Efficient Continuously Moving Top-K Spatial Keyword Query Processing

Dingming Wu [1], Man Lung Yiu [2], Christian S. Jensen [3], Gao Cong [4]

[1] *Department of Computer Science, Aalborg University, Denmark*
dingming@cs.aau.dk

[2] *Department of Computing, Hong Kong Polytechnic University, Hong Kong*
csmlyiu@comp.polyu.edu.hk

[3] *Department of Computer Science, Aarhus University, Denmark*
csj@cs.au.dk

[4] *School of Computer Engineering, Nanyang Technological University, Singapore*
gaocong@ntu.edu.sg

***Abstract*— Web users and content are increasingly being geo-positioned. This development gives prominence to spatial keyword queries, which involve both the locations and textual descriptions of content.**

We study the efficient processing of continuously moving top-k spatial keyword (MkSK) queries over spatial keyword data. State-of-the-art solutions for moving queries employ *safe zones* that guarantee the validity of reported results as long as the user remains within a zone. However, existing safe zone methods focus solely on spatial locations and ignore text relevancy.

We propose two algorithms for computing safe zones that guarantee correct results at any time and that aim to optimize the computation on the server as well as the communication between the server and the client. We exploit tight and conservative approximations of safe zones and aggressive computational space pruning. Empirical studies with real data suggest that our proposals are efficient.

I. Introduction

The Internet is increasingly being accessed by mobile users. This development is fueled by advances in mobile devices, networks, and services. Further, the Internet is acquiring a spatial dimension, with content and users increasingly being geo-positioned.

This development calls for *spatial keyword queries* that integrate location with text search [1]–[5]. Taking a location and a set of keywords as arguments, such queries return relevant content that matches the arguments. However, all existing proposals assume a static query location at a snapshot. In contrast, we consider the *moving top-k spatial keyword* (MkSK) query, which takes into account a continuously moving query location. This query enables a mobile user (or driver) to be continuously aware of the k spatial web objects that best match a query with respect to location and text relevancy. A tourist visiting New York City may activate a "lunch special vegetarian" query when lunch approaches to be alerted about nearby opportunities for lunch. Individuals looking for entertainment may issue a "happy hour free snacks" query in the late afternoon to be alerted about bars with happy hour deals with free snacks. With the MkSK query, a user always has an up-to-date result as the user moves. The user can ignore a result and just keep moving until an appealing result appears.

The straightforward solution to the MkSK query, of periodically invoking an existing snapshot spatial keyword query processing technique (e.g., [2]–[4], [6]), either yields excessive costs or outdated results. Indeed, even with a very small period, correct near-future results are not guaranteed.

State-of-the-art proposals for moving queries [7]–[9] adopt a standard client-server architecture along with *safe zones*. In response to a query, the server computes the result and a safe zone, which are sent to the client. As long as the client stays in the safe zone, the result is guaranteed to remain correct. Only when leaving the safe zone, the client sends a location update to the server, which repeats the above process.

To the best of our knowledge, this paper contains the first study of moving spatial keyword queries. Voronoi cells [7] have been used as safe zones for nearest neighbor queries. However, these consider only location, not text relevancy. We capture also text relevancy by using weights and then define safe zones using multiplicatively weighted Voronoi (MW-Voronoi) cells [10]. MW-Voronoi diagrams have been applied extensively in geographic, market, and urban analysis. For instance, they have been applied to study supply points that result from consumer attraction [11], the U.S. national systems of planning regions [12], and urban settlement and socioeconomic structures [13].

The concept of weighted distance plays an important role in the MW-Voronoi diagram. Assuming a data set $\mathcal{D}$ of points $p = (p_x, p_y)$ with weight $w(p)$, the weighted distance between p and a query point q is defined as $d_w(q, p) = \frac{\|q\ p\|}{w(p)}$, where $\|q\ p\|$ denotes their Euclidean distance. The MW-Voronoi cell of a point defines its influence region based on its weight. Figure 2(b) shows a data set $\{p_1, p_2, p_3, p_4, p^*\}$, where p^* is the top result of q. The shaded region is the MW-Voronoi cell of p^*—all locations in this region have smaller weighted distance to p^* than any other point p_i. In our setting, the weight $w(p)$ of a data object p is query-dependent and is determined by the text relevancy between the object's description and the keywords in the query.

We use the weighted distance $d_w(q, p)$ as the ranking function for MkSK queries. A key strength of this function is that the different measurement units of the weights (i.e., text relevancy) and the spatial distances do not affect the result because the effect of the units is canceled in the ratio of the weights used (covered in Definition 1 in detail).

Existing methods for constructing MW-Voronoi diagrams involve a full scan of the data [14]. In our setting, we need only compute a single MW-Voronoi cell, for which the existing

methods are inefficient. Pre-computation is inapplicable in our setting because the weight $w(p)$ of a point p is known only when a query is received.

We develop two algorithms for computing safe zones using a tree-based index [2]–[4], [6] over a spatial keyword data set, so that only a small fraction of the data objects are accessed during computation. Our objectives are to optimize the computational cost at the server as well as the client/server communication cost. The contributions of the paper are summarized as follows.

1) We formalize the continuously moving top-k spatial keyword query problem.
2) We develop an early stop algorithm (IBD) to compute safe zones. It utilizes topological pruning to discard objects that cannot contribute to a safe zone, and it also incorporates an early stopping condition.
3) We present an advanced algorithm (MSK-uvr) that enables pruning of subtrees of objects that do not contribute to the safe zone, to improve efficiency. It also applies two optimizations to further reduce the search space and the communication cost.
4) We offer empirical insight into the performance of the proposed algorithms.

The rest of the paper is organized as follows. We first present the problem setting and provide background knowledge in Section II. Then we develop the early stop algorithm (IBD) in Section III and the advanced algorithm (MSK-uvr) in Section IV. Next we study the performance of our proposals on real data sets in Section V. Then we discuss related work in Section VI, followed by conclusions in Section VII.

II. Problem Setting and Background

Following a formal problem definition, we provide background on the MW-Voronoi diagram and a tree-based index for spatial keyword data.

A. Problem Definition

We consider a data set $\mathcal{D}$ in which each object $p \in \mathcal{D}$ is a pair $\langle \lambda, \psi \rangle$ of a point location $p.\lambda$ and a text description, or document, $p.\psi$ (e.g., the facilities and menu of a restaurant). Next, a *top-k spatial keyword query* $q = \langle \lambda, \psi, k \rangle$ takes three arguments: a point location λ, a set of keywords ψ, and a number of requested objects k.

Let the function $tr_{q.\psi}(p.\psi)$ denote the text relevancy of $p.\psi$ to $q.\psi$. Our approach is applicable to any information retrieval model, e.g., TFIDF [22].

Intuitively, an object whose description is more relevant to the query keywords and is closer to the query location is preferable. We use a weighted distance [16] in Equation 1 as our ranking function, since it matches the semantics of the query.

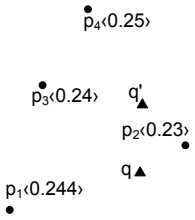
$$rank_q(p) = \frac{\|q.\lambda\ p.\lambda\|}{tr_{q.\psi}(p.\psi)}, \tag{1}$$

where $\|q.\lambda\ p.\lambda\|$ denotes their Euclidean distance. An object p with a smaller value of $rank_q(p)$ is ranked higher (more relevant to the query).

The *top-k spatial keyword query* q returns a list of k objects from $\mathcal{D}$ that minimize the ranking value and that are in ascending order of their ranking values. Formally, the result, denoted by $\mathcal{RS}$, is a list of k objects from $\mathcal{D}$ that satisfy the following condition:

$$\forall p \in \mathcal{RS}(\ \forall p' \in \mathcal{D} - \mathcal{RS}(\ rank_q(p) \leq rank_q(p'))) \tag{2}$$

Figure 1(a) shows the locations of a set of objects $\mathcal{D} = \{p_1, p_2, p_3, p_4\}$. Figure 1(b) shows the word frequencies for each object. Let the query q shown in Figure 1(a) with $q.\psi = \langle a, b \rangle$ and $q.k = 2$ be given. The number in brackets next to each object is its text relevancy to the query keywords $q.\psi$ that are computed on-the-fly using the text relevancy function $tr_{q.\psi}(p.\psi)$. The result of query q is $\langle p_2, p_3 \rangle$ according to function $rank_q(\cdot)$. The ranking value of p_2 and p_3 is 0.478 (= 0.11/0.23) and 0.54 (= 0.13/0.24), respectively. When q moves to q', the result rank becomes $\langle p_2, p_4 \rangle$. The ranking value of p_2 and p_4 is 0.478 and 0.48, respectively.



	tf	a	b
p_1	3	4
p_2	1	2
p_3	2	3
p_4	4	5

	$\|q\ p_i\|$	$\|q'\ p_i\|$
p_1	0.14	0.15
p_2	0.11	0.11
p_3	0.13	0.12
p_4	0.14	0.12

(a) locations (b) descriptions (c) Euclidean distances

Fig. 1. Example moving top-k spatial keyword query

Problem Statement.
We study the efficient processing of *moving top-k spatial keyword* (MkSK) queries over static objects in Euclidean space. Thus, the spatial location of a query changes continuously whereas the keywords of a query remain constant.

We aim for a solution that (i) guarantees that the client has a correct result at any point in time, (ii) optimizes the computational server-side cost, and (iii) optimizes the client/server communication cost.

B. Multiplicatively Weighted Voronoi Diagram

Our approach to computing the moving top-k spatial keyword query utilizes the so-called multiplicatively weighted Voronoi (MW-Voronoi) diagram [10]. Here, we recall the definitions of MW-Voronoi diagrams and regions.

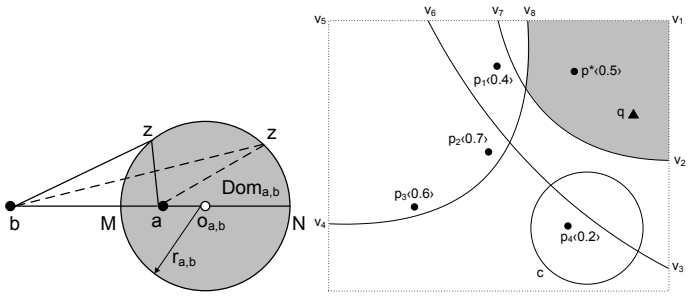
Let $\mathcal{D}$ be a set of weighted points in two-dimensional Euclidean space $\mathcal{U}$. A point a in $\mathcal{D}$ has (i) a weight $w(a)$ and (ii) coordinates (a_x, a_y). The weighted distance between any point z in $\mathcal{U}$ and a is defined as $d_w(z, a) = \frac{\|z\ a\|}{w(a)}$, where $\|z\ a\|$ is the Euclidean distance between z and a.

