_j$  are in the same direction according to the user-given acceptance angle, i.e.,  $\angle p_i q p_j \leq \tau$ , where  $\tau$  is the user-given acceptance angle and (ii)  $p_i$  is closer to  $q$  than  $p_j$ , i.e.,  $d(q, p_i) < d(q, p_j)$ , where  $d(q, p_i)$  denotes the Euclidean distance between  $q$  and  $p_i$ . This query model does not consider to emphasize textual relevance on surrounding objects retrieval. The DSQ query also has two major problems as given as follows: (i) missing result and (ii) instability. The missing result problem is caused by the fact that a non-resultant data object can dominate and filter other data objects. The instability problem is caused by the settings of the user given acceptance angle. A small change in the acceptance angle can provide a completely different result set and causes the instability issue.

Another work on surrounding objects retrieval is the *nearest surrounder queries* (NSQ) proposed by Lee et al. [13, 14]. The authors propose an approach to retrieve nearest surrounder objects of arbitrary shapes w.r.t. the given query point and argue that surrounder objects should be visible from the query point. Unfortunately, none of these works [7, 8, 13, 14] considers textual relevance in their approaches.

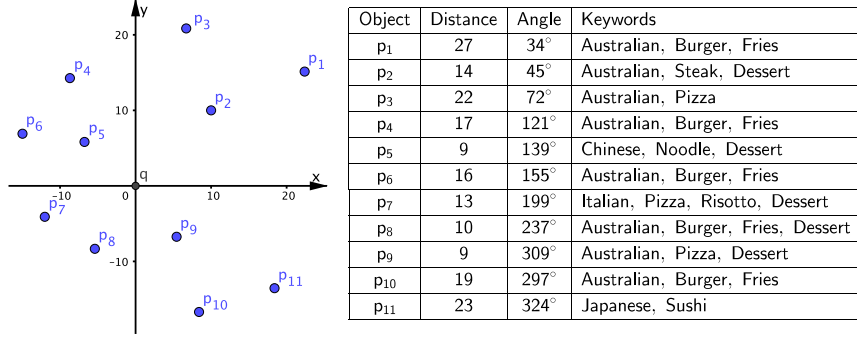


Fig. 2: A toy dataset for exemplifying the definitions and algorithms in this paper

To fill the research gap and alleviate the missing result and instability problems of DSQ queries [7, 8], we propose a novel query called, Textually-relevant Direction-based Spatial Skyline (TDSS), for retrieving textually relevant non-dominating surrounding data objects in spatial databases. We propose directional zone (DZ) to measure directional similarities among spatial data objects. We also develop novel data pruning techniques based on keyword inverted list and R-tree data indexing and propose two different algorithms to process TDSS queries in spatial databases. To be specific, our main contributions are summarized below:

1. we propose directional zone to measure directional similarity between two spatial data objects (Section 2);
2. we present a novel query called TDSS to retrieve textually relevant non-dominating surrounding point data objects (Section 2);
3. we propose efficient algorithms to process TDSS queries in spatial databases by designing novel pruning techniques based on keyword inverted list and R-tree data indexing scheme (Section 3); and
4. we experimentally evaluate our proposed algorithms (Section 4).

## 2 Preliminaries

**Data Model.** We assume that  $P$  is a set of spatial data objects and an individual data object  $p \in P$  is modeled as a point in  $xy$  plane. The  $x$  and  $y$  coordinates of  $p \in P$  are denoted by  $p^x$  and  $p^y$ , respectively. The query object is also a point in  $xy$  plane and is denoted by  $q$ . Each object  $p \in P$  is associated with a set of keywords and is denoted by  $p.\psi$ . The query keywords set is denoted by  $q.\psi$ . Data objects and points are used interchangeably in this paper. We use the toy dataset given in Fig. 2 to exemplify the definitions and algorithms provided the paper.

**Definition 1.** A data object  $p$  is said to be a keyword-match object iff  $p.\psi = q.\psi$ .

**Definition 2.** A keyword match data object  $p_i$  dominates another keyword-match data object  $p_j$  w.r.t. a given query object  $q$ , denoted by  $p_i \prec p_j$ , iff the following

holds: (a)  $p_j$  is directionally similar to  $p_i$  w.r.t.  $q$ ; and (b)  $d(q, p_i) < d(q, p_j)$ , where  $d(q, p_i)$  denotes the distance between  $q$  and  $p_i$ .

