Efficient Safe-Region Construction for Moving Top-K Spatial Keyword Queries

Weihuang Huang[†] Guoliang Li[†] Kian-Lee Tan[‡] Jianhua Feng[†]
[†]Department of Computer Science and Technology, Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing, China
[‡]Department of Computer Science, School of Computing, National University of Singapore , Singapore
huangwh10@mails.thu.edu.cn; {liguoliang,fengjh}@tsinghua.edu.cn; tankl@comp.nus.edu.sg

ABSTRACT

Many real-world applications have requirements to support moving spatial keyword queries. For example a tourist looks for top-k "`seafood restaurants`" while walking in a city. She will continuously issue moving queries. However existing spatial keyword search methods focus on static queries and it calls for new effective techniques to support moving queries efficiently. In this paper we propose an effective method to support moving top-k spatial keyword queries. In addition to finding top-k answers of a moving query, we also calculate a *safe region* such that if a new query with a location falling in the safe region, we can directly use the answer set to answer the query. To this end, we propose an effective model to represent the safe region and devise efficient search algorithms to compute the safe region. We have implemented our method and experimental results on real datasets show that our method achieves high efficiency and outperforms existing methods significantly.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Moving Top-k Spatial Keyword Queries, Safe Region

1. INTRODUCTION

Location based services (LBS) have attracted significant attention from both industry and academic communities in recent years, thanks to the modern mobile phones and new Internet technologies. Many existing systems provide users with location-aware search experiences based on users' location which can be easily gotten from GPS devices equipped in modern mobile phones.

Recently there are many studies on location based services and most of them address the spatial keyword search problem [9, 4, 2, 7], which, given a set of spatio-textual objects with a location and textual description (e.g., points of interest and geo-tagged documents) and a top-k spatial keyword query with a location and a set of keywords, finds top-k relevant answers. However they focus primarily on static queries and cannot support moving queries efficiently. Notice that many real-world applications have requirements to support moving spatial keyword queries. For example, a housewife is driving to a supermarket and may want to find the top-3 "`car parking places`" near the supermarket. Since she is driving, her query location is continuously changing. As another example, a tourist looking for the top-2 "`seafood restaurants`" while walking in a city will require a moving query. Although we can extend existing methods to support moving queries by repeatedly issuing multiple queries, these methods have the following limitations. First, it increases the communication cost between the client (the user who issues the query) and the server (the system that provides the search service), and also wastes the bandwidth in transmission. Second, it aggravates the system burden due to issuing multiple repeated queries.

To address this problem, in this paper we emphasize on efficiently supporting moving top-k spatial keyword queries. We adopt a client-server model. The client is moving and continuously issues a spatial keyword query to the server. The server returns the top-k answers of the query, as well as a `safe region` of the answer set (We will formally define the safe region in Section 2). Then before the client issues a new query at another location, it will first check whether the new location is still in the `safe region`. If yes, it can reuse the answer set; otherwise the client needs to issue a query with the new location to the server. Obviously our method not only avoids unnecessary communication cost but also reduces the system burden. Notice that although Wu et al. [17] studied the moving top-k spatial keyword query, they used an ad-hoc ranking function (see Section 2). In contrast, we use a widely adopted ranking function [2, 11].

Different from existing studies [2, 4] which focus on computing the answer set of a query, we emphasize on how to compute the safe region. There are several research challenges. First, how to represent the safe region? Traditional studies on spatial data (without textual description) use Voronoi diagrams to represent the safe region, which can be pre-computed and materialized for efficient online query processing. However safe region in our problem depends on query keywords and cannot be materialized. To address this issue, we propose an effective representation model in Section 3. Second, how to compute the safe region? For a query issued to the server, we need to find its answer set

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

and compute the safe region simultaneously. Thus the time for computing the safe region cannot be large. To address this issue, we develop efficient algorithms in Section 4. To summarize, we make the following contributions.

- We propose an effective model to represent the safe region of a moving top-k spatial keyword query.
- We devise efficient incremental algorithms to efficiently compute the safe region.
- We develop effective pruning techniques to reduce the computation time on the server and the verification time on the client.
- Experimental results show that our method achieves high efficiency and outperforms existing methods.