Definition 1: The *dominant region* of point a over point b is defined as:

$$Dom_{a,b} = \{z \in \mathcal{U} \mid d_w(z, a) \leq d_w(z, b)\} \tag{3}$$

To characterize the shape of $Dom_{a,b}$, we define the Apollonius circle [17] and explain its relationship with $Dom_{a,b}$. As an example, Figure 2(a) shows the Apollonius circle of

two points a and b. Any location z on the circle satisfies the equation $\|z\ a\| = \mu \cdot \|z\ b\|$.



(a) Appollonius circle (b) MW-Voronoi region

Fig. 2. Example MW-Voronoi region

Definition 2: (Durell [17]) Given two points a and b and a constant $0 \leq \mu \leq 1$, the *Apollonius circle* $C_{a,b}$ is defined as all locations z that satisfy: $\|z\ a\| = \mu \cdot \|z\ b\|$. The *Apollonius circular region* $C_{a,b}$ is defined as the region: $\|z\ a\| \leq \mu \cdot \|z\ b\|$.

Lemma 1: **[Radius and center, Apollonius circle]** (Okabe et al. [10]) Given two points a and b such that $w(a) < w(b)$, $Dom_{a,b} = C_{a,b}$. The center $o_{a,b}$ and the radius $r_{a,b}$ of $C_{a,b}$ are shown in Equations 4 and 5. In addition, we have: $Dom_{b,a} = \mathcal{U} - C_{a,b}$, where $\mathcal{U}$ is the spatial domain.

$$o_{a,b} = \left(\frac{w^2(b)\cdot a_x - w^2(a)\cdot b_x}{w^2(b)-w^2(a)}, \frac{w^2(b)\cdot a_y - w^2(a)\cdot b_y}{w^2(b)-w^2(a)}\right) \quad (4)$$

$$r_{a,b} = \frac{w(a)\cdot w(b)\cdot \|a\ b\|}{w^2(b)-w^2(a)} \quad (5)$$

We will use circle and circular region interchangeably.

For the special case $w(a) = w(b)$, the Apollonius circular region $C_{a,b}$ degenerates to the perpendicular half plane $\perp_{a,b}$. In general, the dominant region $Dom_{a,b}$ is expressed as:

$$Dom_{a,b} = \begin{cases} C_{a,b} & \text{if } w(a) < w(b) \\ \mathcal{U} - C_{b,a} & \text{if } w(a) > w(b) \\ \perp_{a,b} & \text{if } w(a) = w(b) \end{cases} \quad (6)$$

The MW-Voronoi diagram of $\mathcal{D}$ is the collection of MW-Voronoi regions of all points in $\mathcal{D}$. These regions form a (disjoint and complete) partitioning of the spatial domain [10].

Definition 3: Given a (result) point $p^* \in \mathcal{D}$, its *MW-Voronoi region* with respect to $\mathcal{D}$ is defined as:

$$\Upsilon(p^*) = \bigcap_{p' \in \mathcal{D} - \{p^*\}} Dom_{p^*,p'} \quad (7)$$

Based on a (result) point $p^* \in \mathcal{D}$, we can partition $\mathcal{D}$ into $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^- \cup \mathcal{D}^o \cup \{p^*\}$, where set D^+ contains all points with higher weight than p^*, set D^- contains all points with lower weight than p^*, and set $\mathcal{D}^o$ contains the objects whose weights are identical to p^*. According to Mu et al. [14], by applying Equation 6, we can re-express the MW-Voronoi region as Equation 8. Thus, higher-weight neighbors add to the MW-Voronoi region of a point, forming convex edges; lower-weight neighbors subtract from it, forming concave edges; and equal-weight neighbors crop it with straight lines.

$$\begin{aligned} \Upsilon(p^*) &= \bigcap_{p \in \mathcal{D} - \{p^*\}} Dom_{p^*,p} \quad (8) \\ &= \bigcap_{p_j \in \mathcal{D}^+} C_{p^*,p_j} \cap \bigcap_{p_i \in \mathcal{D}^o} \perp_{p^*,p_i} \cap \bigcap_{p_k \in \mathcal{D}^-} (\mathcal{U} - C_{p_k,p^*}) \\ &= \bigcap_{p_j \in D^+} C_{p^*,p_j} \cap \bigcap_{p_i \in \mathcal{D}^o} \perp_{p^*,p_i} - \bigcup_{p_k \in D^-} C_{p_k,p^*} \end{aligned}$$

Figure 2(b) shows a point set $\mathcal{D} = \{p_1, p_2, p_3, p_4, p^*\}$. The number in brackets next to each point is its weight. The MW-Voronoi region of p^*, $\Upsilon(p^*)$, is shaded. Every location in this region has a smaller weighted distance $d_w(\cdot,\cdot)$ to p^* than to any other point in $\mathcal{D}$. We have that $D^+ = \{p_2, p_3\}$ and $D^- = \{p_1, p_4\}$. As shown in the figure, $C_{p^*,p_2} = v_1v_2v_7$, $C_{p^*,p_3} = v_1v_3v_6$, $C_{p_1,p^*} = v_4v_5v_8$, and C_{p_4,p^*} is the circle c. The MW-Voronoi region of p^* is then given as

$$\Upsilon(p^*) = C_{p^*,p_2} \cap C_{p^*,p_3} - (C_{p_1,p^*} \cup C_{p_4,p^*}).$$

C. Preliminaries: The IR-Tree

While our algorithms may utilize several tree-based indexes [2]–[4], [6], we adopt the IR-tree [2] for indexing data objects at the server.

The IR-tree [2] is an R-tree [18] extended with inverted files [19]. Each leaf node contains entries of the form $e_o = (\Lambda, \psi)$, where e_o refers to a data object, $e_o.\Lambda$ is a minimum bounding rectangle (MBR), and $e_o.\psi$ refers to the text description. Each leaf node also contains a pointer to an inverted file with the text descriptions of all objects stored in the node.

An inverted file index has two main components.

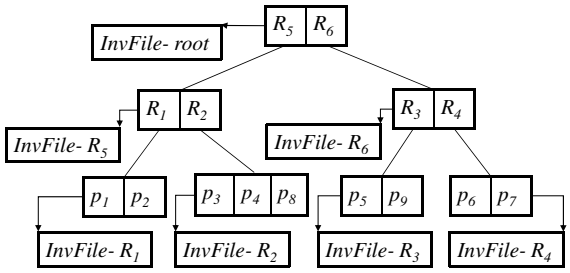
- A vocabulary of all distinct words appearing in the description of some object.
- A posting list for each word t, i.e., a sequence of the identifiers of the objects whose descriptions contain t.

Each non-leaf node N in the IR-tree contains entries of the form $e = (\Lambda, \psi)$, where e is the address of a child node of N, $e.\Lambda$ is the MBR of all rectangles in entries of the child node, and $e.\psi$ refers to a pseudo text description that represents all text descriptions in the entries of the child node. The latter enables derivation of an upper bound on the text relevancy to a query of any object contained in the subtree rooted at e. Each non-leaf node also contains a pointer to an inverted file with the text descriptions of the entries stored in the node.

Figure 3(a) contains nine spatial objects, and Figure 3(b) shows the frequency of words in the description of each object. For example, the description of p_1 contains the words a and c five times each. Figure 4(a) illustrates the corresponding IR-tree, and Figure 4(b) shows the contents of the inverted files associated with the nodes. As a specific example, the weight of the term c in entry R_2 of node R_5 is 7, which is the maximal weight of the term in the three documents in node R_2.

For ease of understanding, we use $tf(t, p.\psi)$ to represent the weight of word t in the running example of the paper.

Lemma 2: **[Monotonicity, text relevancy function]** (Cong et al. [2]) Given a query q and an entry e with its rectangle $e.\Lambda$, we have $\forall p \in e.\Lambda\ (tr_{q.\psi}(e.\psi) \geq tr_{q.\psi}(p.\psi))$.



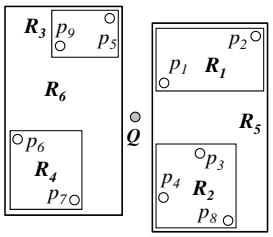
(a) Structure of the IR-tree

InvF-root	*InvF-R_5*	*InvF-R_6*
a: $(R_5,7),(R_6,4)$	a: $(R_1,5),(R_2,7)$	a: $(R_3,4),(R_4,1)$
b: $(R_5,5),(R_6,4)$	b: $(R_1,5),(R_2,3)$	b: $(R_4,4)$
c: $(R_5,7),(R_6,4)$	c: $(R_1,5),(R_2,7)$	c: $(R_3,4),(R_4,4)$
d: $(R_5,1),(R_6,1)$	d: $(R_2,1)$	d: $(R_4,1)$

InvF-R_1	*InvF-R_2*	*InvF-R_3*	*InvF-R_4*
a: $(p_1,5)$	a: $(p_3,7)$	a: $(p_5,4),(p_9,3)$	a: $(p_7,1)$
b: $(p_2,5)$	b: $(p_8,3)$		b: $(p_6,4),(p_7,1)$
c: $(p_1,5),(p_2,5)$	c: $(p_4,7),(p_8,3)$	c: $(p_5,4),(p_9,3)$	c: $(p_6,3),(p_7,4)$
	d: $(p_3,1),(p_4,1)$		d: $(p_7,1)$

(b) Content of the Inverted Files of the nodes

Fig. 4. Example of an IR-tree



(a) object locations

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
a	5	0	7	0	4	0	1	0	3
b	0	5	0	0	0	4	1	3	0
c	5	5	0	7	4	3	4	3	3
d	0	0	1	1	0	0	1	0	0

(b) object descriptions

Fig. 3. A data set of Spatial Keyword Objects

As an example in Figure 4, given any query q, $tr_{q.\psi}(R_5.\psi) \geq tr_{q.\psi}(R_1.\psi) \geq tr_{q.\psi}(p_1.\psi)$.

III. Early Stop Solution

By setting the weight $w(p)$ of each point $p \in \mathcal{D}$ to its text relevancy $tr_{q.\psi}(p.\psi)$, we have that $rank_q(p) = d_w(q,p)$. The *safe zone* of query q is the MW-Voronoi region $\Upsilon(p^*)$, where p^* is the top-1 result of q. In this section, we present an early stop solution that utilizes the IR-tree for computing $\Upsilon(p^*)$, without having to visit all data objects. First, we develop pruning rules for discarding objects that cannot contribute to defining the safe zone. Next, we discuss an efficient ordering for accessing the IR-tree and then present the algorithm. We first consider the case $k = 1$ and end by describing how to extend the solution to arbitrary k.