From Definition 2, it is obvious that we need to establish (i) *directional similarity metric* and (ii) *distance metric* to decide on the dominance between two spatial data objects. For the *distance metric*, we rely on the Euclidean distance measure. For *directional similarity metric*, in this paper we propose *directional zone* to model the directional similarity among the spatial data objects. Firstly, the direction of an arbitrary point object  $p_i$  w.r.t. the query  $q$  is modeled by  $\vec{qp_i}$ . Assume that the intersection point of the perpendicular line from point  $p_j$  to  $\vec{qp_i}$  is denoted by  $I(p_j, \vec{qp_i})$ . The distance  $d(p_j, I)$  can be used to measure how far  $p_j$  deviates from the direction of  $p_i$  w.r.t.  $q$ . On the other hand,  $d(p_i, I)$  can be used to measure how far  $p_j$  is away from  $p_i$  according to the direction  $\vec{qp_i}$ . We consider  $p_j$  has the same direction as  $p_i$  w.r.t.  $q$  if  $d(p_i, I) > d(p_j, I)$ . To model the above, we rotate the ray  $\vec{qp_i}$  by  $45^\circ$  (up to this limit we get  $d(p_i, I) > d(p_j, I)$ ) both in clockwise and anticlockwise considering  $p_i$  as the center of origin. Assume that these rays are  $\vec{p_i\hat{c}}$  and  $\vec{p_i\hat{a}}$ , respectively. Now, the directional zone of  $p_i$ , denoted by  $DZ(p_i)$ , is formed by the area bounded by  $\vec{p_i\hat{a}}$  and  $\vec{p_i\hat{c}}$  as illustrated in Fig. 3 for data object  $p_2$ .

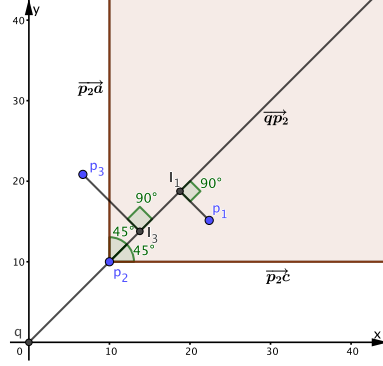


Fig. 3: The directional zone of  $p_2$

**Definition 3.** A data object  $p_j$  is considered to be directionally similar to a data object  $p_i$  w.r.t. the given query object  $q$  in spatial data space if  $p_j \in DZ(p_i)$ .

**Lemma 1.** If  $p_j \in DZ(p_i)$ , then we get  $d(q, p_i) < d(q, p_j)$ .

**Lemma 2.** If  $p_j \in DZ(p_i)$ , then we get  $p_i \prec p_j$ .

**Definition 4.** Given a set of spatial data objects  $P$  and a query point  $q$ , a textually-relevant direction-based spatial skyline (TDSS) query for  $q$ , denoted by  $TDSS(q)$ , retrieves all keyword match data object  $p_i \in P$  if any of the following holds:

1.  $\nexists p_j \in P$  such that  $p_j \prec p_i$ , where  $p_j$  is a keyword match data object; and
2.  $\nexists p_k \in TDSS(q)$  and  $p_j \in P$  such that  $p_k \prec p_j$  and  $p_j \prec p_i$  but  $p_k \not\prec p_i$ , where both  $p_j$  and  $p_k$  are keyword match data objects.

The above definition of TDSS queries for retrieving the surrounding data objects closely matches the idea of global skyline [5] (i.e., union of the skylines in each quadrant of the query point in 2D) which is rotationally invariant in spatial context. Here, we also emphasize textual relevance through Definition 1, i.e., a TDSS query retrieves only textually relevant non-dominated surrounding data objects.

### 3 Our Approach

This section presents our approach of processing TDSS queries in spatial databases.

