Moving Spatial Keyword Queries: Formulation, Methods, and Analysis

DINGMING WU, Hong Kong Baptist University
MAN LUNG YIU, Hong Kong Polytechnic University
CHRISTIAN S. JENSEN, Aarhus University

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 text 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 the safe zone associated with a result. 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 server-side computation 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. We present techniques that aim to compute the next safe zone efficiently, and we present two types of conservative safe zones that aim to reduce the communication cost. Empirical studies with real data suggest that the proposals are efficient.

To understand the effectiveness of the proposed safe zones, we study analytically the expected area of a safe zone, which indicates on average for how long a safe zone remains valid, and we study the expected number of influence objects needed to define a safe zone, which gives an estimate of the average communication cost. The analytical modeling is validated through empirical studies.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms: Algorithms

Additional Key Words and Phrases: Moving query, multiplicatively weighted Voronoi diagram, spatial keyword query, safe zone, Voronoi diagram

1. 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 [Chen et al. 2006; Felipe et al. 2008; Cong et al. 2009; Zhang et al. 2009; Zhang et al. 2010; Cao et al. 2012; Chen et al. 2012; Wu et al. 2012]. Taking a location and a set of keywords as arguments, such queries return relevant content that matches the arguments. We introduce the weighted distance as the ranking function. Assuming a data set $\mathcal{D}$ of objects p with weight $tr_{q.\psi}(p.\psi)$, i.e., text relevancy, the weighted distance between p and a query q is defined as

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

Author's addresses: Dingming Wu, Department of Computer Science, Hong Kong Baptist University, Hong Kong; email: dmwu@comp.hkbu.edu.hk; Man Lung Yiu, Department of Computing, Hong Kong Polytechnic University, Hong Kong; email: csmlyiu@comp.polyu.edu.hk; Christian S. Jensen, Department of Computer Science, Aarhus University, Denmark; email: csj@cs.au.dk.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

where $\|q.\lambda\ p.\lambda\|$ denotes their Euclidean distance. This distance function has been used extensively in geo-sciences [Gambini et al. 1968; Huff 1973]; they represent an urban place as a point and its weight as the population (importance) of that place. A key strength of this function is that the different measurement units of the weights 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 2.1 in detail). 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 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, in order 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.

A straightforward solution to the MkSK query is to periodically invoke an existing snapshot spatial keyword query processing technique (e.g., [Zhou et al. 2005; Felipe et al. 2008; Cong et al. 2009; Zhang et al. 2009; Wu et al. 2012]). To illustrate, the dashed curve in Figure 2(b) shows the trajectory of a query. The correct top-1 result changes from p_2 to p_4 to p_2 and to p_5 as the query moves. Recomputing the query at times $t_1, \ldots, t_4$ has the effect that the result is unnecessarily recomputed (at time t_3), which is undesirable due to the increased computation and communication costs. In addition, the result is at times incorrect (before times t_1, t_2, and t_4). In general, no single inter-computation period exists that avoids unnecessary computations while also offering correctness.

Another solution to the MkSK query is the buffering approach [Song and Roussopoulos 2001]. A central server retrieves the user's $k + \Delta k$ nearest neighbors and uses them to derive a buffer region for the user. While moving within the buffer region, the user's k nearest neighbors can be kept up to date by the user using only the $k + \Delta k$ nearest neighbors. This method only considers the distances between objects and the query. It is non-trivial to adopt this approach to solve our problem, since we take into account of both text relevancies and distances so that the buffer region based on the Euclidean distance is not applicable.

State-of-the-art proposals for moving queries [Zhang et al. 2003; Nutanong et al. 2008; Cheema et al. 2010] adopt a standard client-server architecture along with *safe zones*. The safe zone of a query is a region containing the query location. As long as the query remains in this region, the result of the query does not change, and there is no need to request a new result from the server. The client is able to monitor whether it is inside the safe zone or moves out of it. The client sends a new request to the server only when it leaves the safe zone. Then the server computes and returns a new result and corresponding safe zone to the client. The advantage of safe zones is that the client is able to request a new result exactly when necessary to always provide a correct result, which makes it possible to avoid unnecessary computation and communication. We adopt the same architecture. Figure 1 illustrates the steps involved in the query processing. The service provider maintains a dataset of spatial web objects. The client repeatedly obtains its current location, e.g., using GPS or a network-based positioning service (Step 1a), and checks whether the current location is in its current safe zone (Step 1b). As long as the client stays inside the safe zone, the result is guaranteed to remain correct. When the client exits the safe zone, the client issues the query q with the same query keywords as before, but with the new location (Step 2). Having received the query, the service provider first computes the result of the query (Step 3) and then compute the safe zone of the result (Step 4). Finally, the service provider sends the result and its safe zone to the client (Step 5), and the process starts with with Steps 1a and 1b.

To the best of our knowledge, this paper is the first study of moving spatial keyword queries. Voronoi cells have been used as safe zones for nearest neighbor queries [Zhang et al. 2003]. Figure 2(a) shows the Voronoi diagram of a set of 5 data points. The Voronoi diagram partitions the space into several cells. The Voronoi cell that a data point belongs to has the property that any point

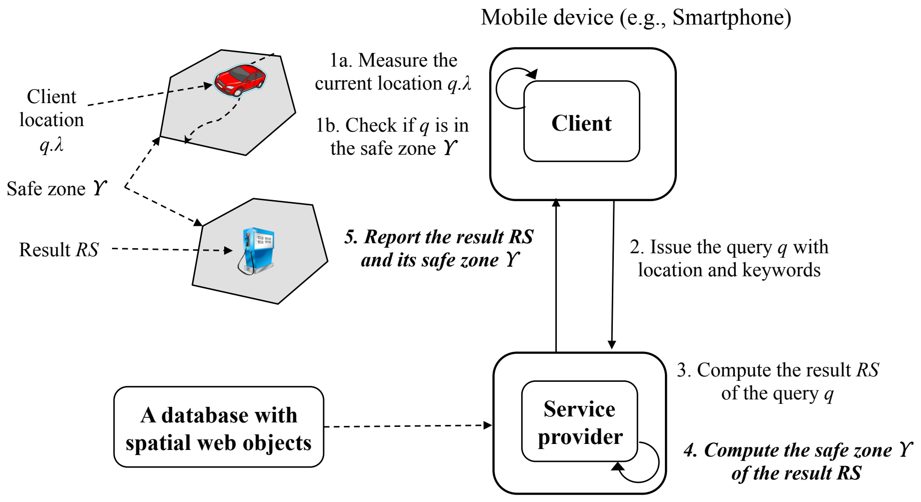

Mobile device (e.g., Smartphone)
Client location $q.\lambda$
Safe zone Υ
Result RS
1a. Measure the current location $q.\lambda$
1b. Check if q is in the safe zone Υ
Client
5. Report the result RS and its safe zone Υ
2. Issue the query q with location and keywords
A database with spatial web objects
Service provider
3. Compute the result RS of the query q
4. Compute the safe zone Υ of the result RS

Fig. 1. Client-Server Architecture

in that cell has the data point as its nearest neighbor. Thus, the Voronoi cell of a data point is the safe zone for the the nearest neighbor query that returns the data point as its result. As an example, consider the query q located inside the cell of p_4. This query has p_4 as its nearest neighbor result, and the safe zone of the result is the containing Voronoi cell. However, Voronoi diagrams consider only spatial distance and do not account for text relevancy.

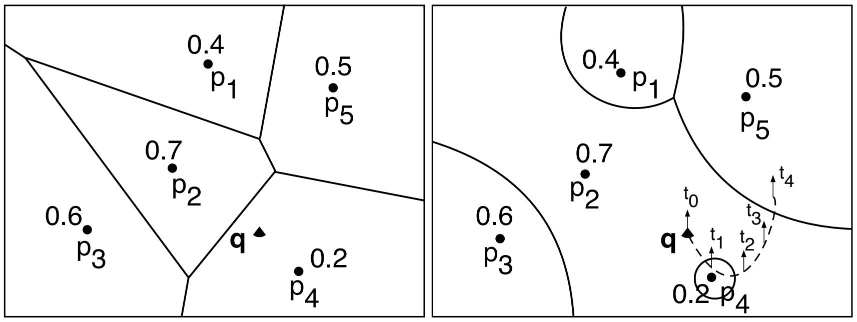
For the MkSK query, a query-keyword dependent relevancy score is assigned to each data point. To take the text relevancy into account, we use a distance function (Equation 1) that weighs the spatial distance with the textual relevancy. Voronoi cells do not work properly as safe zones for such queries. According to Okabe et al. [2000], a so-called multiplicatively weighted Voronoi (MW-Voronoi) diagram captures the correct safe zones when the weighted distance function in Equation 1 is used. The MW-Voronoi cell of a data point defines its influence region based on its weight. 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.

The *safe zone* of an object is its MW-Voronoi cell—all locations in the safe zone have smaller weighted distance to the object than to any other object. Figure 2(b) illustrates the MW-Voronoi diagram of the same 5 data objects as in Figure 2(a). Now each object has a weight. These weights are the text relevancies between the objects and the query, known only when the query submitted. In other words, different queries produce different MW-Voronoi diagrams on the same dataset. In this example, query q is inside the safe zone of p_2, i.e., p_2 is the top-1 object, according to Equation 1, while in Figure 2(a) p_4 is the nearest neighbor according to the Euclidean distance.

MW-Voronoi diagrams have been applied extensively in geographic, market, and urban analysis. For instance, Gambini et al. [1968] examine the geometric properties of market areas of firms serving hoursehold consumers, where a MW-Voronoi cell of a firm is determined by its distance from consumers and its consumer attraction. In the U.S. national systems of planning regions [Huff 1973], the regions that are delineated represent the spheres of influence of urban places.

Existing methods for constructing MW-Voronoi diagrams involve a full scan of the data [Mu 2004]. 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.

Previous work [Wu et al. 2011] presents two algorithms for computing safe zones using a tree-based index [Zhou et al. 2005; Felipe et al. 2008; Cong et al. 2009; Zhang et al. 2009; Li et al. 2011; Wu et al. 2012] over a spatial keyword data set, so that only a small fraction of the data objects are



(a) Voronoi Diagram (b) MW-Voronoi Diagram

Fig. 2. Comparisons between the Traditional Voronoi Diagram and the MW-Voronoi Diagram

accessed during computation. The objectives are to optimize the server-side computation as well as the client/server communication. This work formalizes the continuously moving top-k spatial keyword query problem and presents an early stop algorithm (IBD) for computing a safe zone and an advanced algorithm (MSK-uvr) that enables pruning of subtrees of objects that do not contribute to defining a safe zone, thus improving efficiency.

This paper substantially revises and extends this previous work. The additional contributions of the paper are summarized as follows.

(1) We study analytically the expected area of a safe zone in Section 5, which indicates how long a safe zone remains valid; and we also study analytically the expected number of so-called influence objects that contribute to defining a safe zone, which offers insight into the average communication cost.
(2) We develop techniques that aim to efficiently compute the next safe zone, i.e., to reduce the server-side computation (corresponding to Step 4 in Figure 1) in Section 6. These utilize the previous safe zone and results to construct a smaller temporary safe zone, and they also reuse the weights of entries that have been obtained when computing the previous safe zone.
(3) We introduce two types of conservative safe zones, along with parameter-free variants, that aim to reduce the cost of the communication between the client and the server (corresponding to Step 5 in Figure 1) in Section 7. Since a conservative safe zone is a subset of the real safe zone, the communication frequency may become higher. There exists a tradeoff between the communication cost per round and communication frequency. However, as we show in the experiments, lower total communication costs can be achieved by taking advantage of conservative safe zones.
(4) We offer empirically based insight into the performance of the proposed techniques.

The rest of the paper is organized as follows. We first present the problem setting and provide background knowledge in Section 2. Then we present the early stop algorithm (IBD) in Section 3 and the advanced algorithm (MSK-uvr) in Section 4. Section 5 studies analytically the expected area of a safe zone as well as the expected number of influence objects that define a safe zone. Techniques for efficiently computing the next safe zone are the subject of Section 6. The concept of conservative safe zone is studied in Section 7 with the objective of to reducing the communication cost. Next we study the performance of our proposals on real data sets in Section 8. We discuss related work in Section 9 and conclude in Section 10. The paper is accompanied by an electronic appendix.

2. 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.

2.1. 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 [Salton and McGill 1986].

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 [Aurenhammer and Edelsbrunner 1984] in Equation 2 as our ranking function, since it matches the semantics of the query.

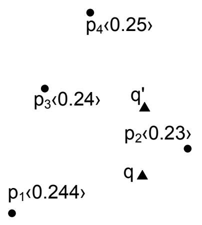
$$rank_q(p) = \frac{\|q.\lambda \; p.\lambda\|}{tr_{q.\psi}(p.\psi)}, \tag{2}$$

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:

$$\forall p \in \mathcal{RS}(\; \forall p' \in \mathcal{D} - \mathcal{RS}(\; rank_q(p) \leq rank_q(p'))) \tag{3}$$

Figure 3(a) shows the locations of a set of objects $\mathcal{D} = \{p_1, p_2, p_3, p_4\}$. Figure 3(b) shows the word frequencies for each object. Let the query q shown in Figure 3(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.



(a) locations

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

(b) descriptions

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

(c) Euclidean distances

Fig. 3. 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 and server communication cost.

2.2. Multiplicatively Weighted Voronoi Diagram

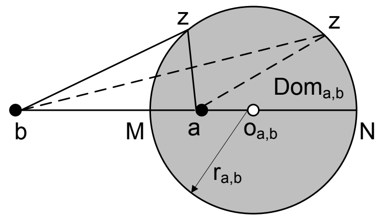
Our approach to computing the moving top-k spatial keyword query utilizes the so-called multiplicatively weighted Voronoi (MW-Voronoi) diagram [Okabe et al. 2000]. Here, we recall the definitions of MW-Voronoi diagrams and regions.

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 $d_w(z,a) = \|z\ a\|/w(a)$, where $\|z\ a\|$ is the Euclidean distance between z and a.

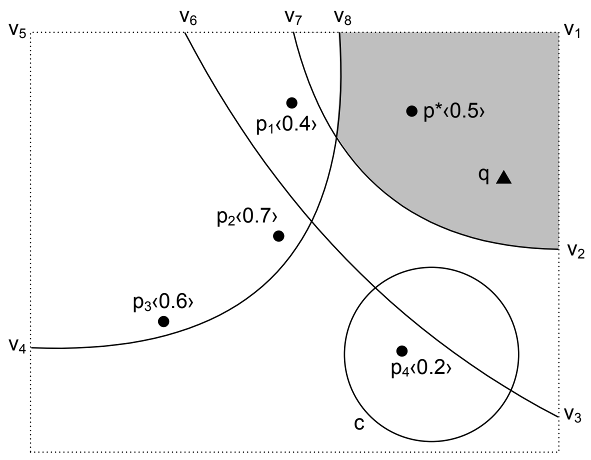
Definition 2.1. The *dominant region* of point a over point b is defined as:

$$Dom_{a,b} = \{z \in \mathcal{U} \ \mid \ d_w(z,a) \le d_w(z,b)\}. \tag{4}$$

To characterize the shape of $Dom_{a,b}$, we define the Apollonius circle [Durell 1961] and explain its relationship with $Dom_{a,b}$. As an example, Figure 4(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) Apollonius circle



(b) MW-Voronoi region

Fig. 4. Example MW-Voronoi Region

Definition 2.2. ([Durell 1961]) Given two points a and b and a constant $0 \le \mu \le 1$, 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: $\|z\ a\| \le \mu \cdot \|z\ b\|$.

LEMMA 2.3. **[Radius and center, Apollonius circle]** *([Okabe et al. 2000]) 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 5 and 6. In addition, we have: $Dom_{b,a} = \mathcal{U} - C_{a,b}$, where $\mathcal{U}$ is the spatial domain.*

$$o_{a,b} = \left(\frac{w^2(b)\cdot a_x - w^2(a)\cdot b_x}{w^2(b) - w^2(a)}, \frac{w^2(b)\cdot a_y - w^2(a)\cdot b_y}{w^2(b) - w^2(a)} \right) \tag{5}$$

$$r_{a,b} = \frac{w(a)\cdot w(b)\cdot \|a\ b\|}{w^2(b) - w^2(a)} \tag{6}$$

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:

$$Dom_{a,b} = \begin{cases} C_{a,b} & \text{if } w(a) < w(b) \\ \mathcal{U} - C_{b,a} & \text{if } w(a) > w(b) \\ \perp_{a,b} & \text{if } w(a) = w(b) \end{cases} \tag{7}$$

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 [Okabe et al. 2000].