A. Pruning Rules for Unseen Objects

It is generally far from every data object that contributes to defining the safe zone $\Upsilon(p^*)$. In the example in Figure 2, the (shaded) region $\Upsilon(p^*)$ is defined by points p_1 and p_2 only: without the points p_3 and p_4, we would obtain exactly the same region $\Upsilon(p^*)$. We proceed to present pruning rules for discarding such irrelevant objects. Given a set I of objects in $\mathcal{D} - \{p^*\}$ that have been seen so far, the *temporary safe zone* is defined as:

$$\Omega_I = \bigcap_{p' \in I} Dom_{p^*,p'} \tag{9}$$

For convenience, we will simply use Ω instead of Ω_I. The following lemma states that temporary safe zone Ω is always a superset of the actual safe zone $\Upsilon(p^*)$.

Lemma 3: **[Temporary safe zone, superset property]** It holds that: (i) $\Omega \supseteq \Upsilon(p^*)$, and (ii) $\forall p' \in I\ (\Omega \subseteq Dom_{p^*,p'})$.

Proof: For part (i), observe that:

$$\begin{aligned}\Upsilon(p^*) &= \bigcap_{p' \in (\mathcal{D}-\{p^*\})} Dom_{p^*,p'} \\ &= \Omega \cap \bigcap_{p' \in (\mathcal{D}-\{p^*\}-I)} Dom_{p^*,p'}\end{aligned}$$

Therefore, we obtain: $\Omega \supseteq \Upsilon(p^*)$. For part (ii), since $\Omega = \bigcap_{p' \in I} Dom_{p^*,p'}$, we get $\Omega \subseteq Dom_{p^*,p'}$ for each $p' \in I$. ■

Let p be an unseen object, i.e., $p \notin I$, and $p \in (\mathcal{D} - \{p^*\})$. If the dominant region $Dom_{p^*,p}$ of an unseen object p contains Ω then the intersection of $Dom_{p^*,p}$ and Ω is still Ω, and so p does not contribute to shrinking Ω. We present several pruning rules to discard such objects that do not help shrink Ω.

Recall from Section II-B that we decompose $\mathcal{D}$ into the sets $\mathcal{D}^+$, $\mathcal{D}^-$, and $\mathcal{D}^o$. Similarly, we express the set of seen objects as $I = I^+ \cup I^- \cup I^o$. Any object $p' \in I$ belongs to either I^+, I^-, or I^o, depending on its weight.

Pruning Rule 1: Let p_+ be an unseen object in $\mathcal{D}^+$. If $\exists p' \in I^+\ (C_{p^*,p_+} \supseteq C_{p^*,p'})$ then p_+ cannot affect Ω.

Proof: Let p' be an object of I^+ such that $C_{p^*,p_+} \supseteq C_{p^*,p'}$. By Equation 6, we have: $Dom_{p^*,p_+} \supseteq Dom_{p^*,p'}$. By Lemma 3, since $p' \in I$, we get: $Dom_{p^*,p'} \supseteq \Omega$. Combining them, we obtain: $Dom_{p^*,p_+} \supseteq \Omega$, so $Dom_{p^*,p_+} \cap \Omega = \Omega$. ■

Pruning Rule 2: Let p_- be an unseen object in $\mathcal{D}^-$. If $\exists p' \in I^+\ (C_{p_-,p^*} \cap C_{p^*,p'} = \emptyset)$ then p_- cannot affect Ω.

Proof: Let p' be an object of I^+ such that $C_{p_-,p^*} \cap C_{p^*,p'} = \emptyset$. Thus, we get: $(\mathcal{U} - C_{p_-,p^*}) \supseteq C_{p^*,p'}$. By Equation 6, we have: $Dom_{p^*,p_-} \supseteq Dom_{p^*,p'}$. By Lemma 3, since $p' \in I$, we get: $Dom_{p^*,p'} \supseteq \Omega$. Combining these, we obtain: $Dom_{p^*,p_-} \supseteq \Omega$, and thus $Dom_{p^*,p_-} \cap \Omega = \Omega$. ■

Pruning Rule 3: Let p_- be an unseen object in $\mathcal{D}^-$. If $\exists p' \in I^-\ (C_{p_-,p*} \subseteq C_{p',p^*})$ then p_- cannot affect Ω.

Proof: Let p' be an object of I^- such that $C_{p_-,p*} \subseteq C_{p',p^*}$. Thus, we get: $(\mathcal{U} - C_{p_-,p^*}) \supseteq (\mathcal{U} - C_{p',p^*})$. By Equation 6, we have: $Dom_{p^*,p_-} \supseteq Dom_{p^*,p'}$. By Lemma 3, since $p' \in I$, we get: $Dom_{p^*,p'} \supseteq \Omega$. Combining them, we obtain: $Dom_{p^*,p_-} \supseteq \Omega$, and thus $Dom_{p^*,p_-} \cap \Omega = \Omega$. ■

Pruning Rule 4: Let p_- be an unseen object in $\mathcal{D}^-$. If $\exists p' \in I^o\ (C_{p_-,p^*} \cap \perp_{p^*,p'} = \emptyset)$ then p_- cannot affect Ω.

Proof: Trivial; similar to the proof of rule 2. ■

Pruning Rule 5: Let p_o be an unseen object in $\mathcal{D}^o$. If $\exists p' \in I^+\ (\perp_{p^*,p_o} \supseteq C_{p^*,p'})$ then p_o cannot affect Ω.

Proof: Trivial; similar to the proof of rule 1. ■

Pruning Rule 6: Let p_o be an unseen object in $\mathcal{D}^o$. If $\exists p' \in I^o$ $(\perp_{p^*,p_o} \supseteq \perp_{p^*,p'})$ then p_o cannot affect Ω.

Proof: Trivial; similar to the proof of rule 1. ∎

We summarize the pruning rules in Table I and proceed to exemplify the power of the pruning rules. Thus, let p^* be the top result in Figure 2, and let Ω be the shaded region, with $I^+ = \{p_2\}$ and $I^- = \{p_1\}$. Next, we examine the unseen objects $p_3 \in \mathcal{D}^+$ and $p_4 \in \mathcal{D}^-$. By rule 1, object p_3 does not affect Ω, since $C_{p^*,p_2} = v_1v_2v_7 \subset C_{p^*,p_3} = v_1v_3v_6$. By rule 2, object p_4 does not affect Ω, since $C_{p^*,p_2} \cap C_{p_4,p^*} = \emptyset$.

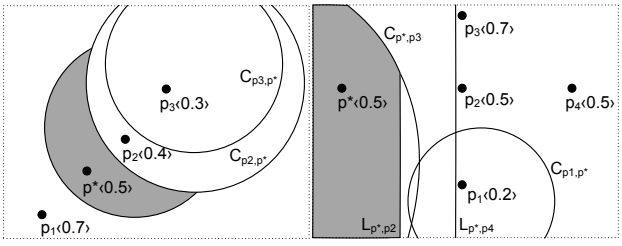
Now let p^* be the top result in Figure 5(a), and let Ω be the shaded region, with $I^+ = \{p_1\}$ and $I^- = \{p_2\}$. We then examine the unseen object $p_3 \in \mathcal{D}^-$. By rule 3, this object does not affect Ω, since $C_{p_2,p^*} \supset C_{p_3,p^*}$.

Finally, let p^* be the top result in Figure 5(b), and let Ω be the shaded region, with $I^+ = \{p_3\}$ and $I^o = \{p_2\}$. We examine unseen objects $p_1 \in \mathcal{D}^-$ and $p_4 \in \mathcal{D}^o$. By rule 4, p_1 does not affect Ω, since $C_{p_1,p^*} \cap \perp_{p^*,p_2} = \emptyset$. By rule 5, p_4 does not affect Ω, since $\perp_{p^*,p_4} \supseteq C_{p^*,p_3}$. Also, by rule 6, p_4 does not affect Ω, since $\perp_{p^*,p_4} \supseteq \perp_{p^*,p_2}$.

TABLE I

PRUNING RULES FOR UNSEEN OBJECTS

	$p_+ \in \mathcal{D}^+$	$p_- \in \mathcal{D}^-$	$p_o \in \mathcal{D}^o$
I^+	Pruning Rule 1: $\exists p' \in I^+$ $(C_{p^*,p_+} \supseteq C_{p^*,p'})$	Pruning Rule 2: $\exists p' \in I^+$ $(C_{p_-,p^*} \cap C_{p^*,p'} = \emptyset)$	Pruning Rule 5: $\exists p' \in I^+$ $(\perp_{p^*,p_o} \supseteq C_{p^*,p'})$
I^-	–	Pruning Rule 3: $\exists p' \in I^-$ $(C_{p_-,p^*} \subseteq C_{p',p^*})$	–
I^o	–	Pruning Rule 4: $\exists p' \in I^o$ $(C_{p_-,p^*} \cap \perp_{p^*,p'} = \emptyset)$	Pruning Rule 6: $\exists p' \in I^o$ $(\perp_{p^*,p_o} \supseteq \perp_{p^*,p'})$



(a) Rule 3 (b) Rules 4, 5, and 6

Fig. 5. Pruning rule examples

Representation of Safe Zone.
The finalized set of objects $I = I^+ \cup I^- \cup I^o$, called the influence set, is used to represent the safe zone. With this set, the client can easily check whether its current location q belongs to the safe zone, by using the Boolean condition: $\bigwedge_{p \in (I^+ \cup I^- \cup I^o)}(q \in Dom_{p^*,p})$. The client needs not render the shape of the safe zone.

Implementation Issues.
For the sake of easy implementation, we convert the topological conditions in the pruning rules into distance-based conditions. For example, in rule 1, the topological condition $\exists p' \in I^+$ $(C_{p^*,p_+} \supseteq C_{p^*,p'})$ is equivalent to the distance-based condition $\exists p' \in I^+$ $(\|o_{p^*,p'}\ o_{p^*,p_+}\| \leq r_{p^*,p_+} - r_{p^*,p'})$, which can be evaluated by using the centers and radii of the respective Appollonius circles. In rule 2, the topological condition $\exists p' \in I^+$ $(C_{p_-,p^*} \cap C_{p^*,p'} = \emptyset)$ is the same as the distance-based condition $\exists p' \in I^+$ $(\|o_{p^*,p'}\ o_{p_-,p^*}\| \geq r_{p_-,p^*} + r_{p^*,p'})$. Similar conversions are applied to the topological conditions of the other rules.