#### 3.1 Dominance Checking

Here, we explain our idea of checking the directional dominance between a pair of spatial data points  $p_i$  and  $p_j$ . Firstly, we identify the perpendicular intersection point  $I$  from the point  $p_j$  to the ray  $\overrightarrow{qp_i}$  and derive the following vector based calculation to calculate the coordinates of the perpendicular intersection point  $I$ .

$$I^x = q^x + t(p_i^x - q^x) \quad (1)$$

$$I^y = q^y + t(p_i^y - q^y) \quad (2)$$

$$0 = (I_x - p_j^x)(p_i^x - q^x) + (I^y - p_j^y)(p_i^y - q^y) \quad (3)$$

By substituting the first two equations into the third, we get the following:

$$t = \frac{(p_j^x - q^x)(p_i^x - q^x) + (p_j^y - q^y)(p_i^y - q^y)}{(p_i^x - q^x)^2 + (p_i^y - q^y)^2} \quad (4)$$

Now, the directional dominance between  $p_i$  and  $p_j$  can be decided by comparing the distance  $d(p_i, I)$  and  $d(p_j, I)$ . However, we need to ensure that the intersection point  $I$  follows  $p_i$  on the ray  $\overrightarrow{qp_i}$ , so that the  $p_j$  can be directionally dominated by  $p_i$ . For the above, we derive the following property:  $d(q, I) > d(q, p_i)$  and  $d(q, I) > d(p_i, I)$  from observation. Finally, for  $p_j$  to appear inside  $DZ(p_i)$  we check the following condition:  $d(p_i, I) \geq d(p_j, I)$ . Otherwise, the point  $p_j$  deviates from the direction  $\overrightarrow{qp_i}$  and it must not be inside the  $DZ(p_i)$ . Based on the above formulation, we can decide that  $p_2 \prec p_1$  and  $p_2 \not\prec p_3$  as illustrated in Fig. 3.

#### 3.2 TDSS Query Processing

This section proposes two algorithms for processing the TDSS queries in spatial databases. Our first algorithm is called Keyword Filtering Based Approach (KFBA). KFBA algorithm utilizes the keyword inverted index data structure to filter out the non-keyword match objects based on the query keywords. The second approach is a Branch and Bound Keyword Matching (BBKM) approach which progressively matches query keywords after indexing the database objects into an R-tree. BBKM also exploits the directional dominance between the current skyline points and the bounding boxes in R-Tree data indexing to expedite the query processing.

##### 3.2.1 Keyword Filtering Based Approach

The main idea of keyword filtering based approach (KFBA) is to take advantage of the pruning power of the keyword inverted index, which is a mapping between the database objects and the keywords. The keyword inverted index allows us to quickly search for the objects that contain a specific keyword and can help us to

Table 1: The keyword inverted index of the toy dataset given in Fig. 2

Keyword	$p_1$	$p_2$	$p_3$	$p_4$	$p_5$	$p_6$	$p_7$	$p_8$	$p_9$	$p_{10}$	$p_{11}$
Australian	1	1	1	1	0	1	0	1	1	1	0
Chinese	0	0	0	0	1	0	0	0	0	0	0
Italian	0	0	0	0	0	0	1	0	0	0	0
Japanese	0	0	0	0	0	0	0	0	0	0	1
Burger	1	0	0	1	0	1	0	1	0	1	0
Fries	1	0	0	1	0	1	0	1	0	1	0
Steak	0	1	0	0	0	0	0	0	0	0	0
Dessert	0	1	0	0	1	0	1	1	1	0	0
Pizza	0	0	1	0	0	0	1	0	1	0	0
Risotto	0	0	0	0	0	0	1	0	0	0	0
Sushi	0	0	0	0	0	0	0	0	0	0	1
Noodle	0	0	0	0	1	0	0	0	0	0	0

filter out the objects that do not match the query keywords  $q.\psi$ . Consider our toy dataset given in Fig. 2 as an example, which contains a set of restaurants and the corresponding keywords which describe the foods provided by each restaurant. The inverted index lists all the keywords appeared and the corresponding objects in our toy dataset as shown in Table 1. Now, if the user is only interested in the restaurants which provide Australian food, then the objects  $\{p_1, p_2, p_3, p_4, p_6, p_8, p_9, p_{10}\}$  are returned as keyword match objects. For multiple query keyword, we can simply calculate the intersection set among keyword match objects of each query keyword. After filtering data objects based on keyword inverted index, the next step is to perform the dominance checking among them. However, the access order of the database objects cannot be random as per the following lemma.