Definition 2.4. Given a (result) point $p^* \in \mathcal{D}$, its *MW-Voronoi region* with respect to $\mathcal{D}$ is defined as:

$$\Upsilon(p^*) = \bigcap_{p' \in \mathcal{D} - \{p^*\}} Dom_{p^*,p'} \tag{8}$$

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. [2004], by applying Equation 7, we can re-express the MW-Voronoi region as Equation 9. 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.

$$\begin{aligned} \Upsilon(p^*) &= \bigcap_{p \in \mathcal{D} - \{p^*\}} Dom_{p^*,p} \\ &= \bigcap_{p_j \in \mathcal{D}^+} C_{p^*,p_j} \cap \bigcap_{p_i \in \mathcal{D}^o} \perp_{p^*,p_i} \cap \bigcap_{p_k \in \mathcal{D}^-} (\mathcal{U} - C_{p_k,p^*}) \\ &= \bigcap_{p_j \in D^+} C_{p^*,p_j} \cap \bigcap_{p_i \in \mathcal{D}^o} \perp_{p^*,p_i} - \bigcup_{p_k \in D^-} C_{p_k,p^*} \end{aligned} \tag{9}$$

Figure 4(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

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

2.3. Choice of Index Structure

The techniques we propose require two primitive operations on spatial text data: (i) retrieving objects by their spatial proximity to a query q, and (ii) retrieving objects by their text relevancy to q. Any index supporting these two operations can be easily integrated with our proposed techniques. The R-tree [Guttman 1984] supports the first operation, and the inverted index [Zobel and Moffat 2006] supports the second operation, but neither supports both operations simultaneously.

We adopt the IR-tree [Cong et al. 2009; Wu et al. 2012] for indexing data objects at the server, which is a state-of-the-art index that enables pruning according to space and text simultaneously and efficiently supports the processing of ranking-aware spatial keyword queries. Our algorithms are also applicable to variants of the IR-tree [Cong et al. 2009; Li et al. 2011; Wu et al. 2012]. We are not aware of any other indexes that can support the computation of text relevancy and spatial approximation simultaneously.

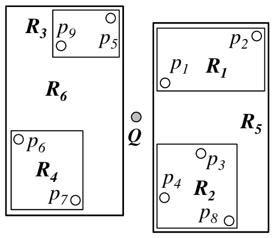
The IR-tree is an R-tree [Guttman 1984] extended with inverted files [Zobel and Moffat 2006]. Each leaf node contains entries of the form $e_o = \langle ptr, \Lambda, \psi \rangle$, where $e_o.ptr$ is a reference to a data object, $e_o.\Lambda$ is a minimum bounding rectangle (MBR), and $e_o.\psi$ refers 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 $e = \langle ptr, \Lambda, \psi \rangle$, where $e.ptr$ is a reference to a child node of N, $e.\Lambda$ is the MBR of all rectangles in entries of the child node, and $e.\psi$ is a reference 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 on the text descriptions of the entries stored in the node.

Figure 5(a) contains 9 spatial objects, and Figure 5(b) shows the frequencies of the words in the description of each object. For example, the description of p_1 contains the words a and c five times each. Figure 6(a) illustrates the corresponding IR-tree, and Figure 6(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.

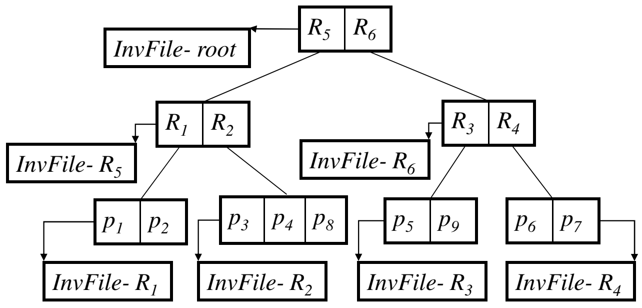


(a) object locations

	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. 5. A Data Set of Spatial Keyword Objects



(a) Structure of the IR-tree

InvF-root	*InvF-R*$_5$	*InvF-R*$_6$
a: $(R_5, 7), (R_6, 4)$ b: $(R_5, 5), (R_6, 4)$ c: $(R_5, 7), (R_6, 4)$ d: $(R_5, 1), (R_6, 1)$	a: $(R_1, 5), (R_2, 7)$ b: $(R_1, 5), (R_2, 3)$ c: $(R_1, 5), (R_2, 7)$ d: $(R_2, 1)$	a: $(R_3, 4), (R_4, 1)$ b: $(R_4, 4)$ c: $(R_3, 4), (R_4, 4)$ d: $(R_4, 1)$

InvF-R$_1$	*InvF-R*$_2$	*InvF-R*$_3$	*InvF-R*$_4$
a: $(p_1, 5)$ b: $(p_2, 5)$ c: $(p_1, 5), (p_2, 5)$	a: $(p_3, 7)$ b: $(p_8, 3)$ c: $(p_4, 7), (p_8, 3)$ d: $(p_3, 1), (p_4, 1)$	a: $(p_5, 4), (p_9, 3)$ c: $(p_5, 4), (p_9, 3)$	a: $(p_7, 1)$ b: $(p_6, 4), (p_7, 1)$ c: $(p_6, 3), (p_7, 4)$ d: $(p_7, 1)$

(b) Content of the Inverted Files of the nodes

Fig. 6. Example of an IR-Tree

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 2.5. **[Monotonicity, text relevancy function]** *([Cong et al. 2009]) Given a query q and an entry e with its rectangle $e.\Lambda$, we have $\forall p \in e.\Lambda$ $(tr_{q.\psi}(e.\psi) \geq tr_{q.\psi}(p.\psi))$.*

As an example in Figure 6, given any query q, $tr_{q.\psi}(R_5.\psi) \geq tr_{q.\psi}(R_1.\psi) \geq tr_{q.\psi}(p_1.\psi)$.

3. 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. For the safe zone, we present its *geometric interpretation* (the region in 2D space) and the *storage representation* in terms of the influence objects. 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.

3.1. 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 4, 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:

$$\Omega_I = \bigcap_{p' \in I} Dom_{p^*,p'}. \tag{10}$$

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 3.1. **[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:

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

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 $p' \in I$. $\square$

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 2.2 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 7, we have: $Dom_{p^*,p_+} \supseteq Dom_{p^*,p'}$. By Lemma 3.1, since $p' \in I$, we get: $Dom_{p^*,p'} \supseteq \Omega$. Combining them, we obtain: $Dom_{p^*,p_+} \supseteq \Omega$, so $Dom_{p^*,p_+} \cap \Omega = \Omega$. $\square$

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 7, we have: $Dom_{p^*,p_-} \supseteq Dom_{p^*,p'}$. By Lemma 3.1, 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 7, we have: $Dom_{p^*,p_-} \supseteq Dom_{p^*,p'}$. By Lemma 3.1, 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.*

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.*

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.*

The proofs of Pruning Rules 4, 5, and 6 are trivial and similar to the proofs of Pruning Rules 2, 1, and 1, respectively.

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 4, 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 $C_{p^*,p_2} \cap C_{p_4,p^*} = \emptyset$.

Table I. Pruning Rules for Unseen Objects

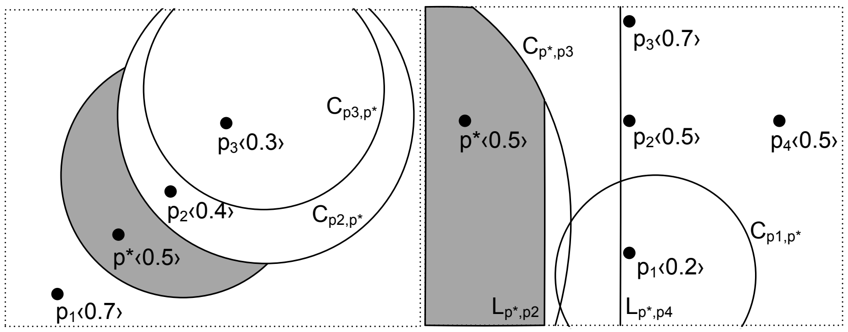
	$p_+ \in \mathcal{D}^+$	$p_- \in \mathcal{D}^-$	$p_o \in \mathcal{D}^o$
I^+	Pruning Rule 1: $\exists p' \in I^+$ $(C_{p^*,p_+} \supseteq C_{p^*,p'})$	Pruning Rule 2: $\exists p' \in I^+$ $(C_{p_-,p^*} \cap C_{p^*,p'} = \emptyset)$	Pruning Rule 5: $\exists p' \in I^+$ $(\perp_{p^*,p_o} \supseteq C_{p^*,p'})$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

Now let p^* be the top result in Figure 7(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 7(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 $\perp_{p^*,p_4} \supseteq \perp_{p^*,p_2}$.

Safe Zone Representation and Client-Side Test.

The finalized set of objects $I = I^+ \cup I^- \cup I^o$, called the influence set (storage representation), 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: $\bigwedge_{p \in (I^+ \cup I^- \cup I^o)} (q \in Dom_{p^*,p})$ (geometric interpretation). This checking is Step 1b in the architecture (Figure 1). The client needs not compute the shape of the safe zone. Specifically, given an influence object $p \in I$, checking whether $q \in Dom_{p^*,p}$ is done by computing the center and radius (Equations 5 and 6) of the



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

Fig. 7. Pruning Rule Examples

Apollonius circle $C_{p^*,p}$ (if $w(p) > w(p^*)$), the Apollonius circle C_{p,p^*} (if $w(p) < w(p^*)$), or the perpendicular half plane $\perp_{p^*,p}$ (if $w(p) = w(p^*)$), and then testing whether q is inside $C_{p^*,p}$ (if $w(p) > w(p^*)$), is outside C_{p,p^*} (if $w(p) < w(p^*)$), or is inside $\perp_{p^*,p}$ (if $w(p) = w(p^*)$).

Implementation Aspects.
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 $\exists p' \in I^+$ $(\|o_{p^*,p'}\ o_{p^*,p_+}\| \le r_{p^*,p_+} - r_{p^*,p'})$, which can be evaluated by using the centers and radii of the respective Apollonius 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 $\exists p' \in I^+$ $(\|o_{p^*,p'}\ o_{p_-,p^*}\| \ge r_{p_-,p^*} + r_{p^*,p'})$. Similar conversions are applied to the topological conditions of the other rules. Table II shows the conversions of the 6 pruning rules in Table I. Table II, $L_{a,b}$ denotes the perpendicular bisector between points a and b.

Table II. Conversions of Pruning Rules

Pruning Rule 1	$\exists p' \in I^+$ $(\|o_{p^*,p'}\ o_{p^*,p_+}\| \le r_{p^*,p_+} - r_{p^*,p'})$
Pruning Rule 2	$\exists p' \in I^+$ $(\|o_{p^*,p'}\ o_{p_-,p^*}\| \ge r_{p_-,p^*} + r_{p^*,p'})$
Pruning Rule 3	$\exists p' \in I^-$ $(\|o_{p',p^*}\ o_{p_-,p^*}\| \le r_{p',p^*} - r_{p_-,p^*})$
Pruning Rule 4	$\exists p' \in I^o$ $(\|L_{p^*,p'}\ o_{p_-,p^*}\| \ge r_{p_-,p^*} \wedge p_- \in \perp_{p',p^*})$
Pruning Rule 5	$\exists p' \in I^+$ $(\|o_{p^*,p'}\ L_{p_o,p^*}\| \ge r_{p^*,p'})$
Pruning Rule 6	$\exists p' \in I^o$ $(parallel(L_{p^*,p'}, L_{p^*,p_o}) \wedge \|p^*\ p_o\| \ge \|p^*\ p'\|)$

3.2. 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 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 4(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 3.2 and cover its computation in Lemma 3.3.

Definition 3.2. 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 3.3. **[Border distance computation]** *The functions $bord_{min}(\cdot,\cdot)$ and $bord_{max}(\cdot,\cdot)$ can be expressed as:*

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

$$bord_{min}(b, C_{a,b}) = \|b\ M\| = \frac{w(b)}{w(a)+w(b)} \cdot \|a\ b\| \tag{12}$$

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

$$bord_{max}(b, C_{a,b}) = \|b\ N\| = \frac{w(b)}{w(b)-w(a)} \cdot \|a\ b\| \tag{14}$$

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. [2000], 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 2.2 (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 $bord_{min}(\cdot,\cdot)$ and $bord_{max}(\cdot,\cdot)$ in terms of $w(a)$, $w(b)$, and $\|a\ b\|$, as in Equations 11–14. □

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 and the maximum border distance of point p from p^* as:

$$bord_{min}(p^*, p) = \begin{cases} bord_{min}(p^*, C_{p^*,p}) & \text{if } w(p^*) \le w(p) \\ bord_{min}(p^*, C_{p,p^*}) & \text{otherwise.} \end{cases} \tag{15}$$

$$bord_{max}(p^*, p) = \begin{cases} bord_{max}(p^*, C_{p^*,p}) & \text{if } w(p^*) \le w(p) \\ bord_{max}(p^*, C_{p,p^*}) & \text{otherwise.} \end{cases} \tag{16}$$

Lemma 3.4 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 3.4. **[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 3.1, 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 8 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.

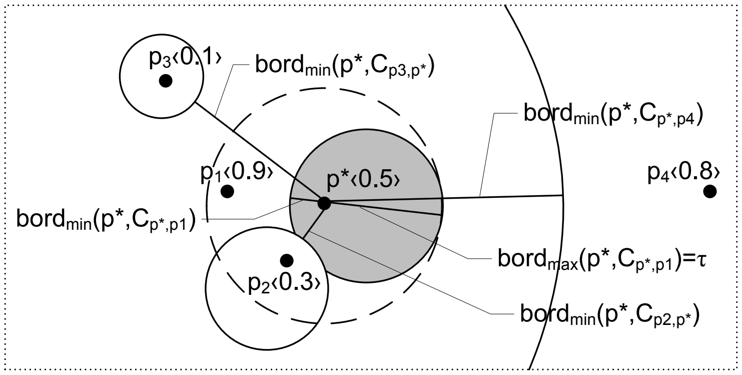


Fig. 8. Early Stopping Example

3.3. 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 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 3.4 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 16. For the first case, we obtain $w(p^*) \leq w(p)$ and then substitute $a = p^*$ and $b = p$ into Equation 11. 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:

$$bord_{min}(p^*, \Lambda) = \frac{w(p^*)}{w(p^*) + w^u(\Lambda)} \cdot \|p^* \ \Lambda\|_{min}. \tag{17}$$

For the second case, we obtain $w(p^*) > w(p)$ and then substitute $b = p^*$ and $a = p$ into Equation 12. 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. [2009] and Wu et al. [2012].

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 the 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.