B. Ordering Accesses of Data Objects with Early Stop

This section studies the order in which it is desirable to access the data objects. An early stopping condition is derived, and processing of unpromising objects that cannot contribute to the safe zone is avoided.

The first step is to study the distance between an Apollonius circular region and data points. Figure 2(a) shows an Apollonius circular region $C_{a,b}$ defined by two data points a and b, where $w(a) < w(b)$. We present the concept of border distance in Definition 4 and cover its computation in Lemma 4.

Definition 4: Consider two points a and b such that $w(a) < w(b)$. The *minimum border distance* $bord_{min}(a, C_{a,b})$ (or $bord_{min}(b, C_{a,b})$) is defined as the minimum distance from a (or b) to the border of $C_{a,b}$. The *maximum border distance* $bord_{max}(a, C_{a,b})$ (or $bord_{max}(b, C_{a,b})$) is defined as the maximum distance between a (or b) and $C_{a,b}$.

Lemma 4: **[Border distance computation]** The functions $bord_{min}(\cdot)$ and $bord_{max}(\cdot)$ can be expressed as:

$$bord_{min}(a, C_{a,b}) = \|a\ M\| = \frac{w(a)}{w(a)+w(b)} \cdot \|a\ b\| \quad (10)$$

$$bord_{min}(b, C_{a,b}) = \|b\ M\| = \frac{w(b)}{w(a)+w(b)} \cdot \|a\ b\| \quad (11)$$

$$bord_{max}(a, C_{a,b}) = \|a\ N\| = \frac{w(a)}{w(b)-w(a)} \cdot \|a\ b\| \quad (12)$$

$$bord_{max}(b, C_{a,b}) = \|b\ N\| = \frac{w(b)}{w(b)-w(a)} \cdot \|a\ b\| \quad (13)$$

Proof: Observe that the line $\overline{o_{a,b}\ a\ b}$ intersects the *border* of $C_{a,b}$ at locations M and N. According to Okabe et al. [10], the center $o_{a,b}$ of $C_{a,b}$ is co-linear with points a and b. Therefore, we have: $bord_{min}(a, C_{a,b}) = \|a\ M\|$, $bord_{min}(b, C_{a,b}) = \|b\ M\|$, $bord_{max}(a, C_{a,b}) = \|a\ N\|$, and $bord_{max}(b, C_{a,b}) = \|b\ N\|$.

Recall that the value $w(a)/w(b)$ is a constant. By Definition 2 (of the Apollonius circle), we derive:

$$\frac{\|a\ M\|}{\|b\ M\|} = \frac{w(a)}{w(b)} \Rightarrow \frac{\|a\ M\|}{\|a\ b\| - \|a\ M\|} = \frac{\|a\ b\| - \|b\ M\|}{\|b\ M\|} = \frac{w(a)}{w(b)}$$

$$\frac{\|a\ N\|}{\|b\ N\|} = \frac{w(a)}{w(b)} \Rightarrow \frac{\|b\ N\| - \|a\ b\|}{\|b\ N\|} = \frac{\|a\ N\|}{\|a\ b\| + \|a\ N\|} = \frac{w(a)}{w(b)}$$

Thus, we can express $bord_{min}(\cdot)$ and $bord_{max}(\cdot)$ in terms of $w(a)$, $w(b)$, and $\|a\ b\|$, as in Equations 10–13. ∎

For the special case $w(a) = w(b)$, the Apollonius circle degenerates into a half plane, so we have: $bord_{min}(a, C_{a,b}) = bord_{min}(b, C_{a,b}) = \|a\ \ b\|/2$ and $bord_{max}(a, C_{a,b}) = bord_{max}(b, C_{a,b}) = \infty$. For convenience, we define the minimum border distance of point p from p^* as:

$$bord_{min}(p^*, p) = \begin{cases} bord_{min}(p^*, C_{p^*,p}) \text{ if } w(p^*) \leq w(p) \\ bord_{min}(p^*, C_{p,p^*}) \text{ otherwise} \end{cases} \quad (14)$$

Lemma 5 presents an early stopping threshold τ that enables us to save significant computational cost by skipping a set of unseen objects. In order to utilize the condition, we propose to visit points in the data set in the ascending order of the $bord_{min}(p^*, p)$ value. With this access order, it suffices to check $bord_{min}(p^*, p_{new})$ of the current point in $\mathcal{D}_{new}$, as all unseen objects have an equal or larger $bord_{min}(p^*, p)$ value.

Lemma 5: **[Early stopping threshold]** Let $\tau = \min_{p \in I^+} bord_{max}(p^*, p)$. Let $\mathcal{D}_{new}$ be a set of unseen objects such that $bord_{min}(p^*, p_{new}) > \tau$ for each $p_{new} \in \mathcal{D}_{new}$. Then no object in $\mathcal{D}_{new}$ can affect Ω.

Proof: Let p_+ be the object in I^+ that produces the value τ. Let $\odot(p^*, \tau)$ be the circular region with center p^* and radius τ. By the property of maximum border distance, we have: $Dom_{p^*,p_+} \subseteq \odot(p^*, \tau)$. By Lemma 3, we get: $\Omega \subseteq Dom_{p^*,p_+}$. Thus, we have: $\Omega \subseteq \odot(p^*, \tau)$. —(★)

Let p_{new} be any object in the unseen object set $\mathcal{D}_{new}$. Since we are given $bord_{min}(p^*, p_{new}) > \tau$, we have: $Dom_{p^*,p_{new}} \supset \odot(p^*, \tau)$. Combining this with Equation ★, we obtain: $\Omega \subseteq Dom_{p^*,p_{new}}$. Therefore, p_{new} cannot affect Ω. ■

Figure 6 illustrates how the early stopping works. We visit the objects according to the minimum border distance order: p_1, p_2, p_3, p_4. After visiting p_1, we add it to I^+ and update $\tau = bord_{max}(p^*, C_{p^*,p_1})$. The dashed circle indicates the *stopping circle* with center p^* and radius τ. When we visit point p_3, we find that its minimum border distance exceeds τ. Thus, we stop and do not visit p_4.

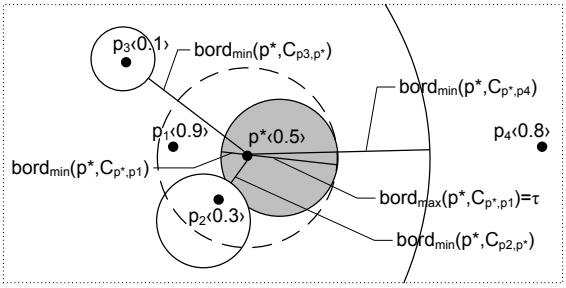


Fig. 6. Early stopping example

C. Incremental Border Distance (IBD) Algorithm

We proceed to present the incremental border distance (IBD) algorithm that utilizes the IR-tree to compute the safe zone $\Upsilon(p^*)$ of the top-1 result p^*. Extension to an arbitrary k is given at the end of the section.

The intended use of the IR-tree calls for a few of definitions. We let Λ be a rectangle that contains a subset of points of $\mathcal{D}$ and let $w^u(\Lambda)$ be the maximum weight of any point in Λ. Also, we let $\|p^* \ \Lambda\|_{min}$ be the minimum distance between p^* and Λ. To be able to apply Lemma 5 to the IR-tree, we need to define the minimum border distance of Λ from p^* such that $bord_{min}(p^*, \Lambda)$ is always a lower-bound of $bord_{min}(p^*, p)$ for any point $p \in \Lambda$.

We consider the two cases in Equation 14. For the first case, we obtain $w(p^*) \leq w(p)$ and then substitute $a = p^*$ and $b = p$ into Equation 10. To minimize the $bord_{min}$ value, we maximize $w(p)$ to $w^u(\Lambda)$, and we minimize the distance $\|p^* \ p\|$ to $\|p^* \ \Lambda\|_{min}$. Thus, we define the minimum border distance of Λ from p^* as:

$$bord_{min}(p^*, \Lambda) = \frac{w(p^*)}{w(p^*) + w^u(\Lambda)} \cdot \|p^* \ \Lambda\|_{min} \qquad (15)$$

For the second case, we obtain $w(p^*) > w(p)$ and then substitute $b = p^*$ and $a = p$ into Equation 11. By minimizing its $bord_{min}$ value, we also obtain the above equation. Thus, we can safely use the above equation for both cases.

Algorithm 1 is the pseudo-code of IBD. It takes the root of the IR-tree, the current query object q, and the result p^* as arguments. The method for computing p^* based on the text relevancy function $rank_q(p_i)$ is orthogonal to our work, and it has been studied by Cong et al. [2].

First, the algorithm creates the sets I^+, I^-, and I^o for storing influence objects. The early stopping threshold τ is initially set to ∞. With a min-heap H, we apply standard best-first search to visit index entries (e.g., nodes or objects) in ascending order of their minimum border distances from p^*.

When the deheaped entry e is an object, we consider three possible cases based on its weight $w(e)$. We then apply the six pruning rules in Table I to discard objects that cannot contribute to the safe zone. Specifically, rule 1 is used to compute I^+; rules 2, 3, and 4 are used to compute I^-; and rules 5 and 6 are used to compute I^o. Each time we insert an object into I^+, we update the threshold τ such that it captures the minimum value of the maximum border distances of objects in I^+. The algorithm terminates when the minimum border distance of the current dequeued entry exceeds τ. Finally, the algorithm reports the set $I^+ \cup I^- \cup I^o$ to the client.

Voronoi Cell Optimization—Enhancing Rules 4 and 6. We propose an optimization to enhance pruning rules 4 and 6. Let Φ be a convex polygon that represents the temporary safe zone defined by the set I^o. As an example, the condition $\exists p' \in I^o(\perp_{p^*,p_o} \supseteq \perp_{p^*,p'})$ of pruning rule 6 can be replaced by $\perp_{p^*,p'} \supseteq \Phi$. The pruning power is higher as Φ is much smaller than the halfplanes $\perp_{p^*,p'}$. We maintain the polygon Φ each time when we insert an object p into I^o. We first derive the halfplane $\perp_{p^*,p}$ and then update Φ by the intersection $\Phi \cap \perp_{p^*,p}$. Instead of keeping the entire I^o, it suffices to use the polygon Φ, whose average number of vertices is six [10].

Extension to Arbitrary k. Let $\mathcal{RS}$ be the top-k result of the query q. According to Okabe et al. [10], the order-k MW-Voronoi region of the set $\mathcal{RS}$, denoted by $\Upsilon^k(\mathcal{RS})$, contains all locations that take the set $\mathcal{RS}$ as the top-k result. In other words, $\Upsilon^k(\mathcal{RS})$ is the safe zone for the result set $\mathcal{RS}$.