**Lemma 3.** *The TDSS of a query point  $q$  will be correct if and only if we access the database objects in order of their distances to  $q$  given that we compare their dominances with the objects accessed so far.*

*Proof.* Assume that there are three points  $\{p_1, p_2, p_3\}$  and the following relationships hold: (a)  $d(q, p_1) < d(q, p_2) < d(q, p_3)$ ; (b)  $p_2 \in DZ(p_1)$ ; (c)  $p_3 \in DZ(p_2)$  and (d)  $p_3 \notin DZ(p_1)$ . Now, assume again that we access these points from the database in the following order: first  $p_2$ , then  $p_1$  and  $p_3$  and compare their directional dominances with the points accessed so far only. The TDSS result for the above would be  $\{p_1, p_2\}$ , which is incorrect as per Definition 4. However, if we access them in the following order: first  $p_1$ , then  $p_2$  and  $p_3$ , the TDSS result would be  $\{p_1, p_3\}$ , which is correct as per Definition 4. Hence, the lemma.

**Algorithm Steps.** The KFBA algorithm firstly constructs the keyword inverted index of the dataset  $P$ . Then, it performs keyword based filtering to find keyword match objects,  $\forall p \in P : p.\psi = q.\psi$ . After that, a min heap  $\mathcal{H}$  is created by inserting the filtered database objects  $P'$  in order of their distances to the query point  $q$ . Finally, we initialize the spatial skyline set  $S$  to  $\emptyset$ , then repeatedly retrieve the root element  $e$  from  $\mathcal{H}$  until  $\mathcal{H}$  become  $\emptyset$  and append  $e$  to  $S$  iff  $\nexists s \in S : s \prec e$ . The above steps are pseudo-coded in Algorithm 1.

---

**Algorithm 1: Keyword Filtering Based Approach (KFBA)**

---

**Input** :  $q$ : query point,  $P$ : dataset  
**Output** :  $S$ : a list of textually relevant spatial skyline objects  
**Initialization:**  $S \leftarrow \emptyset$

```
1 invertedIndex  $\leftarrow$  buildInvertedIndex( $P$ );           // build inverted index
2  $P' \leftarrow$  searchKeywordMatchObjects(invertedIndex,  $q.\psi$ ); // find keyword match
   objects
3  $\mathcal{H} \leftarrow$  insert( $P'$ );                               // insert  $P'$  into min heap  $\mathcal{H}$ 
4 while  $\mathcal{H} \neq \emptyset$  do
5    $e \leftarrow \mathcal{H}.\text{pop}()$ ;                               // pop the root element
6   if  $\nexists s \in S : s \prec e$  then
7      $S \leftarrow$  append( $e$ );                               //  $e$  is a spatial skyline object
8 return  $S$ ;                                              // final spatial skyline set
```

---

### 3.2.2 Branch and Bound Keyword Matching

In this section, we present our branch and bound keyword matching approach which is based on the R-tree indexing structure. The main idea of the BBKM approach is to avoid the object-to-object dominance checking as much as possible by pruning R-tree bounding boxes as per the following lemma.

**Lemma 4.** *A bounding box  $R_k$  in R-Tree can be safely pruned if  $\exists s \in S$  such that all of the vertices of  $R_k$  are inside  $DZ(s)$ , where  $S$  is the current skyline of the database objects accessed so far in order of their distances to the query point  $q$ .*

*Proof.* Every database object  $p \in R_k$  is bounded by the vertices of the bounding box  $R_k$  in R-Tree. These vertices are the corner points of the minimum bounding box  $R_k$  in R-Tree. Therefore, every  $p \in R_k$  is inside  $DZ(s)$  as all the vertices of  $R_k$  are inside  $DZ(s)$ , i.e.,  $p \in DZ(s_i), \forall p \in R_k$ . Now, every database object  $p \in R_k$  has the same direction as  $s$  w.r.t.  $q$  as well as  $d(q, s) < d(q, p)$  as  $p \in DZ(s)$  (as per Section 2 and Lemma 1). Therefore, we get  $s \prec p, \forall p \in R_k$  and the bounding box  $R_k$  can be pruned safely without further processing. Hence, the lemma.