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:  $\tau \leftarrow \infty$;                                   $\triangleright$ $\tau$ 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:      $e \leftarrow H.deheap()$;        $\triangleright$ $e$ is a data structure containing a node or an object and its key
7:      if $e$ contains an object and $e \neq p^*$ then
8:          if $w(e) > w(p^*)$ then                             $\triangleright$ object in $\mathcal{D}^+$
9:              if $\nexists p' \in I^+(C_{p^*,e} \supseteq C_{p^*,p'})$ then          $\triangleright$ rule 1
10:                 insert $e$ into $I^+$;
11:                 $\tau \leftarrow \min\{\tau, bord_{max}(C_{p^*,e})\}$;           $\triangleright$ shrink $\tau$
12:         if $w(e) < w(p^*)$ then                             $\triangleright$ object in $\mathcal{D}^-$
13:             if $\nexists p' \in I^+(C_{e,p^*} \cap C_{p^*,p'} = \emptyset)$ and    $\triangleright$ rule 2
14:                $\nexists p' \in I^-(C_{e,p*} \subseteq C_{p',p^*})$ and            $\triangleright$ rule 3
15:                $\nexists p' \in I^o(\perp_{p^*,p'} \cap C_{e,p^*} = \emptyset)$ then  $\triangleright$ rule 4
16:                 insert $e$ into $I^-$;
17:         if $w(e) = w(p^*)$ then                             $\triangleright$ object in $\mathcal{D}^o$
18:             if $\nexists p' \in I^+(\perp_{p^*,e} \supseteq C_{p^*,p'})$ and     $\triangleright$ rule 5
19:                $\nexists p' \in I^o(\perp_{p^*,p'} \subseteq \perp_{p^*,e})$ then  $\triangleright$ rule 6
20:                 insert $e$ into $I^o$;
21:     else                                                    $\triangleright$ $e$ contains a node
22:         for each entry $e'$ in $e$ do
23:             $H.enheap(e', bord_{min}(p^*, e'.\Lambda))$;
24: return the set $I^+ \cup I^- \cup I^o$;
```

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 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 [Okabe et al. 2000].

Extension to Arbitrary k.
Let $\mathcal{RS}$ be the top-k result of the query q. According to Okabe et al. [2000], 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 [Okabe et al. 2000]:

$$\Upsilon^k(\mathcal{RS}) = \bigcap_{p_j^* \in \mathcal{RS}} \Upsilon_{\mathcal{D}-\mathcal{RS}}(p_j^*), \tag{18}$$

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 to 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 enables 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 18, 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 23) as

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

4. 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.

4.1. 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:

$$Dom_{p^*,\mathfrak{R}} = \bigcap_{p \in \mathfrak{R},\ 0 \le w(p) \le w^u(\mathfrak{R})} Dom_{p^*,p}. \tag{19}$$

With this concept, we can extend the pruning rules of Section 3 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 4.1. **[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 2.1, we have:

$$\forall z \in Dom_{p^*,p} \left(\frac{\|p^*\ z\|}{w(p^*)} \leq \frac{\|p\ z\|}{w(p)} \right).$$

Since $p.\lambda = p'.\lambda$, we have $\|p\ z\| = \|p'\ z\|$. In addition, we have $w(p) \geq w(p')$. Thus,

$$\forall z \in Dom_{p^*,p} \left(\frac{\|p^*\ z\|}{w(p^*)} \leq \frac{\|p'\ z\|}{w(p')} \right).$$

According to the definition of $Dom_{p^*,p'}$, we have $Dom_{p^*,p} \subseteq Dom_{p^*,p'}$. □

LEMMA 4.2. **[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 2.1 and using $w(p) = w(p')$, we obtain the centers and radii of circles $C_{p^*,p}$ and $C_{p^*,p'}$ as:

$$o_{p^*,p} = \left(\frac{-w^2(p^*) \cdot x_1}{w^2(p) - w^2(p^*)}, 0 \right), \quad o_{p^*,p'} = \left(\frac{-w^2(p^*) \cdot x_2}{w^2(p) - w^2(p^*)}, 0 \right),$$

$$r_{p^*,p} = \frac{w(p^*) \cdot w(p) \cdot x_1}{w^2(p) - w^2(p^*)}, \quad r_{p^*,p'} = \frac{w(p^*) \cdot w(p) \cdot x_2}{w^2(p) - w^2(p^*)}.$$

By using $w(p^*) \leq w(p)$, we derive:

$$\|o_{p^*,p}\ o_{p^*,p'}\| = \frac{w^2(p^*) \cdot (x_2 - x_1)}{w^2(p) - w^2(p^*)} \leq \frac{w(p^*) \cdot w(p) \cdot (x_2 - x_1)}{w^2(p) - w^2(p^*)} = r_{p^*,p'} - r_{p^*,p}. \quad (20)$$

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 9, 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 10(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 10(b) will be discussed later.

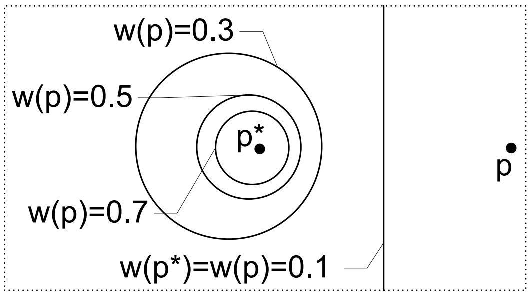
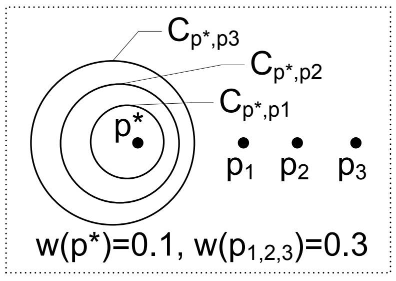
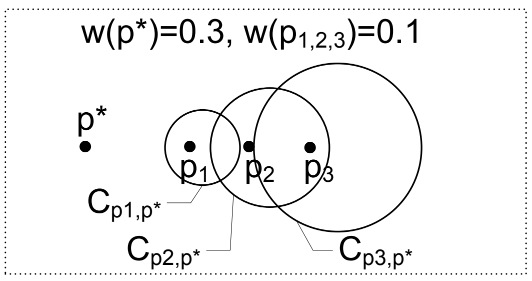


Fig. 9. Varying the Weight of p



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



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

Fig. 10. 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 4.3. Figure 11 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.

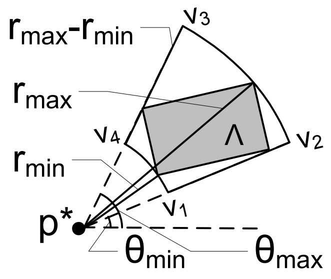


Fig. 11. The Minimum Bounding Ring-Sector

Definition 4.3. 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 4.4. **[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 let arc_s be the arc of rs_Λ closest to p^*. It holds that:*

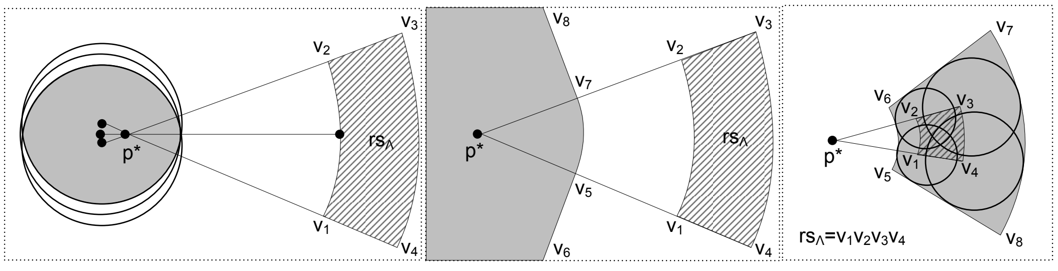
$$Dom_{p^*,\Lambda} \supseteq \bigcap_{p' \in arc_s,\ w(p')=w^u(\Lambda)} Dom_{p^*,p'}, \tag{21}$$

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 4.1 and 4.2, we have $Dom_{p^*,p'} \subseteq Dom_{p^*,p}$. Hence, we have $Dom_{p^*,\Lambda} \supseteq \bigcap_{p' \in arc_s} Dom_{p^*,p'}$. $\square$

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 4.4, 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 12(b)) 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 $v_7 v_8$.



(a) $w^u(\Lambda) > w(p^*)$ (b) $w^u(\Lambda) = w(p^*)$ (c) $w^u(\Lambda) < w(p^*)$

Fig. 12. Representing the Shape of the Minimal Dominant Region $Dom_{p^*,\Lambda}$

The shape of $Dom_{p^*,\Lambda}$ when $w^u(\Lambda) > w(p^*)$.
By Lemma 4.4, 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 coordinate 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: $(p_x^* + \|p^*\ z\| \cdot \cos\theta - o_x)^2 + (p_y^* + \|p^*\ z\| \cdot \sin\theta - o_y)^2 = r_{p^*,p}^2$. Using $F_\alpha = p_x^* - o_x$, $F_\beta = p_y^* - o_y$, and solving for $\|p^*\ z\|$ in the above equation, we obtain Equation 22, which expresses the magnitude $\|p^*\ z\|$ as a function of θ.

$$\|p^*\ z\| = \sqrt{F_\gamma^2 - F_\alpha^2 - F_\beta^2 + r_{p^*,p}^2} - F_\gamma, \tag{22}$$

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 12(a). 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 22, 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 13, we have: $r_{p^*,p} = 1$, $(o_x, o_y) = (0, 0)$, and $(p_x^*, p_y^*) = (0, 0.5)$. Figure 13(b) plots the value of $\|p^*\ z\|$ with respect to θ (see Equation 22). The $\times$-curve is obtained by shifting the $\diamond$-curve to the left by the angle $\delta = \pi/3$. For each angle $\theta = 2\pi i/f$ where $i \in [1, f]$, we compute the minimum $\|p^*\ z\|$ and obtain its (x, y) coordinates (see Figure 13(a)). Then we link these f points to form a polygon, which is guaranteed to be a subset of Dom_{p^*,arc_s}.

The shape of $Dom_{p^*,\Lambda}$ when $w^u(\Lambda) < w(p^*)$.
In this case, we have:

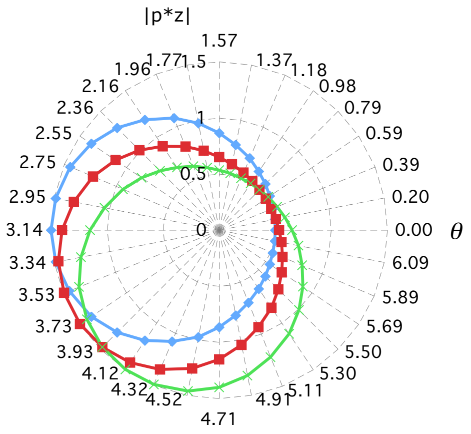
$$Dom_{p^*,\Lambda} = \bigcap_{p\in\Lambda} Dom_{p^*,p} = \bigcap_{p\in\Lambda} (\mathcal{U} - C_{p,p^*}) = \mathcal{U} - \bigcup_{p\in\Lambda} C_{p,p^*}.$$

In order to obtain a subset representation of $Dom_{p^*,\Lambda}$, we need to derive a superset representation of the region $\bigcup_{p\in\Lambda} C_{p,p^*}$.

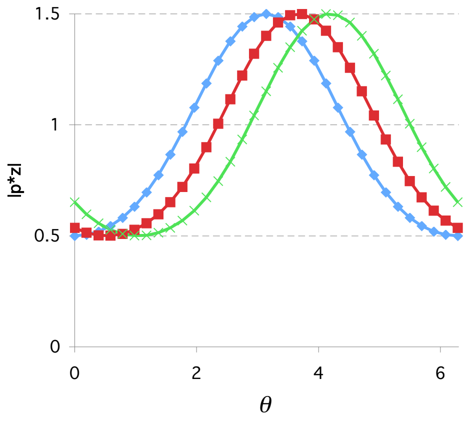
Let p be a point in Λ and consider the example of Figure 10(b) where $w(p^*) > w(p)$. We fix the weight of p while moving its location (e.g., p_1, p_2, p_3) away from p^* along the line segment $\overline{p^*\ p}$. The union of these circles can be represented by a ring-sector. Figure 12(c) illustrates 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.

4.2. Optimizations

Upgraded Voronoi Cell—Enhancing Rules 1, 2, 4, 5, and 6.
The Voronoi cell optimization (presented in Section 3.3) employs a convex polygon Φ to represent



(a) polar view from p^*

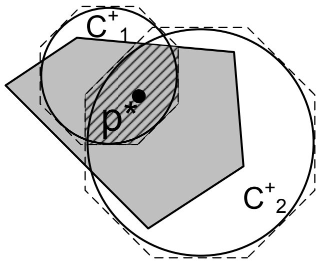


(b) angle-radius plot

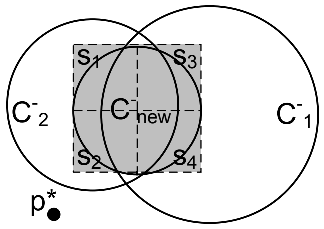
Fig. 13. Expressing $\|p^*\ z\|$ as a Function of θ

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 14(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. 14. Optimization Techniques for the MSK Algorithm

Recursive Refinement of I^-—Enhancing Rule 3.
We finally enhance pruning rule 3. Consider the example in Figure 14(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 p_{new} because neither C_1^- nor C_2^- contains C_{new}^-.

We propose a recursive refinement technique that removes 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 14(b). Then we check whether each square is covered by some circle, e.g., s_1 and s_2 are covered by C_2^-, and 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.

4.3. 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 4.1. 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^-.

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