We can express the region $\Upsilon^k(\mathcal{RS})$ by using the intersections of order-1 MW-Voronoi regions [10]:

$$\Upsilon^k(\mathcal{RS}) = \bigcap_{p_j^* \in \mathcal{RS}} \Upsilon_{\mathcal{D}-\mathcal{RS}}(p_j^*), \qquad (16)$$

Algorithm 1 IBD (Tree root $root$, Query q, Result p^*)

1: $I^+ \leftarrow$ new set; $I^- \leftarrow$ new set; $I^o \leftarrow$ new set;
2: $\tau \leftarrow \infty$; ▷ τ is the early stop threshold
3: $H \leftarrow$ new min-heap;
4: $H.enheap(root, 0)$;
5: **while** (H is **not** empty) **and** ($H.top.key \leq \tau$) **do**
6: $e \leftarrow H.deheap()$;
7: **if** e is an object **and** $e \neq p^*$ **then**
8: **if** $w(e) > w(p^*)$ **then** ▷ object in $\mathcal{D}^+$
9: **if** $\nexists p' \in I^+(C_{p^*,e} \supseteq C_{p^*,p'})$ **then** ▷ rule 1
10: insert e into I^+;
11: $\tau \leftarrow \min\{\tau, bord_{max}(C_{p^*,e})\}$; ▷ shrink τ
12: **if** $w(e) < w(p^*)$ **then** ▷ object in $\mathcal{D}^-$
13: **if** $\nexists p' \in I^+(C_{e,p^*} \cap C_{p^*,p'} = \emptyset)$ **and** ▷ rule 2
$\nexists p' \in I^-(C_{e,p*} \subseteq C_{p',p^*})$ **and** ▷ rule 3
$\nexists p' \in I^o(\perp_{p^*,p'} \cap C_{e,p^*} = \emptyset)$ **then** ▷ rule 4
14: insert e into I^-;
15: **if** $w(e) = w(p^*)$ **then** ▷ object in $\mathcal{D}^o$
16: **if** $\nexists p' \in I^+(\perp_{p^*,e} \supseteq C_{p^*,p'})$ **and** ▷ rule 5
$\nexists p' \in I^o(\perp_{p^*,p'} \subseteq \perp_{p^*,e})$ **then** ▷ rule 6
17: insert e into I^o;
18: **else** ▷ e points to a child node
19: read the child node CN of e;
20: **for** each entry e' in node CN **do**
21: $H.enheap(e', bord_{min}(p^*, e'.\Lambda))$;
22: return the set $I^+ \cup I^- \cup I^o$;

where $\Upsilon_{\mathcal{D}-\mathcal{RS}}(p_j^*)$ denotes the MW-Voronoi region of p_j^* with respect to the object set $\mathcal{D} - \mathcal{RS}$.

A simple solution for computing the safe zone $\Upsilon^k(\mathcal{RS})$ is to run the IBD algorithm for each result $p_j^* \in \mathcal{RS}$ to obtain the influence object set of $\Upsilon_{\mathcal{D}-\mathcal{RS}}(p_j^*)$. The union of these influence object sets enable the client to determine whether the query belongs to the safe region.

Instead, we propose an efficient extension of IBD that only traverses the IR-tree once, regardless of the value of k. Let the result object of $\mathcal{RS}$ be $p_1^*, p_2^*, \cdots, p_k^*$. For each result object p_j^*, we maintain its influence object sets I_j^+, I_j^o, and I_j^-, and also its early stopping threshold $\tau_j = \min_{p \in I_j^+} bord_{max}(p_j^*, p)$. From Equation 16, we learn that the safe region $\Upsilon^k(\mathcal{RS})$ is the intersection of k order-1 MW-Voronoi regions. Therefore, an object (or entry) can be pruned if it cannot contribute to any such order-1 MW-Voronoi region. For this, we add the following checking condition for the deheaped entry e just after Line 6:

$$\bigvee_{j \in [1,k]} bord_{min}(p_j^*, e'.\Lambda) > \tau_j$$

If it evaluates to true then entry e can be safely pruned.

In order to ensure the correctness of the early stopping condition, we perform the following modifications: (i) replace τ in Line 5 by a global threshold $\tau_{max} = \max_{j \in [1,k]} \tau_j$, and (ii) compute the key of an entry e' (in Line 21) as $\min_{j \in [1,k]} bord_{min}(p_j^*, e'.\Lambda)$.

IV. Advanced Solution

Except for the early stopping condition, IBD can only apply pruning at the object level. Here, we develop techniques capable of pruning entire subtrees that cannot contribute to the safe region. Then, we present an advanced solution for computing the safe zone efficiently, with two optimizations for enhancing the power of the pruning rules.

A. Subtree Pruning

Let Λ be the minimum bounding rectangle (MBR) of a subtree, and let $w^u(\Lambda)$ be the upper bound of the weights of all the objects in Λ. We use $\|\Lambda\ p^*\|_{min}$ (or $\|\Lambda\ p^*\|_{max}$) to denote the minimum (or maximum) Euclidean distance between Λ and the result object p^*. Given a region $\mathfrak{R}$, we define its *minimal dominant region* as:

$$Dom_{p^*,\mathfrak{R}} = \bigcap_{p \in \mathfrak{R},\ 0 \leq w(p) \leq w^u(\mathfrak{R})} Dom_{p^*,p} \qquad (17)$$

With this concept, we can extend the pruning rules of Section III for pruning a rectangle Λ. For example, pruning rule 1 can be extended to $\exists p' \in I^+\ (Dom_{p^*,\Lambda} \supseteq C_{p^*,p'})$, i.e., there exists an object $p' \in I^+$ such that the dominant region $C_{p^*,p'}$ is contained in the region $Dom_{p^*,\Lambda}$.

It is challenging to represent the shape of $Dom_{p^*,\Lambda}$ because the Appollonius circles formed by points in Λ can have different centers and radii.

We first present two lemmas that specify how dominant regions vary with respect to location and weight.

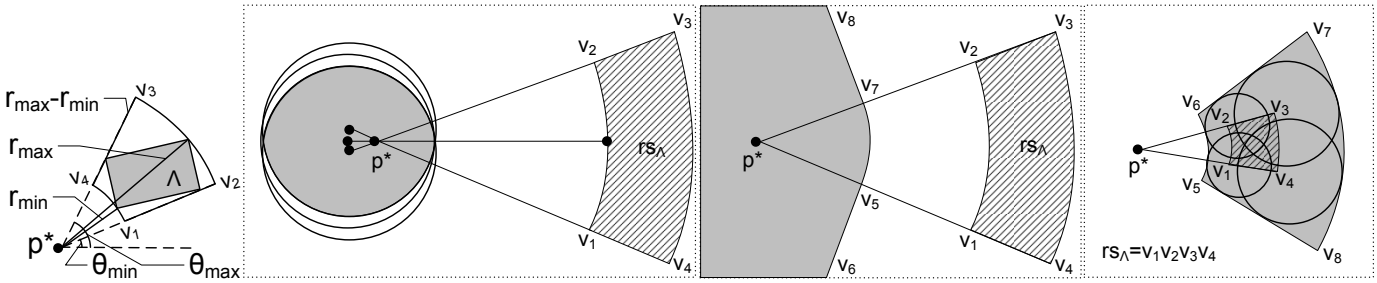
Lemma 6: **[Subset dominant region property at higher weight]** Given three objects p^*, p, and p', such that $w(p) \geq w(p')$ and $p.\lambda = p'.\lambda$ then $Dom_{p^*,p} \subseteq Dom_{p^*,p'}$.

Proof: By Definition 1, we have $\forall z \in Dom_{p^*,p}\ (\frac{\|p^*\ z\|}{w(p^*)} \leq \frac{\|p\ z\|}{w(p)})$. Since $p.\lambda = p'.\lambda$, we have $\|p\ z\| = \|p'\ z\|$. In addition, we have $w(p) \geq w(p')$. Thus $\forall z \in Dom_{p^*,p}(\frac{\|p^*\ z\|}{w(p^*)} \leq \frac{\|p'\ z\|}{w(p')})$. According to the definition of $Dom_{p^*,p'}$, we have $Dom_{p^*,p} \subseteq Dom_{p^*,p'}$. ■

Lemma 7: **[Subset dominant region property at closer distance]** Given three objects p^*, p, and p', such that $w(p^*) \leq w(p) = w(p')$ and $\|p^*\ p\| \leq \|p^*\ p'\|$, and such that the points p^*, p, and p' form a line. Then $Dom_{p^*,p} \subseteq Dom_{p^*,p'}$.

Proof: Without loss of generality, we translate and rotate the space such that $p^* = (0,0)$, $p = (x_1, 0)$, and $p' = (x_2, 0)$, where $0 \leq x_1 \leq x_2$. By Definition 1 and using $w(p) = w(p')$, we obtain the centers and radii of circles $C_{p^*,p}$ and $C_{p^*,p'}$ as: $o_{p^*,p} = (\frac{-w^2(p^*) \cdot x_1}{w^2(p) - w^2(p^*)}, 0)$, $o_{p^*,p'} = (\frac{-w^2(p^*) \cdot x_2}{w^2(p) - w^2(p^*)}, 0)$, $r_{p^*,p} = \frac{w(p^*) \cdot w(p) \cdot x_1}{w^2(p) - w^2(p^*)}$, and $r_{p^*,p'} = \frac{w(p^*) \cdot w(p) \cdot x_2}{w^2(p) - w^2(p^*)}$. By using $w(p^*) \leq w(p)$, we derive: $\|o_{p^*,p}\ o_{p^*,p'}\| = \frac{w^2(p^*) \cdot (x_2 - x_1)}{w^2(p) - w^2(p^*)} \leq \frac{w(p^*) \cdot w(p) \cdot (x_2 - x_1)}{w^2(p) - w^2(p^*)} = r_{p^*,p'} - r_{p^*,p}$. Thus, we have: $C_{p^*,p} \subseteq C_{p^*,p'}$, i.e., $Dom_{p^*,p} \subseteq Dom_{p^*,p'}$. ■

We proceed to illustrate the subset property of the above two lemmas. In Figure 7, we fix the location of p while varying its weight $w(p)$. When $w(p)$ decreases, the dominant region $Dom_{p^*,p}$ becomes a subset of the former dominant region. In Figure 8(a), we fix the weight of p while moving its location (e.g., p_3, p_2, p_1) towards p^* along the line segment $\overline{p\ p^*}$. When $\|p^*\ p\|$ decreases, $Dom_{p^*,p}$ becomes a subset of the former dominant region. The remaining case of Figure 8(b) will be discussed later.