Consider the dataset given in Fig 2 and the R-tree indexing given in Fig. 4. Assume that a user wants to search for the surrounding restaurants around her which provide Australian food. In this case, the query keyword is  $q.\psi = \{\text{"Australian"}\}$ . Here, the bounding box  $R_4$  can be directly eliminated as it is directionally dominated by the data objects  $p_9$  as per Lemma 4. Also, the data objects  $p_5$  and  $p_7$  are ignored during the processing as these two data objects do not contain the keyword "Australian", i.e., are nor keyword match objects.

**Algorithm Steps.** Based on Lemma 3 and Lemma 4, the BBKM approach can be explained as given below. Firstly, the algorithm index the dataset  $P$  into an R-tree, initialize the skyline result set  $S$  to  $\emptyset$  and insert the root element  $e$  of R-Tree into a min-heap  $\mathcal{H}$ . The BBKM keep accessing the top element  $e$  from  $\mathcal{H}$  until  $\mathcal{H}$  becomes  $\emptyset$ , the element  $e$  is examined further if  $\nexists s \in S : s \prec e$ . There are three case to consider as follows: (i)  $e$  is an intermediate node: we insert each child  $e_i$  of

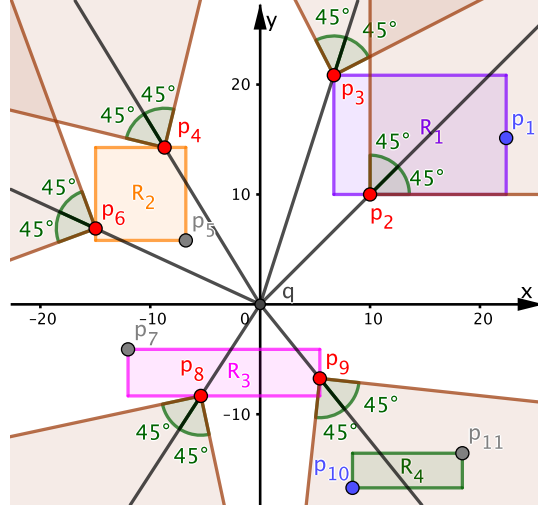


Fig. 4: R-Tree (MAX #entries=4) bounding boxes of the data in Fig. 2 and dominance checking for BBKM

---

**Algorithm 2:** Branch and Bound Keyword Matching (BBKM)

---

**Input** :  $q$ : query point,  $R$ : R-Tree indexing of dataset  $P$   
**Output** :  $S$ : a list of textually relevant spatial skyline objects  
**Initialization:**  $S \leftarrow \emptyset$ ;

```

1  $\mathcal{H} \leftarrow \text{insert}(\text{getRoot}(R))$ ; // insert root of  $R$  into min-heap  $\mathcal{H}$ 
2 while  $\mathcal{H} \neq \emptyset$  do
3    $e \leftarrow \mathcal{H}.\text{pop}()$ ; // pop the root element
4   if  $\nexists s \in S : s \prec e$  then
5     if  $e \neq \text{leaf}$  then // intermediate node in R-Tree
6       foreach child  $e_i$  of  $e$  do
7         if  $\nexists s \in S : s \prec e_i$  then
8            $\mathcal{H} \leftarrow \text{insert}(e_i)$ ; // insert box  $e_i$  into heap  $\mathcal{H}$ 
9       else if  $e = \text{leaf}$  then // leaf node in R-Tree
10        foreach child  $e_i$  of  $e$  do
11          if  $\nexists s \in S : s \prec e_i$  and  $\text{isKeywordMatch}(e_i)$  then
12             $\mathcal{H} \leftarrow \text{insert}(e_i)$ ; // insert object  $e_i$  into heap  $\mathcal{H}$ 
13        else // a database point
14           $S \leftarrow \text{append}(e)$ ; //  $e$  is a spatial skyline point
15 return  $S$ ; // final spatial skyline set

```

---

$e$ , which is a R-tree node, into  $\mathcal{H}$  iff  $\nexists s \in S : s \prec e_i$ ; (ii)  $e$  is a leaf node, then we insert each child  $e_i$  of  $e$ , which is a database object, into  $\mathcal{H}$  iff  $\nexists s \in S : s \prec e_i$  and  $e_i$  is a keyword match object; and (iii)  $e$  is a database object and is a keyword match for  $q$ , i.e.,  $e.\psi = q.\psi$ , we insert  $e$  into the skyline result set  $S$ . The above steps are pseudo-coded in Algorithm 2. It should be noted that the min heap  $\mathcal{H}$  stores only the R-tree nodes and keyword match objects.