```
    System parameters: $f, l_{max}$                                   ▷ used for optimizations
 1: $I^+ \leftarrow$ new set;  $I^- \leftarrow$ new set;
 2: $\Phi \leftarrow$ the space domain $\mathcal{U}$;                              ▷ a convex polygon
 3: $H \leftarrow$ new min-heap;
 4: $H.enheap(root, 0)$;
 5: while $H$ is not empty do
 6:     $e \leftarrow H.deheap()$;     ▷ $e$ is a data structure containing a node or an object and its key
 7:     if $e$ contains an object and $e \neq p^*$ then
 8:         if $w(e) > w(p^*)$ then
 9:             if $\Phi \not\subseteq C_{p^*,e}$ then                                ▷ enhanced rule 1
10:                 insert $e$ into $I^+$;
11:                 $G \leftarrow$ an $f$-sided polygon that contains $C_{p^*,e}$;
12:                 $\Phi \leftarrow \Phi \cap G$;                     ▷ upgraded Vor. cell optimization
13:         else if $w(e) = w(p^*)$ then
14:             if $\Phi \not\subseteq \perp_{p^*,e}$ then                           ▷ enhanced rules 5,6
15:                 $\Phi \leftarrow \Phi \cap \perp_{p^*,e}$;
16:         else if $w(e) < w(p^*)$ then
17:             if $\Phi \cap C_{e,p^*} \neq \emptyset$ then                          ▷ enhanced rules 2,4
18:                 Recur_Refine($e, I^-, l_{max}$);                          ▷ enhanced rule 3
19:     else                                                       ▷ $e$ contains a node
20:         if ($w^u(e) \geq w(p^*)$ and $\Phi \subseteq Dom_{p^*,e}$) or
21:           ($w^u(e) < w(p^*)$ and $\Phi \cap Dom_{e,p^*} = \emptyset$) then
22:             continue;                                              ▷ pruning a node
23:         for each entry $e'$ in $e$ do
24:             $H.enheap(e', bord_{min}(p^*, e'.\Lambda))$;
25: return the set $I^+ \cup I^-$ and the polygon $\Phi$;
```

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 18). 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 these regions 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 these regions does not intersect Φ then we prune the subtree of e.

5. COMMUNICATION FREQUENCY AND COST ANALYSIS

This section presents an analytical study of the communication frequency and cost. The safe zone of a moving top-k spatial keyword query is an order-k $(k \geq 1)$ multiplicatively weighted Voronoi (MWV) cell. For the sake of simplicity, we consider the case $k = 1$.

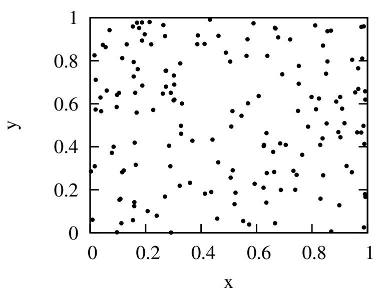
5.1. Communication Frequency Analysis

In this section, we first analyze the expected area of a safe zone, which can be used to indicate on average for how long a safe zone remains valid. Based on the expected area of a safe zone, the expected time for a user to leave that safe zone is derived. The expected safe zone leaving time is for one safe zone. In the real application, a user may enter and leave multiple safe zones. We finally derive the communication frequency for a user whose speed is V and the travel time is T.

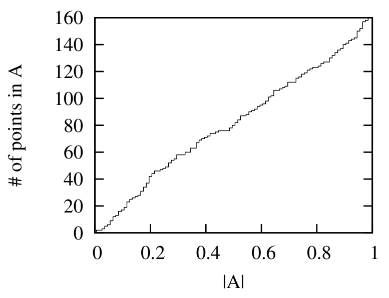
5.1.1. Expected Area of a Safe Zone. We assume that the locations of the objects in a set $\mathcal{D}$ are generated according to a homogeneous Poisson point process, which is a frequently used tool for the modeling and analysis of spatial data [Diggle 2003]. It is the simplest stochastic model for a planar point pattern. The idea of this model is that the point events of interest occur completely independently of each other. It is a spatial generalization of the standard one-dimensional Poisson process in which points occur continuously and independently of one another. For many years, almost all of the available methods for statistical analysis of planar point patterns were based on the assumption that the points in question form a realization of a planar homogeneous Poisson process. Although today there exists a large variety of alternative point process models, the homogeneous Poisson process still provides a basic reference model against which to compare other models. Let A be a region and let $N(A)$ be the number of points placed in A. Formally, the probability, $Pr(N(A) = x)$, that x points are placed in A $(\subseteq S \subseteq \mathbb{R}^2)$ when $\mathcal{N}$ points are generated in S is given by

$$Pr(N(A) = x) = \frac{(\rho|A|)^x e^{-\rho|A|}}{x!}, x = 0, 1, 2, \ldots, \tag{23}$$

where $|A|$ is the area of A and $\rho = \mathcal{N}/|S|$ is the point density of S. Figure 15(a) shows a simulation of a homogeneous Poisson point process on $[0, 1] \times [0, 1]$ with $\rho = 150$. Figure 15(b) shows the Poisson counting process, i.e., the number of points in A, $N(A)$, as a step function of $|A|$.



(a) Simulation of point process



(b) $N(A)$

Fig. 15. Homogeneous Poisson Point Process

Let Υ_{p_i} be the MWV region of a random object p_i in $\mathcal{D}$ and w_i be the weight of p_i. The probability, $\mathbf{P}(z \in \Upsilon_{p_i})$, that a point z in $\mathbb{R}^2$ belongs to Υ_{p_i} is given by

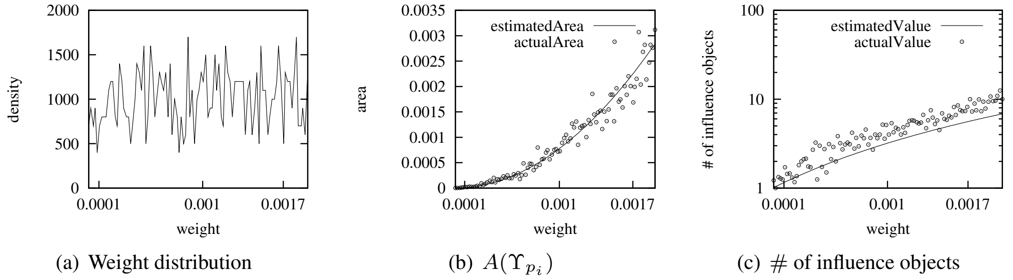
$$\mathbf{P}(z \in \Upsilon_{p_i}) = \mathbf{P}(\bigwedge_{j \in [1,\mathcal{N}] \wedge j \neq i} (\frac{\|z\, p_i\|}{w_i} < \frac{\|z\, p_j\|}{w_j})). \tag{24}$$

The expected area of the safe zone Υ_{p_i} of any object p_i with weight w_i is given by

$$\begin{aligned}
\mathbf{E}(A(w_i)) &= \int_{z \in \mathbb{R}^2} \mathbf{P}(z \in \Upsilon_{p_i}) \\
&= \int_0^{\infty} \prod_{j \in [1,\mathcal{N}] \wedge j \neq i} \mathbf{P}(\|o\, p_j\| > \frac{w_j}{w_i} r)\mathbf{P}(\|o\, p_i\| \in [r, r+dr]) \qquad (25) \\
&= \int_0^{\infty} e^{-\rho\pi r^2 \sum_{j \in [1,\mathcal{N}]} \frac{w_j^2}{w_i^2}} 2\rho\pi r\, dr \\
&= \frac{w_i^2}{\sum_{j \in [1,\mathcal{N}]} w_j^2}. \qquad (26)
\end{aligned}$$

The detailed derivation of the above equations is covered in Section A of the electronic appendix.

The expected area of Υ_{p_i}, defined by Equation 25, is applicable for any type of weight distribution. To validate the above analysis, we experimentally consider the expected area $A(w_i)$ for three types of weight distributions: (1) uniform distribution (Figure 16(a)), (2) Gaussian distribution (Figure 17(a)), and (3) Zipfian distribution (Figure 18(a)). We generate $\mathcal{N} = 1000$ objects in the unit square whose locations follow the Poisson process (Equation 23). We assume $\sum_{j \in [1,\mathcal{N}]} w_j = 1$. The unit square is partitioned into a 1000×1000 grid. By assigning each grid cell to its nearest object in terms of weighted distance (Equation 1), the actual area of the safe zone of each object is collected which is plotted in Figures 16(b), 17(b), and 18(b), where it is denoted by **actualArea**. The estimated area of the safe zone of each object computed from Equation 25 is also shown in the figures, where it is denoted by **estimatedArea**. The estimated values are close to the actual values. The expected area Υ_{p_i} increases as the weight w_i of p_i increases.

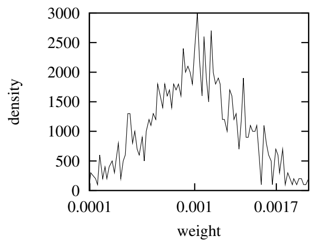


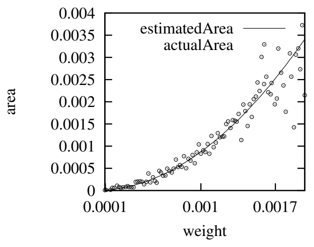
(a) Weight distribution (b) $A(\Upsilon_{p_i})$ (c) # of influence objects

Fig. 16. Weights Following a Uniform Distribution

5.1.2. Average Communication Frequency. To estimate the average communication frequency, we first derive the safe zone leaving time and the average safe zone leaving time.

To estimate the safe zone leaving time, we need to estimate the user's movement. For simplicity, we assume the following movement model. Observe that the user submits the query with an updated location when reaching the border of the safe zone. At the time when the result and the new safe zone are received, the user is located at the center of the safe zone and moves with some velocity


(a) Weight distribution


(b) $A(\Upsilon_{p_i})$

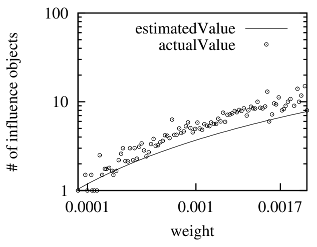
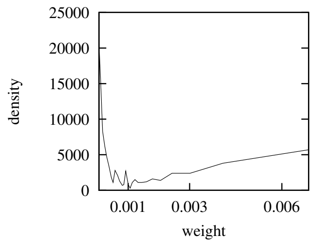
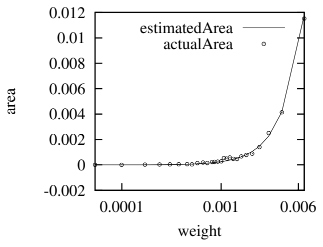

(c) $\#$ of influence objects

Fig. 17. Weights Following a Gaussian Distribution


(a) Weight distribution


(b) $A(\Upsilon_{p_i})$

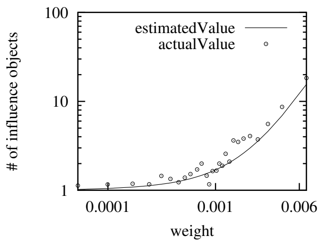

(c) $\#$ of influence objects

Fig. 18. Weights Following a Zipfian Distribution

$|V|$. We have derived the expected area of the safe zone of a random object p_i, $\mathbf{E}(A(w_i))$. For the sake of simplicity, we assume the safe zone Υ_{p_i} is a circle centered at p_i. It then has the following radius.

$$r(w_i) = \sqrt{\frac{\mathbf{E}(A(w_i))}{\pi}}. \tag{27}$$

We assume that the user follows linear movement pattern which models the worst-case communication frequency. The safe zone leaving time for p_i is then $LT(w_i) = r(w_i)/|V|$. Then the average safe zone leaving time is

$$\begin{aligned} \mathbf{E}(LT) &= \sum_{i\in[1,\mathcal{N}]} \mathbf{E}(A(\omega_i))LT(\omega_i) \\ &= \sum_{i\in[1,\mathcal{N}]} \frac{\mathbf{E}(A(\omega_i))^{3/2}}{\pi^{1/2}|V|} \\ &= \frac{\sum_{i\in[1,\mathcal{N}]} \omega_i^3}{|V|\pi^{(1/2)}(\sum_{j\in[1,\mathcal{N}]} \omega_j^2)^{3/2}}, \end{aligned} \tag{28}$$

where $\mathbf{E}(A(\omega))$ can be considered as the probability for a random user falling into the safe zone of an object with weight ω, and $\int_{\omega\in\mathbb{W}} \mathbf{E}(A(\omega)) = 1$. Hence, the average communication frequency is

$$CF = 1/\mathbf{E}(LT).$$

We validate the average communication frequency using the synthetic dataset described in Section 5.1.1 and a real dataset EURO. The details of EURO are described in Section 8.1. We model the location of a query q as $q.\lambda(t) = (x, y) + (v_x, v_y) \cdot \theta$, where θ is an angle. We consider linear movements where θ changes only when the query reaches the boundary of the space). Tables III and IV show the real and estimates values of the average communication frequency of three types of data distributions on the synthetic dataset and EURO, where in EURO (100, 200, 300), the weights of objects are computed according to three query keywords with IDs 100, 200, and 300; in EURO (200, 300), the weights of objects are determined by two query keywords with IDs 200 and 300. The estimate values are close to the real values. As the speed increases, the average communication frequency increases as expected.

Table III. Average Comm. Frequency of Linear Movement on Synthetic Data

speed	uniform distribution		Gaussian distribution		Zipfian distribution	
(cell/timestamp)	real	estimate	real	estimate	real	estimate
1	0.0288	0.0096	0.0295	0.0111	0.0226	0.0057
5	0.0982	0.0474	0.0939	0.0478	0.0834	0.0271
10	0.1833	0.0963	0.1862	0.1078	0.1392	0.0585
20	0.3566	0.1951	0.3589	0.2062	0.2951	0.1140

Table IV. Average Comm. Frequency of Linear Movement on EURO

speed	EURO (200, 300)		EURO (100, 200, 300)	
(cell/timestamp)	real	estimate	real	estimate
1	0.0371	0.0283	0.0408	0.0423
5	0.1590	0.1415	0.2205	0.2115
10	0.2879	0.2830	0.3822	0.4231
20	0.5873	0.5660	0.7815	0.8462

5.2. Communication Cost Analysis

5.2.1. Expected Number of Influence Objects. Let D_i^+, D_i^o, and D_i^- be the sets that contain objects having higher, equal, and lower weights than does p_i. Assume there are $\mathcal{N}$ objects in dataset $\mathcal{D}$. Let $|\mathcal{U}|$ be the area of the whole dataset.

LEMMA 5.1. *Let p_i and p_j be points of the dataset $\mathcal{D}$. The condition of p_j not being an influence object of p_i is:*

$$\|p_i\ p_j\| \geq \frac{r(w_i)}{w_i} w_j + r(w_i).$$

The proof of Lemma 5.1 is presented in Section B of the electronic appendix.

The expected number of influence objects of p_i is:

$$\begin{aligned}\mathbf{E}(X) &= \sum_{j\in[1,\mathcal{N}]\wedge j\neq i} \mathbf{P}(\|p_i\ p_j\| < \frac{r(w_i)}{w_i} w_j + r(w_i)) \\ &= \mathcal{N}\int_{\omega\in\mathbb{W}} \mathbf{P}(\|p_i\ p_j\| < \frac{r(w_i)}{w_i} w_j + r(w_i))PDF(\omega)d\omega \\ &= \mathcal{N}\int_{\omega\in\mathbb{W}} \frac{\pi(\frac{r(w_i)}{w_i}\omega + r(w_i))^2}{|\mathcal{U}|} PDF(\omega)d\omega. \end{aligned} \quad (29)$$

The actual number of influence objects (for each p_i) can be obtained by constructing MWV regions. However, existing construction methods are not scalable in the dataset size (taking $O(\mathcal{N}^2)$ time). Thus, we approximate the actual number of influence objects by using the following grid-based counting method. Using the 1000×1000 grid described in Section 5.1.1, after assigning each grid cell to its nearest object, each object has a list of grid cells. For each grid cell in the list of object p_i, we check its neighbor (up, down, right, and left) cells. If one neighbor cell takes another object p_j as the nearest object, we add p_j to the influence object list of p_i. Figures 16(c), 17(c), and 18(c) show the actual number of influence objects and the estimated number of influence objects. The estimated values are close to the actual values. The expected number of influence objects increases as the weight w_i of p_i increases.

5.2.2. Average Communication Cost within a Time Interval $[0, T]$. We have derived the expected number of influence objects $\mathbf{E}(X)$, which is actual the communication cost per communication trip. We have also derived the average communication frequency CF. Hence, the average communication cost within a time interval $[0, T]$ is

$$\mathbf{E}(X) \cdot CF \cdot T.$$

6. EFFICIENT NEXT SAFE ZONE COMPUTATION

When a query exits its safe zone, it sends a location update to the server to retrieve a new safe zone and a result. We propose techniques that aim to efficiently compute the new safe zone based on the previous safe zone and results. The resulting algorithm uses the techniques described next.

6.1. Restricting the Search Space

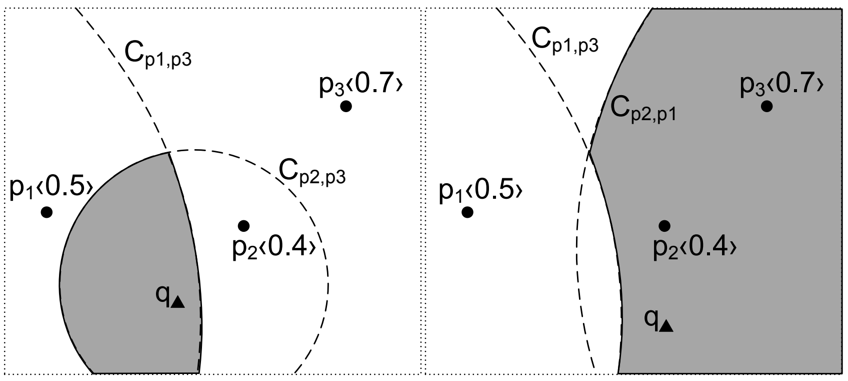
When computing a safe zone from scratch, in the beginning, the temporary safe zone is the whole space of the dataset. However, the current safe zone and the previous safe zone have some common arcs/edges, so that the computations of those shared arcs/edges can be saved. In addition, the influence objects and top-k result objects from the previous safe zone can help restrict the search space, i.e., forming a smaller temporary safe zone that makes the proposed pruning rules more effective.

Algorithm 3 shows the pseudo code of the initialization function that constructs a smaller temporary safe zone instead of using the whole space of the dataset. It takes five parameters: the previous influence object sets I^+_{old}, I^o_{old}, and I^-_{old}, the previous top-k result object set rs_{old}, and the current top-k result object set rs_{new}. Both previous influence objects (Lines 4–12) and previous result objects (Lines 13–18) are used to construct the current temporary safe zone Φ.

Figure 19(a) shows the safe zone obtained when the client last contacted the server. The previous top-2 result object set rs_{old} of the query q is $\{p_1, p_2\}$, and the shaded region is the previous safe zone defined by the Apollonius circles C_{p_1,p_3} and C_{p_2,p_3}. The previous influence object is p_3 ($p_3 \in I^+_{old}$). When the query q moves to the position shown in Figure 19(b), the current top-2 result object set rs_{new} is $\{p_2, p_3\}$. Instead of computing the new safe zone (shaded region in Figure 19(b)) from scratch, we first construct a smaller temporary safe zone. In this example, the Apollonius circle C_{p_1,p_3} can be reused. In addition, since p_1 belongs to the previous top-2 result object set rs_{old}, it is also used to construct the smaller temporary safe zone. Taking advantage of the smaller temporary safe zone, Algorithm 2 is able to compute the final safe zone faster.

6.2. Reusing Text Relevancies of Nodes

Since the keyword part in an MkSK query does not change while the query is moving, the text relevancies (weights) of the nodes in the IR-tree do not change. In Algorithm 2, the text relevancies of some nodes that have been computed before can be reused for the current query position, so that the computation and I/O costs from the inverted files of those nodes are reduced. Specifically, for each MkSK query, a hash table is maintained in main memory to store the weights of the nodes that have been visited. The key is the pair $\langle uid, nid \rangle$, where uid identifies the user and nid identifies the node in the IR-tree. The value is a list of the weights of the entries in that node. When the weight of a node is needed (in Lines 8, 9, 13, 14, 16, 17, and 20 in Algorithm 2), we first check the hash table. If it is not found, we refer to the inverted files on disk to compute its weight. The hash table is created when a query is issued and is destroyed when the query expires.



(a) Top-2 objects: p_1 and p_2 (b) Top-2 objects: p_2 and p_3

Fig. 19. Smaller Temporary Safe Zones

Algorithm 3 Initialization (Set I^+_{old}, Set I^o_{old}, Set I^-_{old}, Result set rs_{old}, Result set rs_{new})