(a) the minimum bounding ring-sector (b) $w^u(\Lambda) > w(p^*)$ (c) $w^u(\Lambda) = w(p^*)$ (d) $w^u(\Lambda) < w(p^*)$

Fig. 9. Representing the shape of the minimal dominant region $Dom_{p^*,\Lambda}$

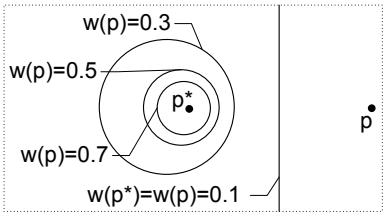
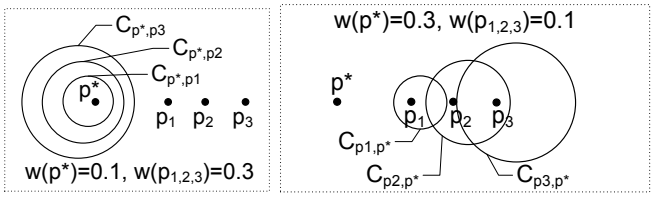


Fig. 7. Varying the weight of p



(a) case of $w(p^*) < w(p)$ (b) case of $w(p^*) > w(p)$

Fig. 8. Varying the spatial location of p

We then consider how to conservatively approximate $Dom_{p^*,\Lambda}$ using a tight subset expressed by simple geometric shapes. First, we propose to enclose Λ by a minimum bounding ring-sector, as shown in Definition 5. Figure 9(a) shows the minimum bounding sector rs_Λ of Λ, with respect to point p^*. It is bounded by the arcs and segments of four vertices v_1, v_2, v_3, and v_4.

Definition 5: Given a rectangle Λ, its *minimum bounding ring-sector* rs_Λ w.r.t. an object p^* is defined by a quadruple $\langle r_{min}, r_{max}, \theta_{min}, \theta_{max}\rangle$ where the minimum and maximum radii are $r_{min} = \|\Lambda\ p^*\|_{min}$ and $r_{max} = \|\Lambda\ p^*\|_{max}$, and θ_{min} and θ_{max} are the bounding angles w.r.t. p^*.

Lemma 8: **[Coverage of the minimum bounding ring-sector]** Given an object p^* and a rectangle Λ such that $w^u(\Lambda) \geq w(p^*)$, let rs_Λ be the minimum bounding ring-sector of Λ, and arc_s be the arc of rs_Λ closest to p^*. It holds that:

$$Dom_{p^*,\Lambda} \supseteq \bigcap_{p' \in arc_s,\ w(p')=w^u(\Lambda)} Dom_{p^*,p'}, \qquad (18)$$

where p' is any possible point on arc_s with weight $w^u(\Lambda)$.

Proof: For each object $p \in \Lambda$, there exists an object p' on arc_s, such that p^*, p', and p form a line, $\|p^*\ p'\| \leq \|p^*\ p\|$, and $w(p') \geq w(p)$. By Lemmas 6 and 7, we have $Dom_{p^*,p'} \subseteq Dom_{p^*,p}$. Hence, we have $Dom_{p^*,\Lambda} \supseteq \bigcap_{p' \in arc_s} Dom_{p^*,p'}$. ■

We then study the shape of $Dom_{p^*,\Lambda}$ for three cases.

The shape of $Dom_{p^*,\Lambda}$ when $w^u(\Lambda) = w(p^*)$.
By Lemma 8, the region Dom_{p^*,arc_s} is a subset of $Dom_{p^*,\Lambda}$. The boundary of Dom_{p^*,arc_s} (the shaded region in Figure 9(c)) is formed by the perpendicular bisectors of p^* and each point on the arc arc_s. It can be represented as the arc $(\widehat{v_5 v_7})$ with center p^*, radius $\frac{1}{2} \cdot r_{min}$ and angle $\angle v_1 p^* v_2$, and the halfplanes formed by the arc's tangent lines $v_5 v_6$ and $v_7 v_8$.

The shape of $Dom_{p^*,\Lambda}$ when $w^u(\Lambda) > w(p^*)$.
By Lemma 8, the region Dom_{p^*,arc_s} is a subset of $Dom_{p^*,\Lambda}$. Observe that any location z on circle $C_{p^*,p}$ can be represented in the polar system of p^* as a magnitude $\|p^*\ z\|$ and an angle θ. Thus, we have: $z = (p^*_x + \|p^*\ z\| \cdot \cos\theta, p^*_y + \|p^*\ z\| \cdot \sin\theta)$. Since z falls onto the circle $C_{p^*,p}$, it satisfies:

$$(p^*_x + \|p^*\ z\| \cdot \cos\theta - o_x)^2 + (p^*_y + \|p^*\ z\| \cdot \sin\theta - o_y)^2 = r^2_{p^*,p}$$

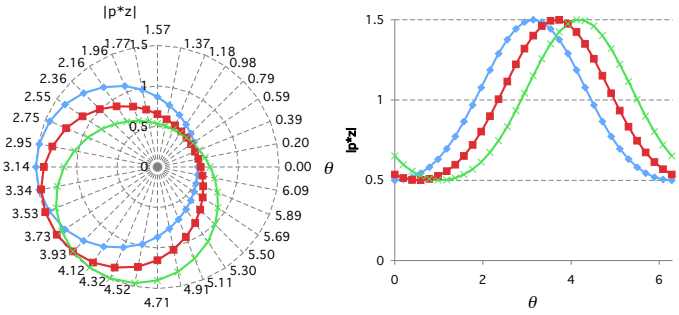
Using $F_\alpha = p^*_x - o_x$, $F_\beta = p^*_y - o_y$ and solving $\|p^*\ z\|$ in the above equation, we obtain Equation 19, which expresses the magnitude $\|p^*\ z\|$ as a function of θ.

$$\|p^*\ z\| = \sqrt{F^2_\gamma - F^2_\alpha - F^2_\beta + r^2_{p^*,p}} - F_\gamma \qquad (19)$$

where $F_\alpha = p^*_x - o_x$, $F_\beta = p^*_y - o_y$, and $F_\gamma = F_\alpha \cdot \cos\theta + F_\beta \cdot \sin\theta$.

The dominant region of p^* over arc_s, i.e., Dom_{p^*,arc_s}, is the intersection of the Apollonius circle $C(p^*, p)$ for each point on arc_s. It is the shaded region in Figure 9(b). Consider the endpoint v_1 of arc_s in the figure. By rotating circle C_{p^*,v_1} with the angle of arc_s, i.e., replacing θ by $\theta+\delta$ in Equation 19, and taking the intersection, we obtain Dom_{p^*,arc_s} that is a subset of $Dom_{p^*,\Lambda}$.

Specifically, we capture the region Dom_{p^*,arc_s} by a subset f-sided polygon as described next. The parameter f decides the tightness of the approximation. In the example of Figure 10, we have: $r_{p^*,p} = 1$, $(o_x, o_y) = (0, 0)$, and $(p^*_x, p^*_y) = (0, 0.5)$. Figure 10(b) plots the value of $\|p^*\ z\|$ with respect to θ (see Equation 19). The $\times$-curve is obtained by shifting the $\diamond$-curve to the left by the angle $\delta = \pi/3$. For each angle $\theta = \frac{2\pi i}{f}$ where $i \in [1, f]$, we compute the minimum $\|p^*\ z\|$ and obtain its (x, y) coordinates (see Figure 10(a)). Then we link these f points to form a polygon, which is guaranteed to be a subset of Dom_{p^*,arc_s}.



(a) polar view from p^* (b) angle-radius plot

Fig. 10. Expressing $\|p^*\ z\|$ as a function of θ

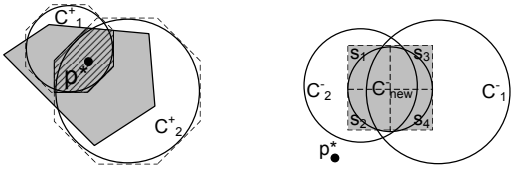
The shape of $Dom_{p^*,\Lambda}$ when $w^u(\Lambda) < w(p^*)$. In this case, we have: $Dom_{p^*,\Lambda} = \bigcap_{p\in\Lambda} Dom_{p^*,p} = \bigcap_{p\in\Lambda}(\mathcal{U} - C_{p,p^*}) = \mathcal{U} - \bigcup_{p\in\Lambda} C_{p,p^*}$. In order to obtain a subset representation of $Dom_{p^*,\Lambda}$, we need to derive a superset representation of the region $\bigcup_{p\in\Lambda} C_{p,p^*}$.

Let p be a point in Λ and consider the example of Figure 8(b) where $w(p^*) > w(p)$. We fix the weight of p while moving its location (e.g., p_1, p_2, p_3) away from p^* along the line segment $\overline{p^*\ p}$. The union of these circles can be represented by a ring-sector. Figure 9(d) illustrates how to represent the superset of the region $\bigcup_{p\in rs_\Lambda} C_{p,p^*}$, which can also be approximated by a ring-sector.

B. Optimizations

Upgraded Voronoi Cell—Enhancing Rules 1, 2, 4, 5, and 6. The Voronoi cell optimization (presented in Section III-C) employs a convex polygon Φ to represent the temporary safe zone defined by the set I^o. We now upgrade this technique by using the set I^+ with a parameter f that enables a trade-off between pruning power and computational overhead. Each time we insert an object p into I^+, we derive the circle $C_{p^*,p}$ and compute an f-sided polygon (say, G) such that it encloses $C_{p^*,p}$. Next, we update Φ to the intersection region $\Phi \cap G$. This upgraded Φ enhances pruning rules 1, 2, 4, 5, and 6. Correctness is ensured because G is a superset of $C_{p^*,p}$, so no pruned object can contribute to the actual safe zone.

In the example in Figure 11(a), the shaded region is the temporary safe zone Φ formed by objects in I^o. The circles C_1^+ and C_2^+ are formed by objects in I^+. After intersecting Φ with the f-sided polygons of the two circles, Φ becomes the striped polygon, which is much smaller than the initial Φ and both circles.



(a) upgraded Voronoi cell, $f = 8$ (b) reducing I^-, $level = 2$

Fig. 11. Optimization techniques for the MSK algorithm

Recursive Refinement of I^-—Enhancing Rule 3. We finally enhance pruning rule 3. Consider the example in Figure 11(b) where the set I^- has two objects, forming two circles C_1^- and C_2^- with the result object p^*, respectively. Now we encounter a new object $p_{new} \in \mathcal{D}^-$ and define the circle C_{new}^- (w.r.t. p^*). Rule 3 cannot prune away p_{new} because neither C_1^- nor C_2^- contains C_{new}^-.