## 4 Experiments

**Setup.** The experiments are conducted on both real and synthetic datasets. The real dataset is a POI dataset, which contains 104770 locations in California (CA) [16]. We randomly select 25K, 50K, 75K and 100K points from this dataset as candidate points. We also generate synthetic (SYN) datasets consisting of 125K, 250K, 375K and 500K uniformly distributed points. We randomly create 50 query points following the distribution of the datasets to report the performance of the tested algorithms. Each point is assigned 10 random keywords from a keyword pool of 63 keywords. We randomly create 50 query points by following the distribution of the dataset and assign 1-4 keywords to conduct our experiments. To index the datasets in R-tree, we set MAX #entries in a R-Tree node to 20 – 50 and default setting is 30. The algorithms are implemented in Java and experiments are conducted on a Mac Laptop with 2 GHz Intel Core i7 CPU and 8 GB 1600 MHz DDR3 main memory.

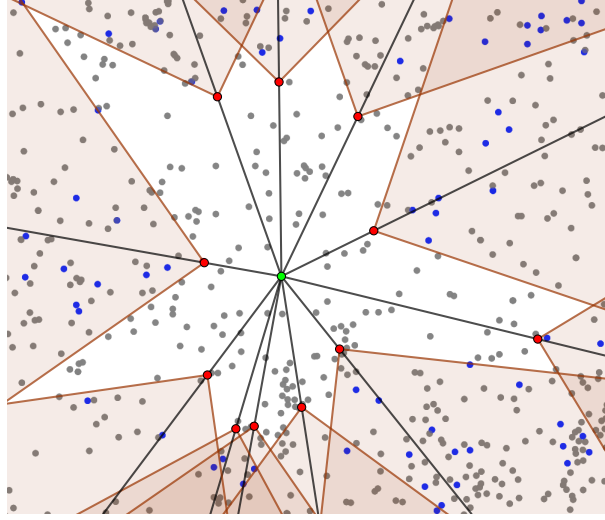


Fig. 5: Non-dominated keyword match surrounding objects retrieval for an arbitrary user via TDSS query model in CA dataset

### 4.1 Effectiveness Study

Here, we demonstrate the usefulness of our TDSS query model through a case study. The case study result for an arbitrary user (query) in CA dataset is visualized in Fig. 5, where the user is shown as green point. The gray points represent the non-keyword matched POI objects, whereas the candidate keyword match POI objects and the resultant objects are shown as blue and red, respectively. It is obvious that our proposed TDSS query model can retrieve non-dominated surrounding keyword match objects for an arbitrary user by trading-off the direction and distance.

## 4.2 Efficiency Study

This section presents the efficiency study of the proposed algorithms.

**Effect of Query Keywords.** Here, we examine the effect of query keywords on the efficiency of the proposed algorithms and the results are illustrated in Fig. 6. It is evident from Fig. 6 that the proposed BBKM algorithm outperforms our KFBA algorithm for lower number of query keywords (1 – 3). On the other hand, KFBA algorithm performs better for higher number of query keywords (3 – 4 and more). This is because, the KFBA algorithm ends up having only a few data objects after keyword filtering and thereby, reduces the size of the heap significantly at run time.

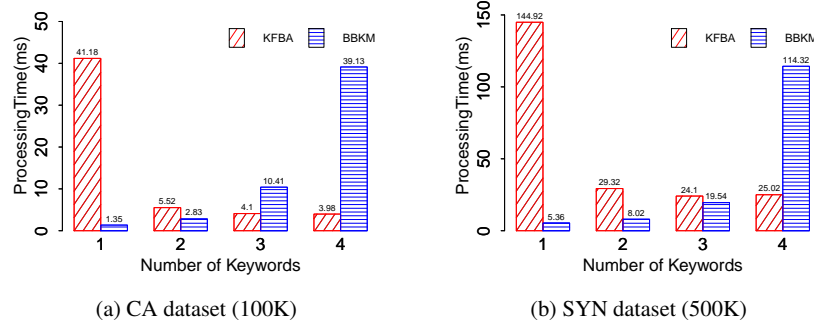


Fig. 6: Effect of keywords: avg. processing time (ms) in (a) CA and (b) SYN datasets