```
1: $I^+_{new} \leftarrow$ new set; $I^o_{new} \leftarrow$ new set; $I^-_{new} \leftarrow$ new set;
2: $\Phi \leftarrow$ the space domain $\mathcal{U}$;                    ▷ a convex polygon
3: $I \leftarrow I^+_{old} \cup I^o_{old} \cup I^-_{old}$;
4: **for** each influence object $o$ in $I$ **do**
5:     **if** $o \notin rs_{new}$ **then**
6:         **for** each result object $r$ in $rs_{new}$ **do**
7:             **if** $\exists r(r \in rs_{old} \wedge o \in I^+_{old})$ **then**
8:                 $\Phi \leftarrow \Phi \cap G$;      ▷ $G$ is an $f$-sided polygon that contains $C_{p^*,e}$
9:                 insert $o$ into $I^+_{new}$;
10:            **else if** $\exists r(r \in rs_{old} \wedge o \in I^o_{old})$ **then**
11:                $\Phi \leftarrow \Phi \cap \perp_{p^*,e}$;
12:                insert $o$ into $I^o_{new}$;
13:            **else if** $\exists r(r \in rs_{old} \wedge o \in I^-_{old})$ **then**
14:                Recur_Refine($e, I^-_{new}, l_{max}$);
15:            **else**
16:                $e \leftarrow o$;
17:                $p^* \leftarrow r$;
18:                Execute Lines 8–18 in Algorithm 2;
19: **for** each result object $r'$ in $rs_{old}$ **do**
20:     **if** $r' \notin rs_{new} \wedge r' \notin I$ **then**
21:         **for** each result object $r$ in $rs_{new}$ **do**
22:             $e \leftarrow r'$;
23:             $p^* \leftarrow r$;
24:             Execute Lines 8–18 in Algorithm 2;
25: return the set $I^+_{new} \cup I^o_{new} \cup I^-_{new}$ and the polygon $\Phi$;
```

6.3. Tighter Bound for Retrieving Top-k Results

The previous top-k result object set rs_{old} and the current top-k result object set rs_{new} probably have some common result objects. Figures 19(a) and 19(b) offer an example where the previous top-2 result object set contains p_1 and p_2 and the current top-2 result object set contains p_2 and p_3. Since top-k results are obtained using best search method, previous top-k results can be considered as candidates of the current top-k results and provide a tighter bound of the stopping threshold.

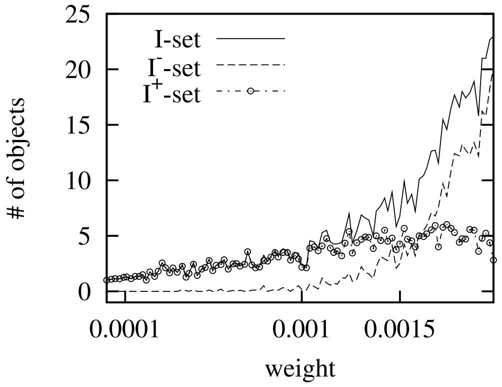


Fig. 20. Sizes of the I^+ and I^- Sets

7. CONSERVATIVE SAFE ZONE FOR REDUCING COMMUNICATION

In general, the shape of a safe zone is an irregular region with holes, each of which corresponds to the Apollonius circle of an influence object whose weight is smaller than the weights of the result objects. It is possible for the number of holes to be so large that the communication cost (i.e., the number of influence objects to be sent) becomes overly high. As a case study, we count the number of influence objects of generated objects with different weights. The locations of generated objects follow a homogeneous Poisson point process, and the weights of generated objects follow a Zipfian distribution. Figure 20 shows the number of influence objects (the I-set curve), the I^+ objects (the I^+-set curve), and the I^- objects (the I^--set curve). The generated objects have unique weights, so the size of the I^o set in this example is 0. The influence objects (the I-set curve) of an object with a small weight are dominated by the I^+ objects (the I^+-set curve), while the influence objects of an object with a large weight are dominated by the I^- objects (the I^--set curve). An object with a large weight can have a large influence object set, of which most are the I^- objects.

To avoid safe zones with many influence objects and thus obtain lower communication costs, we study the use of conservative safe zones. Definition 7.1 gives the definition of the conservative safe zone. Lemma 7.2 shows that the conservative safe zones guarantee the correctness of the result of a query.

Definition 7.1. Let Υ be the real safe zone (defined by Equation 9) of a query q. A conservative safe zone Υ_{con} is a subset of Υ that contains q, i.e., $\Upsilon_{con} \subset \Upsilon \wedge q \in \Upsilon_{con}$.

The computation of conservative safe zones can be implemented efficiently at the server. When the query exits the conservative safe zone and sends a location update to the server, the updated location may remain inside the real safe zone. Hence, in order to save computation and achieve fast query response, for each query, the real safe zone is cached at the server side. When a location update arrives, the server first checks whether it remains inside the real safe zone. If yes, the cached real safe zone is used to construct a new conservative safe zone. Otherwise, the real safe zone is recalculated.

LEMMA 7.2. *The use of conservative safe zones guarantees the correctness of the results of queries.*

PROOF. Let Υ be the real safe zone of a query q and let $\mathcal{RS}$ be the result when q stays inside Υ. Since a conservative safe zone Υ_{con} is a subset of Υ and q is also inside Υ_{con}, q's result is $\mathcal{RS}$. If q exits Υ_{con}, but remains inside Υ, a new conservative safe zone Υ'_{con} is constructed. By definition (Definition 7.1), Υ'_{con} is a subset of Υ, and the result of q is $\mathcal{RS}$. If q exits Υ_{con}, and also outside

Υ, a new real safe zone Υ^{new} is calculated that has result $\mathcal{RS}^{new}$. Then a new conservative safe zone Υ_{con}^{new} is constructed based on Υ^{new}, and the above arguments are again applied. □

The underlying motivation for considering the user of conservative safe zones is that these have a smaller description, i.e., has fewer influence objects, than the corresponding real safe zone, which implies a lower cost per communication round. Although the area of a conservative safe zone is also smaller than that of the corresponding real safe zone, it can be expected that relatively few additional communication rounds are incurred. This is because a real safe zone may well have influence objects that contribute lines or arcs that will never be reached by the user and thus are wasted.

We study two approaches for constructing conservative safe zones. Section 7.1 proposes a conservative circle, and Section 7.2 introduces a conservative freeform. Moreover, for the sake of ease use, we propose parameter-free variants of the two types of conservative safe zones, in Section 7.3. Our experimental results (in Section 8.4) show that: (i) the conservative freeform achieves good performance in scenarios with low overhead per communication round, and (ii) the conservative circle performs well in scenarios with high overhead per communication round.

7.1. Conservative Circle

The conservative circle centers at the query location and has a predefined radius. It conservatively approximates the real safe zone while using a simpler representation. Figure 21(a) shows an example where the shaded region is the safe zone of the query q and the two dashed circles are conservative circles with different radiuses. In an actual application, only one conservative circle is needed. Instead of sending the real safe zone, which is defined by influence objects (e.g., p_2, p_3, p_4, and p_5 in Figure 21(a)), only the influence objects whose Apollonius circles intersect the conservative circle are sent.

Formally, let $\mathit{InfObj}(q)$ be the influence object set of query q. Let $C_{p_i,*}$ denote the Apollonius circle of influence object p_i and one of the top-k results of q, and let $\mathcal{C}$ is the conservative circle. The message size per communication round is defined as:

$$\mathit{Size}(q, \mathcal{C}) = |\{p_i \in \mathit{InfObj}(q) \ \mid \ C_{p_i,*} \cap \mathcal{C} \neq \emptyset\}| + 1. \tag{30}$$

Given a query, it is an interesting problem to determine the radius of the conservative circle. We aim at finding an optimized value of the radius that has low message size and low communication frequency.

A smaller radius results in a lower message size, but probably also a higher communication frequency. As shown in Figure 21(a), the smaller conservative circle does not intersect any boundary of the safe zone, and hence only this circle is a valid conservative approximation of the safe zone. However, this circle is small, which means that the probability of the query leaving the conservative safe zone is high, thus resulting in more frequent communication. There exists a tradeoff, i.e., a smaller radius gives lower message size, while larger radius gives a lower communication frequency. Consider the example in Figure 21(a) again. If we choose the bigger conservative circle (dashed circle), it intersects the Apollonius circle C_{p_4,p_1}. Then both the conservative circle and the influence object p_4 are sent to the client. Comparing with the smaller conservative circle, it has one more influence object to be sent, but has a larger area.

We introduce an integer parameter τ that indicates the radius of the conservative circle. To do so recall that the boundaries of the real safe zone are defined by influence objects. We sort the influence objects in terms of their minimum border distances. The parameter τ implies that a circle with radius equal to the τ^{th} minimum border distance is used. In Figure 21(a), the smaller conservative circle is used when $\tau = 1$. The bigger conservative circles is used when $\tau = 2$. How different values of τ affect the performance is evaluated in Section 8.4.1.

When the query exits a safe zone given by a conservative circle, it sends an update of its location to the server. Then the server computes a new conservative circle based on the current query location. In Figure 21(b), when q moves to q', the new conservative circle is the dashed circle centered at q' ($\tau = 1$).

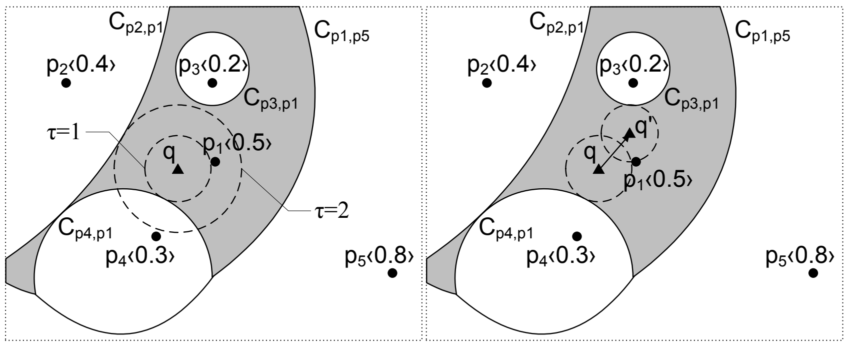


Fig. 21. Conservative Circle, Top-1 Object: p_1

7.2. Conservative Freeform

The conservative freeform is a conservative safe zone without any holes. It is obtained by cropping the real safe zone using the tangent lines of the holes in the real safe zone. Each tangent line is perpendicular to the line formed by the query position and the center of its hole. Formally, let Υ be the real safe zone of a query q and let O_{C_i} denote the center of any hole C_i. The tangent line l_{q,C_i} of C_i satisfies $l_{q,C_i} \perp \overline{q\ O_{C_i}}$. Line l_{q,C_i} divides the space into two half planes. Let L_{q,C_i} be the half plane containing q. The conservative freeform Υ_{con} is defined as:

$$\Upsilon_{con} = \bigcap_{i=1}^{n} \{L_{q,O_i}\} \cap \Upsilon, \tag{31}$$

where n is the number of holes in Υ.

As an example, the shaded region is the real safe zone Υ of the query q, and there are three holes C_{p_2,p_1}, C_{p_3,p_1}, and C_{p_4,p_1}. The three dashed lines $l_{q,C_{p_3,p_1}}$, $l_{q,C_{p_2,p_1}}$, and $l_{q,C_{p_4,p_1}}$ are the tangent lines of the three holes. The shaded region enclosed by the dashed lines and C_{p_1,p_5} is the conservative freeform Υ_{con} to be sent to the client.

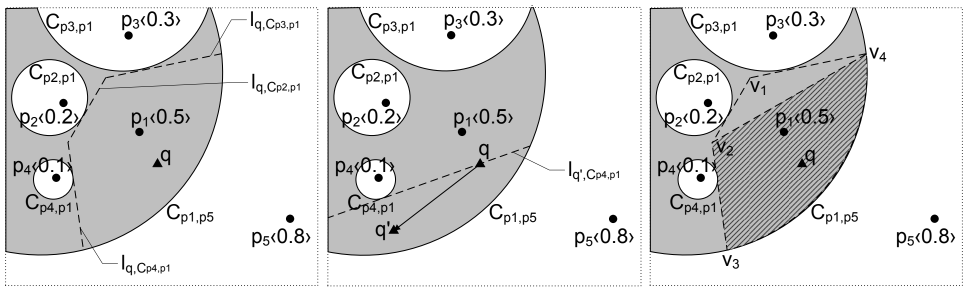


Fig. 22. Conservative Freeform, Top-1 Object: p_1

When the query exits a conservative freeform Υ_{con}, it sends a location update to the server. Then the server computes a new conservative freeform based on the current query location. Figure 22(b) shows the new conservative freeform (the shaded region enclosed by the dashed line $l_{q',C_{p_4,p_1}}$ and C_{p_1,p_5}) when q moved to q'. In this example, the tangent lines of C_{p_2,p_1} and C_{p_3,p_1} are disregarded, since they do not intersect the current conservative freeform.

The conservative freeform is represented by several vertices and arcs that are sent to the client. Sometimes, the conservative freeform may contain many vertices, yielding a large message size. We propose to shrink the conservative freeform to avoid some vertices. A parameter μ is introduced to control the shrinkage.

$$\mu = \frac{Area(\Upsilon'_{con})}{Area(\Upsilon_{con})}, \tag{32}$$

where $Area(\Upsilon_{con})$ is the area of the conservative freeform and Υ'_{con} denotes the conservative freeform after removing some vertices.

We propose a greedy removal approach to remove the vertices of the conservative freeform. Algorithm 4 shows the pseudo code. We keep on removing the vertex such that the remaining freeform has larger area than when removing any other vertex and such that its area is at least $\mu\%$ of the freeform before shrinking. In Figure 22(c), vertex v_1 is removed when $\mu = 0.75$. The shaded, striped region is the conservative freeform after shrinking. How different values of μ affect the performance is evaluated in Section 8.4.2.

Algorithm 4 ConFreeform (q, Υ_{con}, μ)

```
1: $v_{rem} \leftarrow null$;
2: repeat
3:     $area \leftarrow -\infty$;
4:     for each vertex $v_i$ in $\Upsilon_{con}$ do
5:         $\Upsilon'_{con} \leftarrow \Upsilon_{con} \setminus v_i$;
6:         if $area < Area(\Upsilon'_{con})$ and $q \in \Upsilon'_{con}$ then
7:             $area \leftarrow Area(\Upsilon'_{con})$;
8:             $v_{rem} \leftarrow v_i$;
9:     if $\frac{area}{Area(\Upsilon_{con})} \geq \mu$ then
10:        $\Upsilon_{con} \leftarrow \Upsilon_{con} \setminus v_{rem}$;
11:    else
12:        $v_{rem} \leftarrow null$;
13: until $v_{rem}$ is $null$
```

7.3. Parameter-Free Conservative Safe zone

Both the conservative circle and the conservative freeform techniques require parameters (τ and μ), which might not be easy to tune manually. To improve usability, we proceed to render both safe zone computation techniques parameter free. We first define the goodness of a conservative safe zone Υ_{con} as

$$Goodness(\Upsilon_{con}) = \frac{Area(\Upsilon_{con})}{\#InfObj}, \tag{33}$$

where $\#InfObj$ is the number of influence objects defining the conservative safe zone Υ_{con}. The parameter-free versions then aim to maximize $Goodness(\Upsilon_{con})$, i.e., finding a conservative safe zone whose area is large (achieving low communication frequency) and whose number of influence object is small (achieving low communication cost).

For the parameter-free conservative circle, parameter τ is encapsulated inside the algorithm (Algorithm 5), i.e., for each possible value of τ, we compute the goodness of the corresponding conservative circle and return the τ of the conservative circle with the largest goodness. Note that parameter τ implies the τ^{th} minimum border distance and the influence object set is finite, so that the domain of τ is finite.

Algorithm 5 GoodConCircle (q, Υ_{con})

```
1: PriorityQueue $queue \leftarrow null$;
2: $G \leftarrow 0$;
3: $radius \leftarrow 0$;
4: for each influence object $o$ of $\Upsilon_{con}$ do
5:     $d \leftarrow$ minimum border distance of $o$ from $q$;
6:     $queue.add(d)$;          ▷ Minimum border distances are sorted in ascending order.
7: while $queue$ is not empty do
8:     $d \leftarrow queue.pop()$;
9:     if $\pi d^2 < G$ then
10:        break;
11:    $c \leftarrow$ the number of influence objects that intersect the circle centered at $q$ with radius $d$;
12:    if $G < \pi d^2/c$ then
13:        $G \leftarrow \pi d^2/c$
14:        $radius \leftarrow d$;
15: return $radius$;
```

For the parameter-free conservative freeform, we propose a greedy algorithm, shown in Algorithm 6, that finds a good conservative safe zone. In each iteration (lines 3–10), we remove the vertex such that the goodness of the conservative safe zone is maximized. Across iterations (lines 11–12), we keep the conservative safe zone with the biggest goodness. We evaluate the performance of parameter-free conservative safe zones in the empirical study.

Algorithm 6 GoodConFreeform (q, Υ_{con})