We propose a recursive refinement technique to remove an object p_{new} whose circle is covered by the union of the circles in I^-. This can significantly reduce the size of I^-. The idea is to partition the circle C_{new}^- into four squares, say s_1, s_2, s_3, and s_4, as shown in Figure 11(b). Then we check whether each square is covered by some circle, e.g., s_1 and s_2 are covered by C_2^-; s_3 and s_4 are covered by C_1^-. If each square s_i satisfies the condition then circle C_{new}^- is covered by the union of circles in I^-, and thus C_{new}^- is pruned. In case a square s_i does not intersect any circle of I^- then C_{new}^- cannot be pruned. When a square s_i only partially intersects some circle of I^-, we apply the above process recursively on the square s_i until reaching a pre-defined maximum recursion level l_{max}. In practice, it is sufficient to use a small constant for l_{max}; we study its effect empirically.

C. MSK-uvr Algorithm

Algorithm 2 shows the pseudo code of the advanced algorithm for computing the safe zone $\Upsilon(p^*)$ of the top-1 result p^*. It differs from Algorithm 1 in several respects. First, it is able to prune an unqualified node (or a subtree) in Lines 20–21 by using the techniques developed in Section IV-A. Second, it applies the upgraded Voronoi cell optimization (Line 12) for enhancing pruning rules 1, 2, 4, 5, and 6. Third, it performs the recursive refinement (Line 18) in order to reduce the size of the set I^-.

Extension to Arbitrary k. Recall that the order-k MW-Voronoi cell $\Upsilon^k(\mathcal{RS})$ is the safe zone of the result set $\mathcal{RS}$ (see Equation 16). Here, we extend Algorithm 2 to arbitrary k, so that it visits each IR-tree node at most once. Let result set $\mathcal{RS}$ be $\{p_1^*, p_1^*, \cdots, p_k^*\}$. The key of an entry e' (in Line 24) is computed as $\min_{j\in[1,k]} bord_{min}(p_j^*, e'.\Lambda)$. When we encounter an object e in the IR-tree, we compute its dominant region for each result p_i^*. If any of them does not intersect the polygon Φ then we prune the object e. Otherwise, we update Φ by its intersection with all such dominant regions. When we visit a non-leaf entry e, we also compute the dominant region for each result p_i^*. If any of them does not intersect Φ then we prune the subtree of e.

V. Empirical Study

A. Experimental Setup

We use four real data sets, each containing objects with a point location and a set of keywords, for studying the robustness and performance of the proposals. Data set HOTEL contains US hotels (www.allstays.com); GN is obtained from the U.S. Board on Geographic Names (geonames.usgs.gov); and EURO and USCAN contain points of interest (e.g., ATMs,

Algorithm 2 MSK-uvr (Tree root $root$, Query q, Result p^*)

System parameters: f, l_{max} ▷ used for optimizations
$I^+ \leftarrow$ new set; $I^- \leftarrow$ new set;
$\Phi \leftarrow$ the space domain $\mathcal{U}$; ▷ a convex polygon
$H \leftarrow$ new min-heap;
$H.enheap(root, 0)$;
while H is **not** empty **do**
  $e \leftarrow H.deheap()$;
  if e is an object **and** $e \neq p^*$ **then**
    if $w(e) > w(p^*)$ **then**
      if $\Phi \not\subseteq C_{p^*,e}$ **then** ▷ enhanced rule 1
        insert e into I^+;
        $G \leftarrow$ an f-sided polygon that contains $C_{p^*,e}$;
        $\Phi \leftarrow \Phi \cap G$; ▷ upgraded vor. cell optimization
    else if $w(e) = w(p^*)$ **then**
      if $\Phi \not\subseteq \perp_{p^*,e}$ **then** ▷ enhanced rules 5,6
        $\Phi \leftarrow \Phi \cap \perp_{p^*,e}$;
    else if $w(e) < w(p^*)$ **then**
      if $\Phi \cap C_{e,p^*} \neq \emptyset$ **then** ▷ enhanced rules 2,4
        Recur_Refine(e, I^-, l_{max}); ▷ enhanced rule 3
  else ▷ e points to a child node
    if $(w^u(e) \geq w(p^*)$ **and** $\Phi \subseteq Dom_{p^*,e})$ **or**
      $(w^u(e) < w(p^*)$ **and** $\Phi \cap Dom_{e,p^*} = \emptyset)$ **then**
      continue; ▷ pruning a child node
    read the child node CN of e;
    for each entry e' in node CN **do**
      $H.enheap(e', bord_{min}(p^*, e'.\Lambda))$;
return the set $I^+ \cup I^-$ and the polygon Φ;

hotels, stores) in Europe, and in USA and Canada, respectively (www.pocketgpsworld.com). Table II offers additional detail. We normalize object locations to fit a square domain with side-length $10,000$ meters.

TABLE II
DATA SET STATISTICS

data set	# of objects	# of distinct words	average # of words per object
HOTEL	21,021	602	3
GN	1,868,821	222,407	4
EURO	162,033	35,315	18
USCAN	13,977	5,106	15

Brinkhoff's generator [20] is used to generate a trajectory of 100 points, using one location acquisition per timestamp (second). Each experiment has 100 moving queries with such trajectories. The keyword set of each query is generated randomly within the word domain of the data sets used.

The TF function is used to measure the relevancy between two keyword sets. We generally report the average value per query per timestamp of the following: (i) server-side elapsed time, (ii) client-side elapsed time, (iii) communication cost (number of influence objects transferred), and (iv) communication frequency (the probability of sending a request to the server).

We study our proposed solutions: IBD and MSK. To observe the effects of the optimizations on MSK, we consider three variants: MSK-v (using Voronoi cells), MSK-uv (using upgraded Voronoi cells), and MSK-uvr (using upgraded Voronoi cells and I^- refinement).

By default, we set the number k of results to 1, the number of query keywords to 3, and the client speed to 10 m/s. For the optimizations of MSK, we set f (the number of sides per polygon) to 16, and l_{max} (the level of recursive pruning) to 5. We used disk-based IR-trees with the page size fixed at 8 KBytes. All the algorithms are implemented in Java and run on a Linux server with one processor (Pentium-R Dual-Core CPU E5200 @ 2.50GHz) and 4GB memory.

B. Performance of Continuous MkSK Queries

We study the proposed methods under varying settings.

Results for the Real Data Sets.

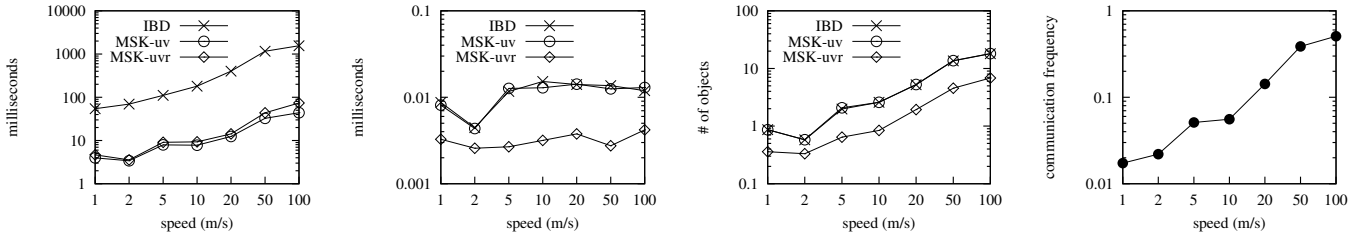
Table III reports the average server elapsed time, server I/O cost, client elapsed time, communication cost, and communication frequency for the four methods. Since the MSK variants can prune objects at higher levels in the IR-tree, they beat IBD significantly in terms of server elapsed time and I/O cost. MSK-uv benefits from upgraded Voronoi cells that offer tighter bounds than do the Voronoi cells used in MSK-v and IBD. MSK-uv thus achieves some 20% lower cost than MSK-v. In contrast, MSK-uvr's recursive refinement of I^- incurs an overhead, and so its server elapsed time is slightly higher than that of MSK-uv. On the positive side, MSK-uvr is able to shrink I^- significantly, reducing the communication cost and also client elapsed time. All methods use safe zones and have the same low communication frequency. As the methods perform consistently across the different data sets, we use EURO as the default data set in subsequent experiments. We drop MSK-v as it is always worse than MSK-uv.

TABLE III
PERFORMANCE PER TIMESTAMP ON REAL DATA SETS

Data Set		HOTEL	GN	EURO	USCAN
Server elapsed time (milliseconds)	IBD	2.673	7285.350	182.606	1.454
	MSK-v	0.549	79.602	9.350	0.534
	MSK-uv	0.531	54.651	7.897	0.523
	MSK-uvr	0.582	54.490	9.215	0.554
Server I/O cost (page accesses)	IBD	9.288	1413.180	163.435	6.159
	MSK-v	0.503	16.006	8.014	0.622
	MSK-uv	0.459	10.563	7.340	0.568
	MSK-uvr	0.459	10.563	7.340	0.568
Client elapsed time (milliseconds)	IBD	0.002	0.012	0.014	0.002
	MSK-v	0.002	0.009	0.016	0.002
	MSK-uv	0.004	0.010	0.015	0.003
	MSK-uvr	0.003	0.003	0.003	0.001
Communication cost (objects)	IBD	0.611	1.620	2.589	0.332
	MSK-v	0.606	1.587	2.575	0.332
	MSK-uv	0.623	1.584	2.583	0.338
	MSK-uvr	0.489	0.809	0.841	0.279
Communication frequency	All methods	0.036	0.078	0.056	0.025

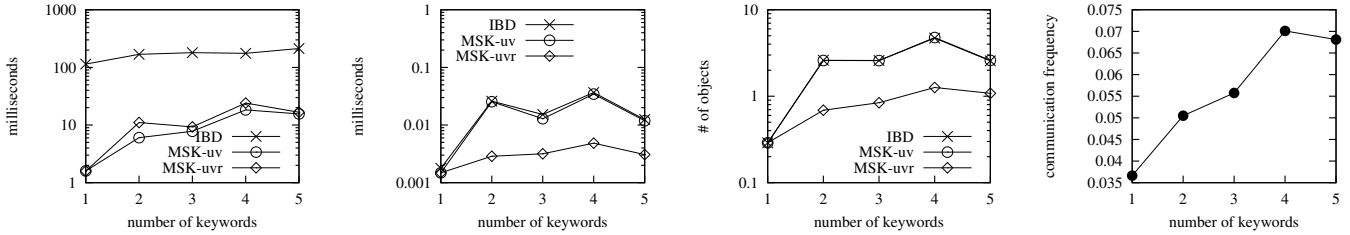
Varying the Speed of Moving Queries.