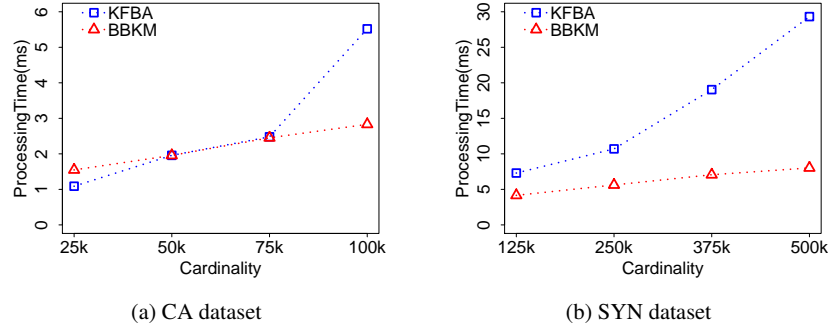


Fig. 7: Effect of cardinality: avg. processing time (ms) in (a) CA and (b) SYN datasets ( $|q.\psi| = 2$ )

**Effect of Data Cardinality.** Here, we examine the effect of cardinality on the efficiency of our proposed algorithms and the results are shown in Fig 7. From Fig 7(a), it is evident that the KFBA algorithm runs faster than the BBKM algorithm for lower cardinality(25K – 50K). On the other hand, the BBKM algorithm performs much better for higher cardinality settings. From Fig 7(b), it is obvious that the BBKM algorithm outperforms our KFBA algorithm for all cardinality settings in SYN dataset. This is because the KFBA algorithm is time-consuming when there are many objects after keyword filtering. Also, multiple keywords searching in inverted index structure for higher cardinality setting is time consuming.

**Effect of R-tree Parameter.** Here, we examine the effect of R-tree parameter on the efficiency of BBKM algorithm and the results are shown in Table 2 for  $|q, \psi| = 2$ . The proposed BBKM algorithm is tolerant in all different R-tree MAX #entries for both CA and SYN

datasets. However, it is hard to decide which R-tree parameter setting is optimum. For the larger R-tree parameter setting, the BBKM algorithm may need to examine more objects to find the TDSS result. However, the larger R-tree MAX #entries may also have the advantage of pruning more objects that are inside a bounding box.

Table 2: Avg. running time (ms) of BBKM algorithm in CA and SYN datasets for different settings of MAX #entries in a R-tree node

Approach	20	30	40	50
CA dataset	2.877	2.830	2.324	2.758
SYN dataset	5.92	8.02	9.18	7.4

## 5 Related Work

The skyline operator proposed by Borzsony et al. [1] has gained significant attention among the researchers [4, 5, 9–12, 18, 19] including the spatial database community [15, 17, 21, 23, 24]. However, none of these approach considers surroundingness in the non-dominating objects retrieval by trading off the distance and direction.

The direction-based spatial skyline query (DSQ) was first discussed by Guo et al. [7, 8] and they propose to retrieve data objects that surround the query point based on user given angle threshold. The DSQ queries have missing result and instability problems as discussed in Section 1. The *nearest surrounder queries* (NSQ) proposed by Lee et al. [13, 14] retrieve nearest surrounder objects of arbitrary shapes that are visible from the given query point. Both DSQ and NSQ queries do not emphasize textual relevance of the retrieved objects.

Existing works on the retrieval of textually relevant objects in spatial databases are quite established ([2, 3, 6, 20, 22, 25] for survey). These works emphasize on trading off the spatial proximity and the textual relevance in spatio-textual data context, rather than distance and direction. As these works do not consider surroundingness in their retrieval system, they may return multiple objects in the same direction.

Our work differs from the existing works in the sense that we emphasize not only the textual relevance in the retrieved objects, but also we trade off the direction and distance to retrieve non-dominating surrounding objects for a given query point. Our TDSS queries are also fair and free from missing result and instability problems.

## 6 Conclusion

In this paper, we have presented a novel query called textually-relevant direction-based spatial skyline (TDSS) for retrieving textually relevant non-dominating surrounding data objects in spatial databases. We have also presented efficient algorithms to process TDSS queries in spatial databases. The proposed algorithms have been evaluated by experimenting with both real and synthetic datasets.