```
1: $\Upsilon^*_{con} \leftarrow \Upsilon_{con}$;
2: $G_g \leftarrow 0$;                                   ▷ Global goodness.
3: while $Area(\Upsilon_{con}) > 0$ do
4:     $G_l \leftarrow 0$;                               ▷ Local goodness.
5:     for each vertex $v_i$ in $\Upsilon_{con}$ do
6:         $\Upsilon'_{con} \leftarrow \Upsilon_{con} \setminus v_i$;
7:         if $G_l < Goodness(\Upsilon'_{con})$ and $q \in \Upsilon'_{con}$ then
8:             $G_l \leftarrow Goodness(\Upsilon'_{con})$;
9:             $v_{rem} \leftarrow v_i$;
10:    $\Upsilon_{con} \leftarrow \Upsilon_{con} \setminus v_{rem}$;
11:    if $G_g < G_l$ then
12:        $\Upsilon^*_{con} \leftarrow \Upsilon_{con}$;
13: return $\Upsilon^*_{con}$;
```

8. EMPIRICAL STUDY

We conduct empirical studies to evaluate our proposals. Section 8.1 presents datasets, queries, parameters, and the platform used in experiments. The proposed algorithms for the processing of MkSK queries are evaluated in Section 8.2. Sections 8.3 and 8.4 study the techniques of efficient next safe zone computation and the effectiveness of conservative safe zones. Section 8.5 summarizes the findings.

8.1. 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 contains points of interest (e.g., ATMs, hotels, stores) in Europe, and in USA and Canada, respectively (www.pocketgpsworld.com). Table V offers additional detail. We normalize object locations to fit a square domain with side length $10,000$ meters.

Table V. 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 [Brinkhoff 2002] 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 based on the word distribution of the data sets used. Specifically, we first randomly pick an object and then randomly choose consecutive words from the document of the object as query keywords. 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 the 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 client- and server-side 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.

8.2. Performance of Continuous MkSK Queries

We study the proposed methods under varying settings.

Results for the Real Data Sets.

Table VI 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 the 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 and GN in subsequent experiments. We drop MSK-v as it is always worse than MSK-uv. Table VII shows detailed information of the indexes (IR-trees) on the four datasets. The size of the index on dataset GN is the biggest, around 3.8GB. The whole index is not buffered by the operating system, given 4GB physical main memory.

Varying the Speed of Moving Queries.

This experiment compares IBD, MSK-uv, and MSK-uvr when varying the speed of the moving queries on datasets EURO and GN. As the speed increases, the server elapsed time (Figures 23(a)

Table VI. Performance per Timestamp on Real Data Sets

Data Set		HOTEL	GN	EURO	USCAN
Server elapsed time (milliseconds)	IBD	2.673	12327.555	182.606	1.454
	MSK-v	0.549	374.921	9.350	0.534
	MSK-uv	0.531	257.403	7.897	0.523
	MSK-uvr	0.582	855.629	9.215	0.554
Server I/O cost (page accesses)	IBD	9.288	2183.079	163.435	6.159
	MSK-v	0.503	30.168	8.014	0.622
	MSK-uv	0.459	19.909	7.340	0.568
	MSK-uvr	0.459	19.909	7.340	0.568
Client elapsed time (milliseconds)	IBD	0.002	0.225	0.014	0.002
	MSK-v	0.002	0.223	0.016	0.002
	MSK-uv	0.004	0.223	0.015	0.003
	MSK-uvr	0.003	0.017	0.003	0.001
Communication cost (objects)	IBD	0.611	23.635	2.589	0.332
	MSK-v	0.606	23.105	2.575	0.332
	MSK-uv	0.623	22.277	2.583	0.338
	MSK-uvr	0.489	2.597	0.841	0.279
Communication frequency	All methods	0.036	0.091	0.056	0.025

Table VII. Index Statistics

data set	index size (MB)	# of levels in the IR-tree	# of non-leaf nodes	# of leaf nodes
HOTEL	4.05	2	1	168
GN	3934.51	3	148	16771
EURO	142.37	3	12	1423
USCAN	12.23	2	1	190

and 24(a)), the client elapsed time (Figures 23(b) and 24(b)), and the communication cost (Figures 23(c) and 24(c)) exhibit increasing trends. The query leaves the safe zone faster at a higher speed. Indeed, Figures 23(d) and 24(d) show that the communication frequency increases with increasing speed, leading to larger elapsed time and communication cost. MSK-uv and MSK-uvr outperform IBD significantly.

Varying the Number of Query Keywords.
We consider again IBD, MSK-uv, and MSK-uvr on datasets EURO and GN. As expected, the MSK variants outperform IBD significantly in terms of server elapsed time as shown in Figures 25(a) and 26(a). Figures 25(b), 25(c), 26(b), and 26(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^-. Figures 25(d) and 26(d) show that the communication frequency increases slightly with the number of keywords. This is because larger query keyword sets yield smaller safe zones.

Varying k.
We evaluate the performance of our proposals on datasets EURO and GN when varying k. Figures 27(a) and 28(a) shows that MSK-uv and MSK-uvr have much lower server elapsed time than IBD. Figures 27(b), 28(b), 27(c), and 28(c) show that MSK-uvr has the lowest client elapsed time and communication cost. The communication frequency increases slightly with k increases (see Figures 27(d) and 28(d)), as a larger k yields smaller safe zones.

Scalability.
This experiment studies the scalability of the proposed approaches beyond main memory by increasing the dataset size. We generated 4 datasets containing approximately 3.6M, 5.4M, 7.2M, and 9M objects by shifting GN spatially a given number of times. The intuition is that if GN is realistic for one region, it can also be considered as realistic for another region. So suppose that GN occupies a cell in a grid. We then create 1, 2, 3, and 4 adjacent grid cells by copying GN to those cells. The

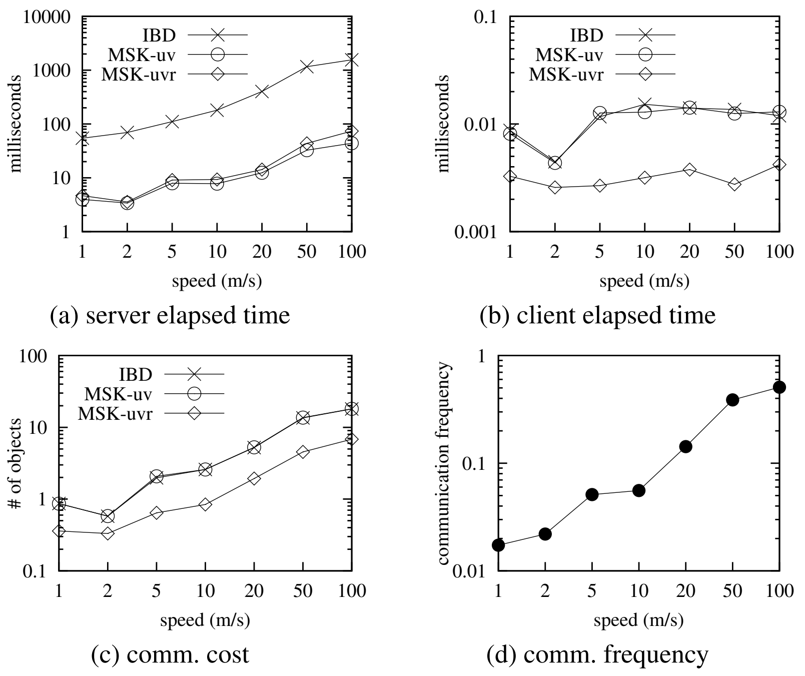


Fig. 23. Performance per Timestamp, Varying Speed, on EURO

sizes of the indexes of these 4 generated datasets are 13.55GB (0.26GB R-tree + 13.29GB inverted files), 29.02GB (0.39GB R-tree + 28.63GB inverted files), 50.36GB (0.52GB R-tree + 49.84GB inverted files), and 77.58GB (0.65GB R-tree + 76.93GB inverted files), respectively. We use a Core2 Quad CPU Q8400 @ 2.66GHz with 4GB memory for this experiment and report averages for five queries. Figure 29 shows that the server elapsed time exhibits a moderately increasing trend, which suggests good scalability for out-of-core data. However, the client elapsed time, the communication cost, and the communication frequency are insensitive to the dataset size.

Effect of f on the Upgraded Voronoi Cell Optimization.
Figure 30 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 31 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.

8.3. Evaluating the Safe Zone Computation Enhancements

We study the performance of the three proposed techniques presented in Section 6 that aim to increase the performance of safe zone computation, and we compare them with the MSK-uvr algorithm. The algorithm with all three enhancements is called **MSK-uvr-SP**.

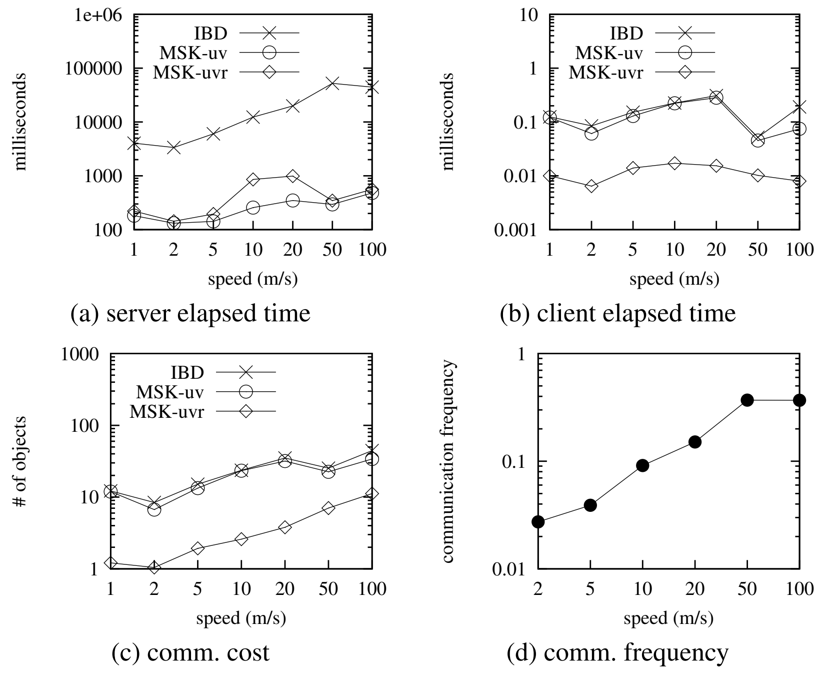


Fig. 24. Performance per Timestamp, Varying Speed, on GN

As defaults, we set the number k of results to 10, the number of query keywords to 3, and the client speed to 10 m/s. Figures 32, 33, and 34 show the elapsed time and I/O cost when varying the speed of the moving queries, the number of keywords, and k, respectively.

MSK-uvr-SP and MSK-uvr differ in terms of safe zone computation at the server. Hence, MSK-uvr-SP has the same client elapsed time, communication cost, and communication frequency as does MSK-uvr (shown in Figures 23, 25, and 27). This experiment evaluates the performance of MSK-uvr-SP and MSK-uvr in terms of server elapsed time and server I/O cost. As expected, the server elapsed time is proportional to the server I/O cost. As the client speed increases, the server elapsed time and the server I/O cost of both MSK-uvr-SP and MSK-uvr increase. This is so because if the client leaves the safe zone more quickly, more requests are sent to the server, and thus more safe zone and result computations occur at the server. Thanks to the techniques incorporated into MSK-uvr-SP, its server elapsed time and server I/O cost increase more slowly than do those of MSK-uvr. The same trend can be observed as the number of keywords and k increase. All in all, MSK-uvr-SP with all three speedup techniques achieves significant improvements when compared with MSK-uvr.

8.4. Evaluating Conservative Safe Zones

We proceed to study the performance of the proposals for conservative safe zones presented in Section 7. We tune the parameter used in the proposals so that each achieves its best performance, i.e., τ for the conservative circle and μ for the conservative freeform, and we compare with their parameter-free versions and the MSK-uvr algorithm. As defaults, we set the number k of results to 10, the number of query keywords to 3, and the client speed to 10 m/s.

8.4.1. Conservative Circle. Figure 35 shows the performance of the conservative circle (**ConCircle**) when varying τ and its parameter-free version (**ConCircle***). The server elapsed times of the ConCircle, the ConCircle* and the MSK-uvr algorithm are very similar in Figure 35(a), in-

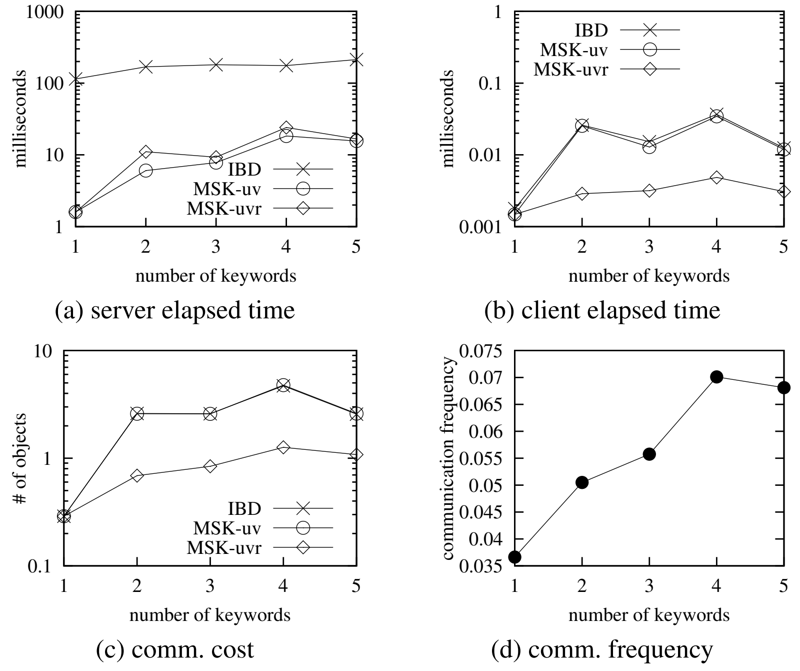


Fig. 25. Performance per Timestamp, Varying the Number of Query Keywords, on EURO

dicating that the additional computational costs of ConCircle and ConCircle* are low. Figure 35(b) shows that the client elapsed time of the MSk-uvr is reduced by around 70% when using the conservative circle, due to the reason that the number of transferred influence objects is reduced by around 65%, as shown in Figure 35(c).

As τ increases, Figures 35(b) and 35(c) show a slightly increasing trend, since the conservative circle becomes bigger, i.e., the number of influence objects to be sent increases, and the client elapsed time also increases. When τ is small, the communication frequency of ConCircle is higher than that of MSK-uvr, as shown in Figure 35(d). This is because the conservative circle is small, making the probability of leaving the conservative safe zone high. As τ increases, the communication frequency of ConCircle drops significantly, approaching the communication frequency of MSK-uvr. The optimal value of τ found in this experiment is between 7 and 9. Both ConCircle and ConCircle* yield low client elapsed time, less influence objects to be sent, and low communication frequency.

8.4.2. Conservative Freeform. Figure 36 shows the performance of the conservative freeform algorithm (**ConFreeform**) when varying μ and its parameter-free version (**ConFreeform***). The server elapsed time of ConFreeform, ConFreeform*, and MSK-uvr algorithm are very close in Figure 36(a), indicating that the additional computation costs of ConFreeform and ConFreeform* are low. Figure 36(b) shows that the client elapsed time of MSk-uvr is reduced by around 70% when using the conservative freeform, due to the reason that the number of transferred influence objects is reduced by around 40%, as shown in Figure 36(c). As μ decreases, Figures 36(b) and 36(c) show a slightly decreasing trend, since more vertices are removed and the conservative freeform becomes smaller, i.e., the number of influence objects to be sent decreases and the client elapsed time also decreases. When μ is small, the communication frequency of ConFreeform is higher than that of MSK-uvr, as shown in Figure 36(d). This is because the conservative freeform is small and the probability of leaving the conservative safe zone is high. As μ increases, the communication frequency of