This experiment compares IBD, MSK-uv, and MSK-uvr when varying the speed of the moving queries. As the speed increases, the server elapsed time (Figure 12(a)), the client elapsed time (Figure 12(b)), and the communication cost (Figure 12(c)) exhibit an increasing trend. The query leaves the safe zone faster at a higher speed. Indeed, Figure 12(d) shows that the communication frequency increases with increasing



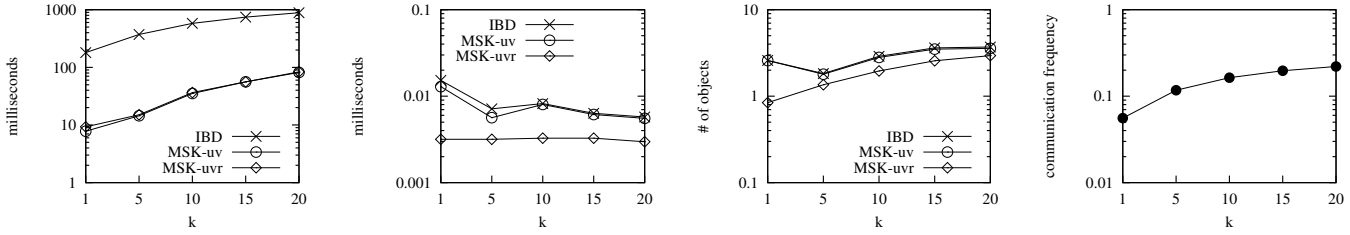
(a) server elapsed time (b) client elapsed time (c) communication cost (d) communication frequency

Fig. 12. Performance per timestamp, varying speed, on EURO



(a) server elapsed time (b) client elapsed time (c) communication cost (d) communication frequency

Fig. 13. Performance per timestamp, varying the number of query keywords, on EURO



(a) server elapsed time (b) client elapsed time (c) communication cost (d) communication frequency

Fig. 14. Performance per timestamp, varying k, on EURO

speed, leading to more elapsed time and communication costs. MSK-uv and MSK-uvr outperform IBD significantly.

Varying the Number of Query Keywords.
We consider again IBD, MSK-uv, and MSK-uvr. As expected, the MSK variants outperform IBD significantly in terms of server elapsed time as shown in Figure 13(a). Figures 13(b) and 13(c) show that the communication cost is proportional to the client elapsed time, with MSK-uvr being the winner due to its recursive refinement of I^-. Figure 13(d) shows that the communication frequency increases slightly with the number of keywords. This is because large query keyword sets yield smaller safe zones.

Varying k.
Figure 14(a) shows that MSK-uv and MSK-uvr have much lower server elapsed time than IBD. Figures 14(b) and 14(c) show that MSK-uvr has the lowest client elapsed time and communication cost. The communication frequency increases slightly with k increases (see Figure 14(d)), as a larger k yields smaller safe zones.

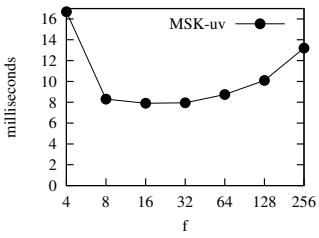
Effect of f on the Upgraded Voronoi Cell Optimization.
Figure 15 shows the elapsed time and I/O cost of MSK-uv when varying the number f of edges used for circles in the upgraded Voronoi cell optimization. As f increases, it more tightly represents a circle by a (superset) f-sided polygon. This makes the temporary safe zone tighter, enabling more tree nodes to be pruned and thus reducing the I/O cost. The elapsed time decreases when $f < 32$, but increases when $f > 32$. A larger f leads to higher computational overhead, which counteracts the savings from pruned nodes. The elapsed time is low for a wide range of f (8–128).

Effect of l_{max} on the Recursive Refinement.
We end by observing the effect of varying the maximum recursion level l_{max} on the recursive refinement. Figure 16 reports the elapsed time and the size of I^-. As l_{max} increases, partition squares become smaller, making it easier to prune an object (in D^-) whose dominant region is covered by the union of circles formed by objects in I^-. Even at a high l_{max} (e.g., 8), MSK-uvr only incurs slightly more elapsed time.

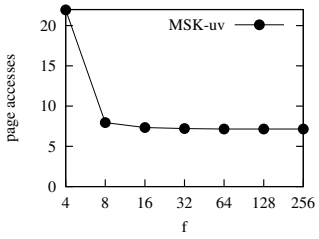
In summary, MSK-uv achieves the lowest server elapsed time and MSK-uvr is the best in terms of communication cost.

VI. Related Work

Spatial Keyword Queries.
A spatial keyword query retrieves the best object(s) with respect to a given location and a set of keywords. Efficient implementation of such queries, with varying semantics, has

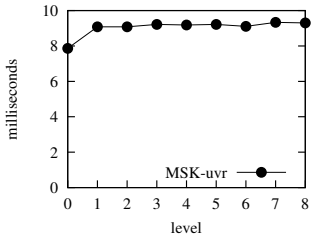


(a) server elapsed time

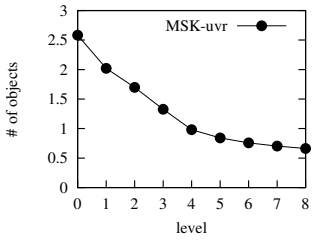


(b) server I/O cost

Fig. 15. Performance per timestamp, varying f, on EURO



(a) server elapsed time



(b) communication cost

Fig. 16. Performance per timestamp, varying l_{max}, on EURO

been studied. Zhou et al. [6] build a separate R*-tree on the objects for each distinct keyword. The IR^2-tree [3] is an R-tree augmented with signatures, and the IR-tree [2] is an R-tree augmented with inverted files. The former is applicable when the keywords serve as a Boolean filter; the latter supports the ranking of objects based on a weighted sum of spatial distance and text relevancy. The recently proposed mCK query retrieves m objects within a minimum diameter that match given keywords. The bR*-tree and the virtual bR*-tree [4], [5], which augment each node with a bitmap and MBRs for keywords, are used for computing this query

None of the above techniques take into account continuous queries, and our focus, the computation of safe zones for query results, has not been studied.

Moving Queries.

In our setting, the query moves continuously while the data objects (e.g., hotels, restaurants) are stationary. We consider related work for this setting.

Given the known future movement of the user, a time-parameterized query [21] retrieves the current result along with a time interval indicating its validity. When the user's movement is not known in advance, the safe zone approach [7]–[9] is more suitable and may reduce the client-server communication cost significantly. The server returns a query result to the user along with a safe zone. It is guaranteed that the result remains unchanged as long as the user remains in the safe zone. When leaving the safe zone, the user requests a new result and safe zone from the server. The safe zone of a kNN query can be captured by an order-k Voronoi cell [7], or a V*-diagram [8]. The safe zone of a moving circular range query can be captured by a set of objects that affect the safe zone [9].

None of the above techniques consider the text relevancy of the data objects. In our work, the safe zone is an MW-Voronoi region, whose efficient computation has not been studied.

VII. Conclusions

This paper studies the moving top-k spatial keyword query and utilizes the concept of the safe zone to save computation and communication costs. We develop two solutions for computing a safe zone: (i) an early stop algorithm IBD and (ii) an advanced algorithm MSK-uvr that prunes subtrees of objects that do not contribute to the safe zone and applies two optimizations to further reduce the search space and communication cost. Empirical studies on real data sets demonstrate that MSK-uv (a variant of MSK-uvr) has the lowest server elapsed time and that MSK-uvr has the lowest communication cost. In future work, it is of interest to examine the processing of moving top-k spatial keyword queries over road networks.

References

[1] Y.-Y. Chen, T. Suel, and A. Markowetz, "Efficient query processing in geographic web search engines," in *SIGMOD*, pp. 277–288, 2006.

[2] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial web objects," in *PVLDB*, pp. 337–348, 2009.

[3] I. De Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *ICDE*, pp. 656–665, 2008.

[4] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa, "Keyword search in spatial databases: Towards searching by document," in *ICDE*, pp. 688–699, 2009.

[5] D. Zhang, B. C. Ooi, and A. Tung, "Locating mapped resources in web 2.0," in *ICDE*, pp. 521–532, 2010.

[6] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma, "Hybrid index structures for location-based web search," in *CIKM*, pp. 155–162, 2005.

[7] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, "Location-based spatial queries," in *SIGMOD*, pp. 443–454, 2003.

[8] S. Nutanong, R. Zhang, E. Tanin, and L. Kulik, "The V*-Diagram: a query-dependent approach to moving kNN queries," in *PVLDB*, pp. 1095–1106, 2008.

[9] M. A. Cheema, L. Brankovic, X. Lin, W. Zhang, and W. Wan, "Multi-guarded safe zone: An effective technique to monitor moving circular range queries," in *ICDE*, pp. 189–200, 2010.

[10] A. Okabe, B. Boots, K. Sugihara, and S. Chiu, *Spatial Tessellations, Concepts and Applications of Voronoi Diagrams*. Wiley, 2000.

[11] R. Gambini, D. L. Huff, and G. F. Jenks, "Geometric properties of market areas," *Regional Science*, vol. 20, no. 1, pp. 85–92, 1968.

[12] D. L. Huff, "The delineation of a national system of planning regions on the basis of urban spheres of influence," *Regional Studies*, vol. 7, no. 3, pp. 323–329, 1973.

[13] B. N. Boots, "Patterns of urban settlements revisited," *The Professional Geographer*, vol. 27, no. 4, pp. 426–431, 1975.

[14] L. Mu, "Polygon characterization with the multiplicatively weighted Voronoi diagram," *The Professional Geographer*, vol. 56, no. 2, pp. 223–239, 2004.

[15] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in *SIGIR*, pp. 275–281, 1998.

[16] F. Aurenhammer and H. Edelsbrunner, "An optimal algorithm for constructing the weighted Voronoi diagram in the plane," *Pattern Recognition*, vol. 17, no. 2, pp. 51–57, 1984.

[17] C. V. Durell, *Modern geometry: the straight line and circle*. Macmillan, 1961.

[18] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *SIGMOD*, pp. 47–57, 1984.

[19] J. Zobel and A. Moffat, "Inverted files for text search engines," in *ACM Comput. Surv.*, vol. 38, no. 2, p. 6, 2006.

[20] T. Brinkhoff, "A framework for generating network-based moving objects," *Geoinformatica*, vol. 6, no. 2, pp. 153–180, 2002.

[21] Y. Tao and D. Papadias, "Time-parameterized queries in spatio-temporal databases," in *SIGMOD*, pp. 334–345, 2002.

[22] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1986.