**Acknowledgement.** This work was partially supported by a Griffith University's 2018 New Researcher Grant with Dr. Md Saiful Islam being the Chief Investigator. The first and second authors contributed equally in this paper.

## References

1. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
2. Cao, X., Chen, L., Cong, G., Jensen, C.S., Qu, Q., Skovsgaard, A., Wu, D., Yiu, M.L.: Spatial keyword querying. In: ER, pp. 16–29 (2012)
3. Cary, A., Wolfson, O., Rishé, N.: Efficient and scalable method for processing top-k spatial boolean queries. In: SSDBM, pp. 87–95 (2010)
4. Chan, C.Y., Jagadish, H.V., Tan, K., Tung, A.K.H., Zhang, Z.: Finding k-dominant skylines in high dimensional space. In: SIGMOD, pp. 503–514 (2006)
5. Dellis, E., Seeger, B.: Efficient computation of reverse skyline queries. In: VLDB, pp. 291–302 (2007)
6. Felipe, I.D., Hristidis, V., Rishé, N.: Keyword search on spatial databases. In: ICDE, pp. 656–665 (2008)
7. Guo, X., Ishikawa, Y., Gao, Y.: Direction-based spatial skylines. In: MobiDE, pp. 73–80 (2010)
8. Guo, X., Zheng, B., Ishikawa, Y., Gao, Y.: Direction-based surrounder queries for mobile recommendations. VLDB J. **20**(5), 743–766 (2011)
9. Islam, M.S., Liu, C.: Know your customer: computing k-most promising products for targeted marketing. VLDB J. **25**(4), 545–570 (2016)
10. Islam, M.S., Liu, C., Rahayu, J.W., Anwar, T.: Q+tree: An efficient quad tree based data indexing for parallelizing dynamic and reverse skylines. In: CIKM, pp. 1291–1300 (2016)
11. Islam, M.S., Rahayu, J.W., Liu, C., Anwar, T., Stantic, B.: Computing influence of a product through uncertain reverse skyline. In: SSDBM, pp. 4:1–4:12 (2017)
12. Islam, M.S., Zhou, R., Liu, C.: On answering why-not questions in reverse skyline queries. In: ICDE, pp. 973–984 (2013)
13. Lee, K.C.K., Lee, W., Leong, H.V.: Nearest surrounder queries. IEEE Trans. Knowl. Data Eng. **22**(10), 1444–1458 (2010)
14. Lee, K.C.K., Schiffman, J., Zheng, B., Lee, W., Leong, H.V.: Tracking nearest surrounders in moving object environments. In: ICPS, pp. 3–12 (2006)
15. Lee, M.W., Son, W., Ahn, H.K., Hwang, S.W.: Spatial skyline queries: exact and approximation algorithms. GeoInformatica **15**(4), 665–697 (2011)
16. Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G., Teng, S.: On trip planning queries in spatial databases. In: SSTO, pp. 273–290 (2005)
17. Lin, Q., Zhang, Y., Zhang, W., Li, A.: General spatial skyline operator. In: DASFAA, pp. 494–508 (2012)
18. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting stars: The k most representative skyline operator. In: ICDE, pp. 86–95 (2007)
19. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: SIGMOD, pp. 467–478 (2003)
20. Rocha-Junior, J.B., Gkorgkas, O., Jonassen, S., Nøravåg, K.: Efficient processing of top-k spatial keyword queries. In: SSTO, pp. 205–222 (2011)
21. Sharifzadeh, M., Shahabi, C.: The spatial skyline queries. In: VLDB, pp. 751–762 (2006)
22. Shi, J., Wu, D., Mamoulis, N.: Textually relevant spatial skylines. IEEE Trans. Knowl. Data Eng. **28**(1), 224–237 (2016)
23. Sohail, A., Cheema, M.A., Taniar, D.: Social-aware spatial top-k and skyline queries. Comput. J. **61**(11), 1620–1638 (2018)
24. Son, W., Lee, M.W., Ahn, H.K., Hwang, S.W.: Spatial skyline queries: An efficient geometric algorithm. In: SSTO, pp. 247–264. Springer (2009)
25. Wu, D., Yiu, M.L., Cong, G., Jensen, C.S.: Joint top-k spatial keyword query processing. IEEE Trans. Knowl. Data Eng. **24**(10), 1889–1903 (2012)