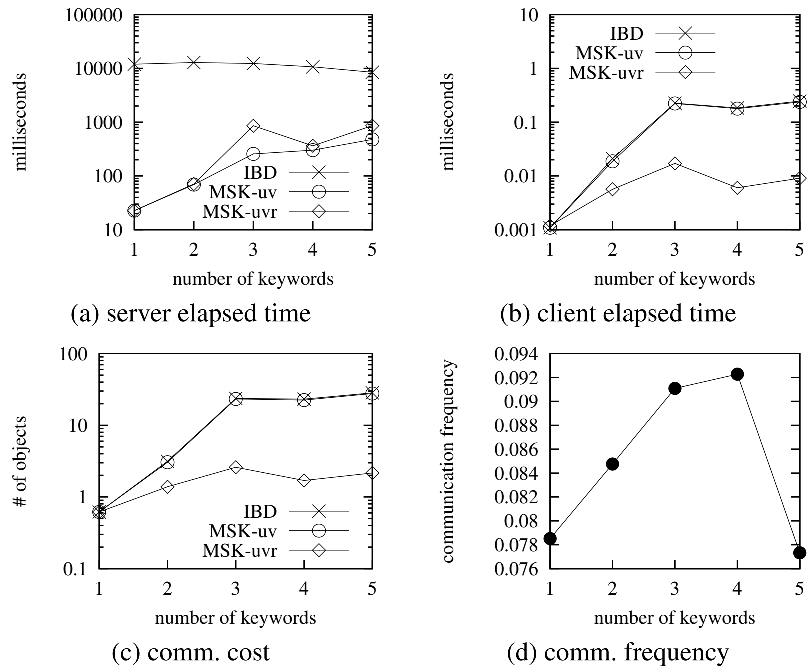


Fig. 26. Performance per Timestamp, Varying the Number of Query Keywords, on GN

ConFreeform drops significantly, approaching the communication frequency of MSK-uvr. The optimal value of μ is found to be 0.95. Both ConFreeform and ConFreeform* yield low client elapsed time, less influence objects to be sent, and low communication frequency.

8.4.3. Recommendation. In applications where the overhead per communication round is low, use of the conservative freeform is recomended. A small μ can be chosen to achieve low client elapsed time and low communication cost. If the overhead per communication round is high, use of the conservative circle is recommended. A large τ can be chosen to obtain low communication frequency while also keeping the client elapsed time and communication cost low. The parameter-free conservative safe zones offer increased usability.

8.5. Summary

The experimental study shows that MSK-uv achieves the lowest server elapsed time, while MSK-uvr is only slightly worse than MSK-uv in terms of server elapsed time. The added cost is due to the recursive refinement of I^-. MSK-uvr is the best in terms of communication cost, since it reduces the size of the I^- set significantly. Hence, MSK-uvr achieves the lowest client elapsed time. The safe zone computation enhancements improve the performance of MSK-uvr significantly, and conservative safe zones achieve significantly lower client elapsed time and communication cost at the expense of only slightly higher server elapsed time and communication frequency.

9. RELATED WORK

The MkSK query is a combination of two kinds of queries, i.e., the spatial keyword query and the spatial moving query. Spatial keyword query processing and spatial moving query processing have been well studied separately. This section covers existing work related to these two kinds of queries.

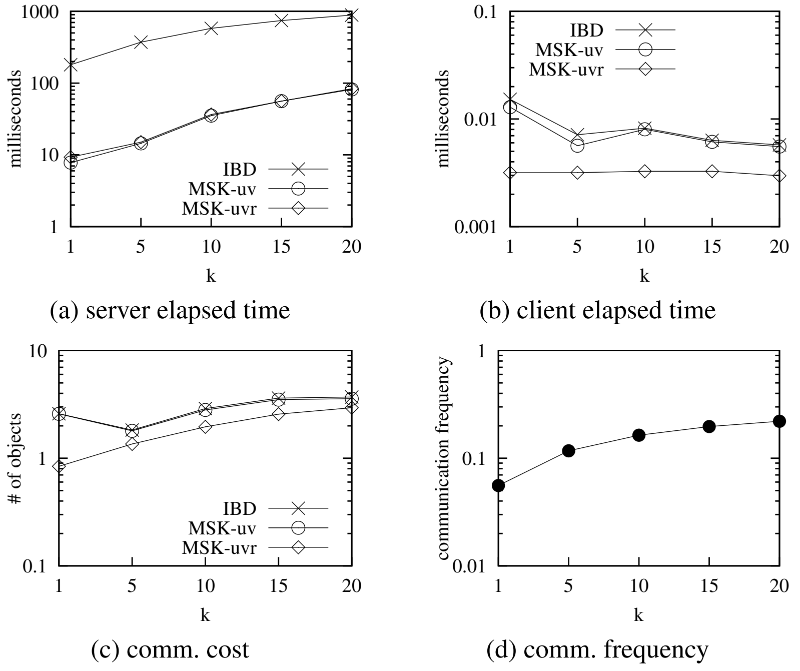


Fig. 27. Performance per Timestamp, Varying k, on EURO

9.1. Spatial Keyword Queries

A spatial keyword query retrieves the best object(s) with respect to a given location and a set of keywords. Cao et al. [2012] provides an introduction to the subject, and Chen et al. [2012] provides an experimental comparison of indexes. Efficient implementation of spatial keyword queries, with varying semantics, has been studied.

Zhou et al. [2005] use a hybrid index structure that integrates inverted files and R*-trees for computing both textual and location aware queries. Three different index variations are studied: (1) inverted file and R*-tree double index, (2) first inverted file then R*-tree, (3) first R*-tree then inverted file. All the three variants are used for the processing of spatial keyword queries in two phases, i.e., either first spatial filtering or first keyword filtering. The IR^2-tree [Felipe et al. 2008] is another hybrid index that targets spatial keyword queries. It combines an R-tree with superimposed text signatures. However, it is applicable only when the keywords serve as a Boolean filter. In the IR-tree [Cong et al. 2009; Li et al. 2011; Wu et al. 2012], each node in the R-tree is augmented with inverted files. It 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 [Zhang et al. 2009; Zhang et al. 2010], which augment each node with a bitmap and MBRs for keywords, are used for computing this query.

Cao et al. [2010] propose the concept of prestige-based relevance to capture both the textual relevance of an object to a query and the effects of nearby objects. They introduce a new type of query, the Location-aware top-k Prestige-based Text retrieval (LkPT) query that retrieves the top-k spatial web objects ranked according to both prestige-based relevance and location proximity.

Cao et al. [2011] study a variant of the spatial keyword query that retrieves a group of spatial web objects such that the group's keywords cover the query's keywords and such that objects are the nearest to the query location and have the lowest inter-object distances.

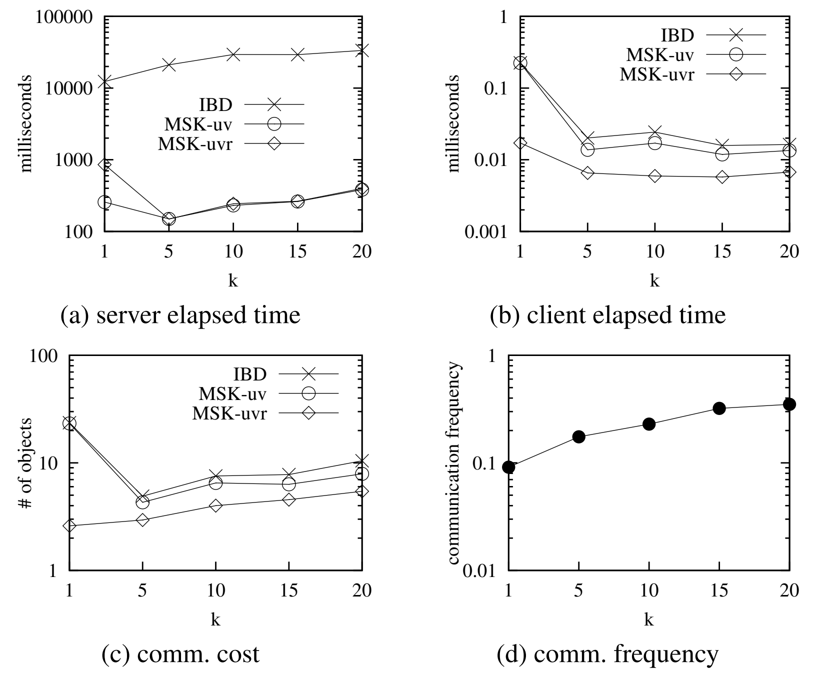


Fig. 28. Performance per Timestamp, Varying k, on GN

Lu et al. [2011] consider the Reverse Spatial Textual k Nearest Neighbor (RSTkNN) query, i.e., finding objects that take the query object as one of their k most spatial-textual similar objects. They propose a hybrid index tree called IUR-tree (Intersection-Union R-Tree) that effectively combines location proximity with textual similarity. Based on the IUR-tree, they design a branch-and-bound search algorithm and a cluster enhancement to improve the performance of the IUR-tree.

Fan et al. [2012] study the spatio-textual similarity search on regions of interest (ROIs) that contain region-based spatial information and textual descriptions. Result ROIs overlap the query region spatially and are similar to the query text. Li, Feng, and Xu [2012] study the use of direction in the spatial keyword search. Their direction-aware spatial keyword query finds k nearest neighbors of the query that are in the search direction and contain all input keywords. Li, Yao, Tang, and Hadjieleftheriou [2012] study a spatial approximate string (SAS) query that is a range query augmented with a string similarity search predicate in both Euclidean space and road networks.

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.

9.2. Spatial Moving Queries

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

In moving query processing, Benetis et al. [2002; 2006] study persistent and continuous nearest and reverse nearest neighbor query processing for for moving users and data points, Tao et al. [2002] compute nearest neighbors for each possible query point on a line segment, and Iwerks et al. [2006] study how to maintain a users future k nearest neighbors when the users velocity changes. These works all model the users movement as a linear function. Given the known future movement of a user, a time-parameterized query [Tao and Papadias 2002] retrieves the current result along with a time interval indicating its validity.

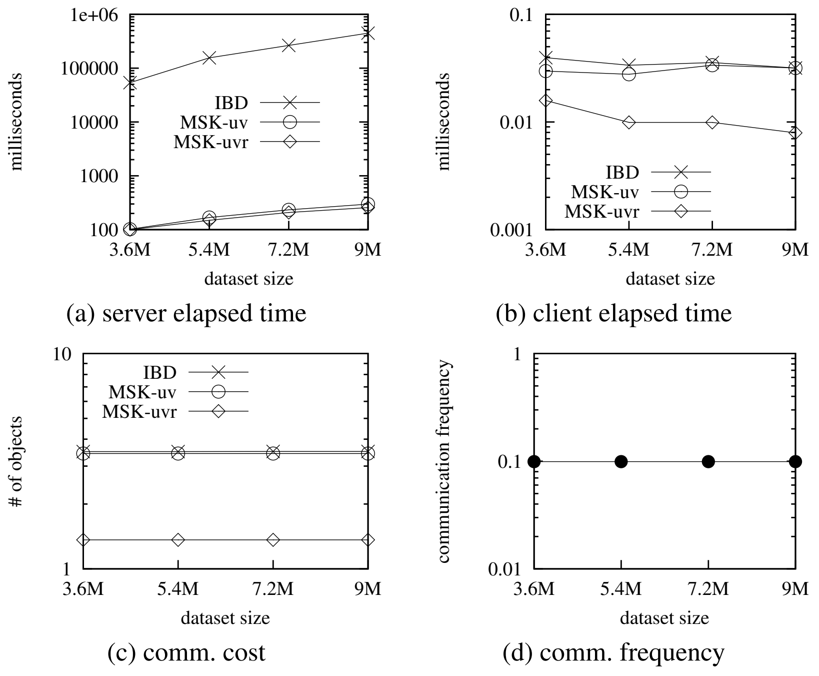


Fig. 29. Performance per Timestamp, Varying Dataset Size

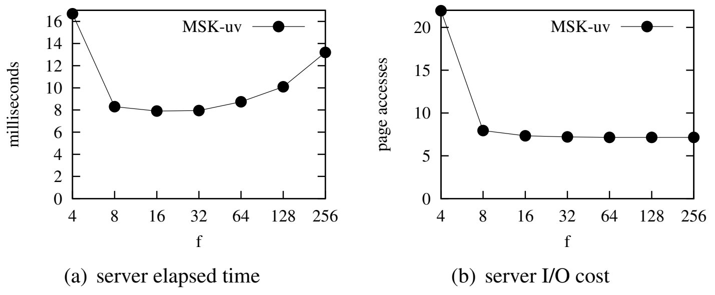
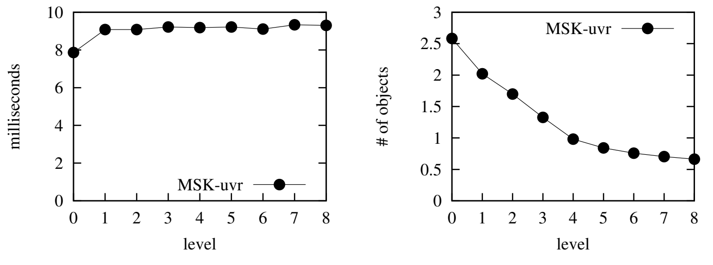


Fig. 30. Performance per Timestamp, Varying f, on EURO

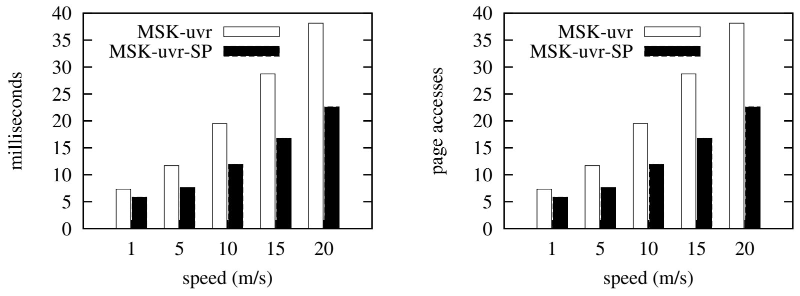
When the user's movement is not known in advance, a buffering approach [Song and Roussopoulos 2001] retrieves the user's $k + \Delta k$ nearest neighbors and uses them to derive a buffer region for the user. While moving within the buffer region, the user's k nearest neighbors can be kept up to date by using only the $k + \Delta k$ nearest neighbors. However, it is not easy to find the optimal value of Δk in practice, and that value used strongly influences the communication frequency and the number of objects transferred per communication round.

The safe zone approach [Zhang et al. 2003; Nutanong et al. 2008; Cheema et al. 2010] 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



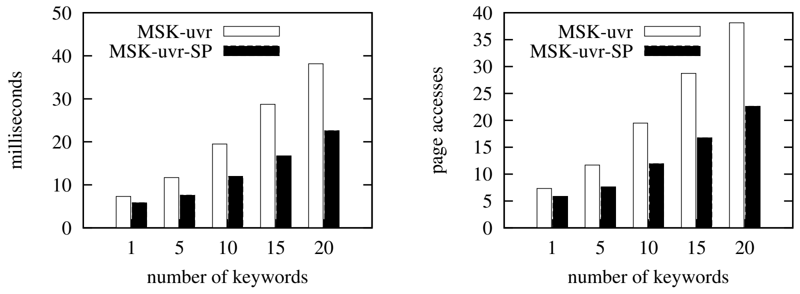
(a) server elapsed time (b) communication cost

Fig. 31. Performance per Timestamp, Varying l_{max}, on EURO



(a) server elapsed time (b) server I/O cost

Fig. 32. Performance per Timestamp, Varying Speed, on EURO



(a) server elapsed time (b) server I/O cost

Fig. 33. Performance per Timestamp, Varying # of Keywords, on EURO

cell [Zhang et al. 2003], or a V*-diagram [Nutanong et al. 2008]. The safe zone of a moving circular range query can be captured by a set of objects that affect the safe zone [Cheema et al. 2010].

Gao et al. [2009] study continuous nearest neighbor queries in the presence of obstacles, i.e., continuous obstructed nearest neighbor (CONN) queries. They consider the impact of obstacles on the distance between objects. Given a data set P, an obstacle set O, and a query line segment q in a two-dimensional space, a CONN query retrieves the nearest neighbor of each point on q according to the obstructed distance, i.e., the shortest path between them without crossing any obstacle. In addition, they extend their solution to handle the continuous obstructed k-nearest neighbor (COkNN) query that finds the k nearest neighbors to every point along q based on obstructed distances.

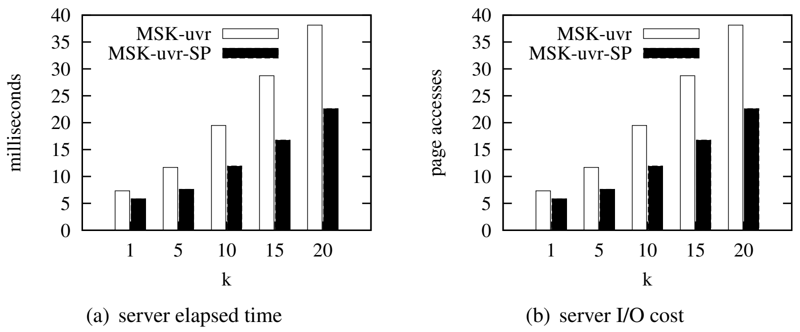


Fig. 34. Performance per Timestamp, Varying k, on EURO

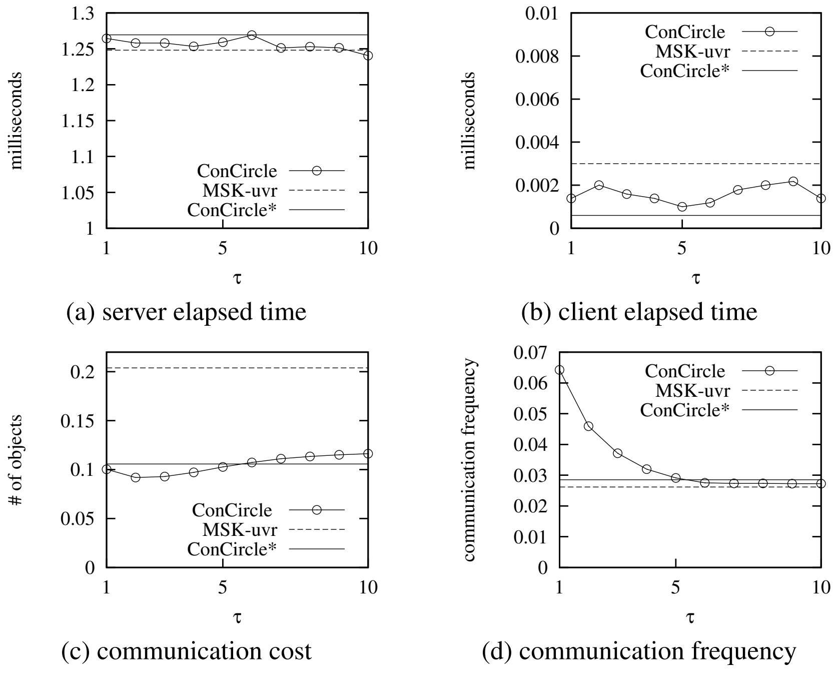


Fig. 35. Performance of Conservative Circle, Varying τ, on EURO

Chen et al. [2009] address the problem of monitoring the k nearest neighbors to a dynamically changing path in road networks, namely the k-Path Nearest Neighbor query (k-PNN). Given a destination of a user, the query returns the kNN with respect to the shortest path connecting the destination and the user's current location, and it thus provides a list of nearest candidates for reference by considering the whole journey ahead. As the user is moving and may not always follow the shortest path, the query path keeps changing. Chen et al. [2009] propose a three-phase Best-first Network Expansion (BNE) algorithm for monitoring the k-PNN and the corresponding shortest path. In its search phase, the BNE finds the shortest path to the destination, during which a candidate set that guarantees to include the k-PNN is generated at the same time. Then in its verification phase, a heuristic algorithm examines the candidates' exact distances to the query path, and it achieves

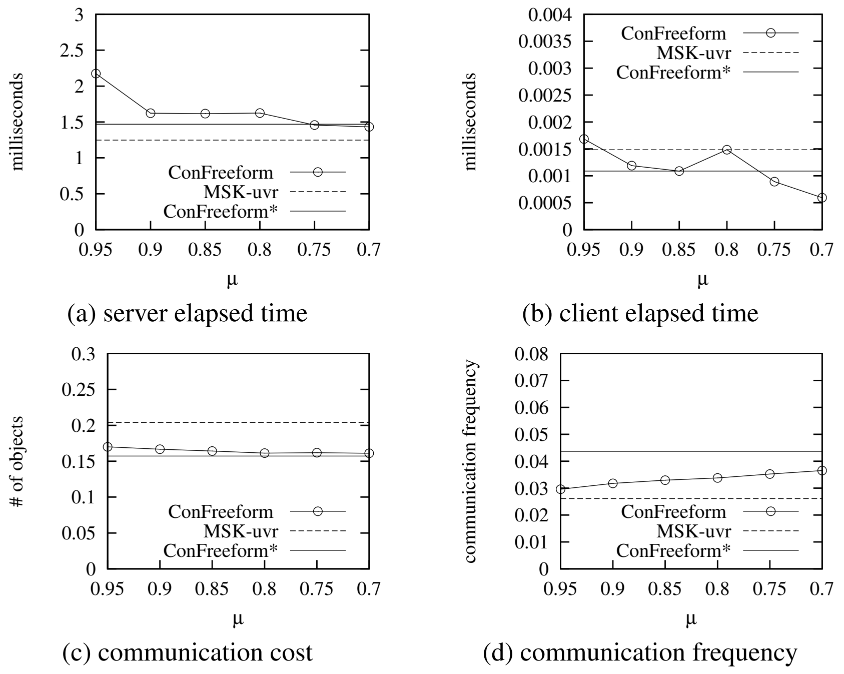


Fig. 36. Performance of Conservative Freeform, Varying μ, on EURO

significant reduction in the number of visited nodes. The monitoring phase deals with computing update locations as well as refreshing the k-PNN for different user movements.

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.

10. CONCLUSIONS AND RESEARCH DIRECTIONS

10.1. 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. We present techniques that aim to compute the next safe zone efficiently, and we present two types of conservative safe zones, along with parameter-free variants, that aim to reduce the 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. To understand the effectiveness of the proposed safe zones, we study analytically the expected area of a safe zone, which indicates on average for how long a safe zone remains valid, and we study the expected number of influence objects needed to define a safe zone, which gives an estimate of the average communication cost. The analytical study is verified by extensive empirical studies.

10.2. Research Directions

In future work, it is of interest to examine the processing of moving range spatial keyword queries. The term “range” can be described in multiple ways. It could be “Independent Thresholding,” i.e., imposing two separate thresholds on the spatial range and the text relevancy. A pedestrian may use a "within walking distance (5 km)" range and may ask for objects with text relevancy higher than 0.5 within that spatial range. If only few objects are found, the threshold can be lowered by using a slider in a graphical user interface. The objects can be ordered according their text relevancy. The "range" could also be "Weighted Thresholding," i.e., imposing a single threshold on the weighted distance (Equation 1) directly. For example, objects with weighted distances higher than 0.5 are then returned. By issuing such a query, the user can have an overview of how relevant objects are distributed within a certain "range." A different safe zone and algorithms are needed for the processing of moving range spatial keyword queries.

It is also of interest to examine the processing of moving top-k spatial keyword queries over spatial networks. When users' movements are constrained by road networks, the Euclidian distance in Equation 1 becomes network distance. The network distance makes the shapes of dominant regions and representation of a safe zone different. Inspired by Kolahdouzan and Shahabi [2004] who propose a network Voronoi diagram for k nearest neighbor search in spatial network databases, a spatial network-based MW-Voronoi diagram may be needed for the processing of moving top-k spatial keyword queries over spatial networks. In addition to the influence objects used for representing a safe zone, some points on road networks may also determine the border of the safe zone. This direction of research offers non-trivial challenges to the algorithms that are applicable to Euclidean space.

Another possible research direction is to consider the future path of a moving query when computing the safe zone of the current result. If and when such paths are available, it may be beneficial to construct a conservative safe zone that is refined along the near-future path and is coarse elsewhere. An alternative goodness criteria may be considered instead of Equation 33, i.e., $Goodness(\Upsilon_{con}) = EscapeDistance(\Upsilon_{con})/\#InfObj$, where the estimation of $EscapeDistance(\Upsilon_{con})$ takes the knowledge of the path into account.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library. It provides a detailed derivation of Equation 26 and a proof of Lemma 5.1.

ACKNOWLEDGMENTS

This research was supported in part by Geocrowd, funded by the European Commission under grant agreement number 264994. Man Lung Yiu was supported by the grant PolyU 5302/12E from Hong Kong RGC.

REFERENCES

AURENHAMMER, F. AND EDELSBRUNNER, H. 1984. An optimal algorithm for constructing the weighted Voronoi diagram in the plane. *Pattern Recognition 17,* 2, 51–57.

BENETIS, R., JENSEN, C. S., KARCIAUSKAS, G., AND SALTENIS, S. 2002. Nearest neighbor and reverse nearest neighbor queries for moving objects. In *IDEAS*. 44–53.

BENETIS, R., JENSEN, C. S., KARCIAUSKAS, G., AND SALTENIS, S. 2006. Nearest neighbor and reverse nearest neighbor queries for moving objects. *VLDB J. 15,* 3, 229–249.

BRINKHOFF, T. 2002. A framework for generating network-based moving objects. *GeoInformatica 6,* 2, 153–180.

CAO, X., CHEN, L., CONG, G., JENSEN, C. S., QU, Q., SKOVSGAARD, A., WU, D., AND YIU, M. L. 2011. Spatial keyword querying. In *ER*. 16–29.

CAO, X., CONG, G., AND JENSEN, C. S. 2010. Retrieving top-k prestige-based relevant spatial web objects. *PVLDB 3,* 1, 373–384.

CAO, X., CONG, G., JENSEN, C. S., AND OOI, B. C. 2011. Collective spatial keyword querying. In *SIGMOD Conference*. 373–384.

CHEEMA, M. A., BRANKOVIC, L., LIN, X., ZHANG, W., AND WANG, W. 2010. Multi-guarded safe zone: An effective technique to monitor moving circular range queries. In *ICDE*. 189–200.

CHEN, L., CONG, G., JENSEN, C. S., AND, WU, D. 2012. Spatial keyword query processing: an experimental evaluation. *PVLDB*, to appear.

CHEN, Y.-Y., SUEL, T., AND MARKOWETZ, A. 2006. Efficient query processing in geographic web search engines. In *SIGMOD Conference*. 277–288.

CHEN, Z., SHEN, H. T., ZHOU, X., AND YU, J. X. 2009. Monitoring path nearest neighbor in road networks. In *SIGMOD Conference*. 591–602.

CONG, G., JENSEN, C. S., AND WU, D. 2009. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB 2,* 1, 337–348.

DIGGLE, P. J. 2003. *Statistical Analysis of Spatial Point Patterns, 2nd edition*. ARNOLD.

DURELL, C. V. 1961. *Modern geometry: the straight line and circle*. Macmillan.

FAN, J., LI, G., ZHOU, L., CHEN, S., AND HU, J. 2012. SEAL: spatio-textual similarity search. *PVLDB 5,* 9, 824–835.

FELIPE, I. D., HRISTIDIS, V., AND RISHE, N. 2008. Keyword search on spatial databases. In *ICDE*. 656–665.

GAMBINI, R., HUFF, D. L., AND JENKS, G. F. 1968. Geometric properties of market areas. *Regional Science 20,* 1, 85–92.

GAO, Y. AND ZHENG, B. 2009. Continuous obstructed nearest neighbor queries in spatial databases. In *SIGMOD Conference*. 577–590.

GUTTMAN, A. 1984. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*. 47–57.

HUFF, D. L. 1973. The delineation of a national system of planning regions on the basis of urban spheres of influence. *Regional Science 7,* 3, 323–329.

IWERKS, G. S., SAMET, H., AND SMITH, K. P. 2006. Maintenance of k-nn and spatial join queries on continuously moving points. *ACM TODS 31,* 2, 485–536.

KOLAHDOUZAN, M. R. AND SHAHABI, C. 2004. Voronoi-based k nearest neighbor search for spatial network databases. In *VLDB*. 840–851.

LI, F., YAO, B., TANG, M., AND HADJIELEFTHERIOU, M. 2012. Spatial approximate string search. *TKDE* preprint.

LI, G., FENG, J., AND XU, J. 2012. DESKS: direction-aware spatial keyword search. In *ICDE*. 474–485.

LI, Z., LEE, K. C. K., ZHENG, B., LEE, W.-C., LEE, D. L., AND WANG, X. 2011. IR-tree: an efficient index for geographic document search. *TKDE 23,* 4, 585–599.

LU, J., LU, Y., AND CONG, G. 2011. Reverse spatial and textual k nearest neighbor search. In *SIGMOD Conference*. 349–360.

MU, L. 2004. Polygon characterization with the multiplicatively weighted Voronoi diagram. *The Professional Geographer 56,* 2, 223–239.

NUTANONG, S., ZHANG, R., TANIN, E., AND KULIK, L. 2008. The v*-diagram: a query-dependent approach to moving knn queries. *PVLDB 1,* 1, 1095–1106.

OKABE, A., BOOTS, B., SUGIHARA, K., AND CHIU, D. S. N. 2000. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd Edition*. Wiley.

SALTON, G. AND MCGILL, M. J. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill.

SONG, Z. AND ROUSSOPOULOS, N. 2001. K-nearest neighbor search for moving query point. In *SSTD*. 79–96.

TAO, Y. AND PAPADIAS, D. 2002. Time-parameterized queries in spatio-temporal databases. In *SIGMOD Conference*. 334–345.

TAO, Y., PAPADIAS, D., AND SHEN, Q. 2002. Continuous nearest neighbor search. In *VLDB*. 287–298.

WU, D., CONG, G., AND JENSEN, C. S. 2012. A framework for efficient spatial web object retrieval. *VLDB J. 21,* 6, 797–822.

WU, D., YIU, M. L., JENSEN, C. S., AND CONG, G. 2011. Efficient continuously moving top-k spatial keyword query processing. In *ICDE*. 541–552.

ZHANG, D., CHEE, Y. M., MONDAL, A., TUNG, A. K. H., AND KITSUREGAWA, M. 2009. Keyword search in spatial databases: Towards searching by document. In *ICDE*. 688–699.

ZHANG, D., OOI, B. C., AND TUNG, A. K. H. 2010. Locating mapped resources in web 2.0. In *ICDE*. 521–532.

ZHANG, J., ZHU, M., PAPADIAS, D., TAO, Y., AND LEE, D. L. 2003. Location-based spatial queries. In *SIGMOD Conference*. 443–454.

ZHOU, Y., XIE, X., WANG, C., GONG, Y., AND MA, W.-Y. 2005. Hybrid index structures for location-based web search. In *CIKM*. 155–162.

ZOBEL, J. AND MOFFAT, A. 2006. Inverted files for text search engines. *ACM Comput. Surv. 38,* 2.

Online Appendix to: Moving Spatial Keyword Queries: Formulation, Methods, and Analysis

DINGMING WU, Hong Kong Baptist University
MAN LUNG YIU, Hong Kong Polytechnic University
CHRISTIAN S. JENSEN, Aarhus University

In this appendix, we provide detailed derivation and proofs of some equations and lemmas in the paper.

A. DERIVATION OF EQUATION 26

In Equation 25, o is the origin. The probability

$$\mathbf{P}(\|o\, p_j\| > \frac{w_j}{w_i} r) = Pr(N(A_j) = 0) = e^{-\rho|A_j|} = e^{-\rho\pi \frac{w_j^2}{w_i^2} r^2} \quad (34)$$

means that A_j is the circle centered at o with radius $\frac{w_j}{w_i} r$ containing no object in $\mathcal{D}$. The probability $\mathbf{P}(\|o\, p_i\| \in [r, r+dr])$ in Equation 25 is given in Lemma A.1.

LEMMA A.1.
[Deriving the probability $\mathbf{P}(\|o\, p_i\| \in [r, r+dr])$] *Let o be the origin and p_i be a random object. The probability that the distance between o and p_i falls into range $[r, r+dr]$ is:*

$$\mathbf{P}(\|o\, p_i\| \in [r, r+dr]) = 2\rho\pi r e^{-\rho\pi r^2}\, dr.$$

PROOF. The probability $\mathbf{P}(\|o\, p_i\| \in [r, r+dr])$ can be interpreted as the probability that there is only one object p_i in a thin ring ΔA with thickness dr defined by the outer circle centered at o with radius $r + dr$ and the inner circle centered at o with radius r multiplied by the probability that the circle A centered at o with radius r contains no object, i.e.,

$$\begin{aligned} \mathbf{P}(\|o\, p_i\| \in [r, r+dr]) &= Pr(N(\Delta A) = 1) Pr(N(A) = 0) \\ &= \rho|\Delta A| e^{-\rho|\Delta A|} e^{-\rho|A|}, \end{aligned} \quad (35)$$

where $|\Delta A| = \pi(r+dr)^2 - \pi r^2$ and $|A| = \pi r^2$. The area of ΔA can be estimated by $2\pi r\, dr$, since

$$\begin{aligned} \lim_{dr\to 0} \frac{|\Delta A|}{2\pi r\, dr} &= \lim_{dr\to 0} \frac{\pi(r+dr)^2 - \pi r^2}{2\pi r\, dr} \quad (36) \\ &= \lim_{dr\to 0} \frac{2\pi r\, dr + \pi(dr)^2}{2\pi r\, dr} \\ &= \lim_{dr\to 0} \frac{2\pi r + 2\pi\, dr}{2\pi r} \quad \text{(L'Hopital's rule)} \\ &= 1. \end{aligned}$$

Hence, we have

$$\mathbf{P}(\|o\, p_i\| \in [r, r+dr]) = 2\rho\pi r e^{-(\rho\pi r^2 + 2\pi r\, dr)}\, dr = 2\rho\pi r e^{-\rho\pi r^2}\, dr, \quad (37)$$

where $\rho\pi r^2 + 2\pi r\, dr$ is approximated by $\rho\pi r^2$ when $dr \to 0$. □

© 2012 ACM 0362-5915/2012/-ART00 $10.00
DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

B. PROOF OF LEMMA 5.1

PROOF.

If $p_j \in D_i^+$, i.e., $w_j > w_i$, the condition of p_j not being one influence object of p_i is:

$$\|o_{p_i,p_j}\ p_i\| \leq r_{p_i,p_j} - r(w_i).$$

Since $\|o_{p_i,p_j}\ p_i\| = r_{p_i,p_j} - bord_{min}(p_i, C_{p_i,p_j})$ (defined in Lemma 3.3), we have:

$$bord_{min}(p_i, C_{p_i,p_j}) = \frac{w_i}{w_i + w_j}\|p_i\ p_j\| \geq r(w_i),$$

i.e.,

$$\|p_i\ p_j\| \geq \frac{w_i + w_j}{w_i} r(w_i). \tag{38}$$

If $p_j \in D_i^o$, i.e., $w_j = w_i$, the condition of p_j not being one influence object of p_i is:

$$\|p_i\ p_j\| \geq 2r(w_i). \tag{39}$$

If $p_j \in D_i^-$, i.e., $w_j < w_i$, the condition of p_j not being one influence object of p_i is:

$$\|o_{p_j,p_i}\ p_i\| \geq r_{p_j,p_i} + r(w_i).$$

Since $\|o_{p_j,p_i}\ p_i\| = r_{p_j,p_i} + bord_{min}(p_i, C_{p_j,p_i})$, we have:

$$bord_{min}(p_i, C_{p_j,p_i}) = \frac{w_i}{w_i + w_j}\|p_i\ p_j\| \geq r(w_i),$$

i.e.,

$$\|p_i\ p_j\| \geq \frac{w_i + w_j}{w_i} r(w_i). \tag{40}$$

Having Equations 38, 39, and 40, we conclude that the condition of any object p_j (with either bigger, equal, or smaller weight) not being one influence object of p_i is as stated by the lemma. □