The rest of the paper is organized as follows. We first formulate our problem in Section 2. A model to represent the safe region is discussed in Section 3. We devise efficient algorithms to compute the safe region in Section 4. Section 5 extends our techniques to support moving top-k queries. We conduct experimental results in Section 6 and review related work in Section 7. Section 8 concludes the paper.

2. PRELIMINARY

2.1 Problem Formulation

We adopt a client-server model. The server contains a set of spatio-textual objects, O. Each object $o \in O$ contains a location o_s and textual description o_t, denoted by $o = (o_s, o_t)$. In the paper we consider two-dimensional space, and use x-coordinate $o_s.x$ and y-coordination $o_s.y$ to denote a location o_s. We use a set of terms to denote o_t. The client is moving and continuously issues a top-k spatial keyword query q to the server. Query q consists of a query location q_s, a set of keywords q_t, and an integer k to restrict the result size, denoted by $q = (q_s, q_t, k)$. For ease of presentation, we first define the answer of a top-k spatial keyword query.

Definition 1 (Top-k Spatial Keyword Query). *Given an object set O and a query q, the answer of a top-k spatial keyword query is a subset of O, $\mathcal{A}$, such that*

(1) The size of $\mathcal{A}$ is k, i.e., $|\mathcal{A}| = k$, and

(2) $\forall o^* \in \mathcal{A}, \forall o \in O - \mathcal{A}$, $\texttt{score}(q, o^*) \leq \texttt{score}(q, o)$,

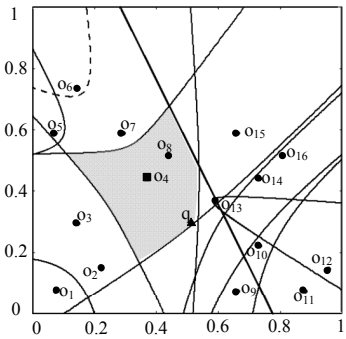
where `score` *is a ranking function to evaluate the relevance between a query and an object. The smaller the value is, the more relevant is the object to the query. We will define the ranking function in Section 2.2.*

For example, there is a dataset O as shown in Figure 1 and a top-k spatial keyword query $q = ((0.515, 0.294), \{\texttt{Chinese}, \texttt{restarant}\}, 1)$. We can get $\mathcal{A} = \{o_4\}$ under the ranking function defined in Section 2.2. If $k = 2$ with the same q_s and q_t, then the answer set $\mathcal{A} = \{o_4, o_9\}$.

For a moving top-k spatial keyword query q, in addition to finding the answer set $\mathcal{A}$, we also need to find a *safe region* for query q. Note that the safe region depends on the keyword set q_t and k, and we denote it by $\mathcal{R}(q_t, k)$. For a query $q' = (q'_s, q_t, k)$ with a new location q'_s, if $q'_s \in \mathcal{R}(q_t, k)$, the answer set of q' is the same as that of q. Thus we can use the answer set $\mathcal{A}$ to answer q'. If the context is clear, $\mathcal{R}(q_t, k)$ and $\mathcal{R}$ are used interchangeably. Next we formally define the safe region $\mathcal{R}$.

Definition 2 (Safe Region). *Given an object set O and a query q, the safe region of query q is*

$$\mathcal{R} = \{q'_s | \forall o^* \in \mathcal{A}, \forall o \in O - \mathcal{A}, \texttt{score}(q', o^*) \leq \texttt{score}(q', o)\},$$



o_1:$\langle k_1, 1\rangle, \langle k_2, 2\rangle, \langle k_3, 3\rangle, \langle k_4, 4\rangle$
o_2:$\langle k_2, 3\rangle, \langle k_4, 1\rangle, \langle k_5, 1\rangle$
o_3:$\langle k_1, 1\rangle, \langle k_3, 3\rangle, \langle k_5, 2\rangle$
o_4:$\langle k_1, 2\rangle, \langle k_2, 1\rangle, \langle k_3, 4\rangle, \langle k_4, 1\rangle$
o_5:$\langle k_3, 1\rangle, \langle k_5, 4\rangle$
o_6:$\langle k_1, 1\rangle, \langle k_2, 3\rangle, \langle k_4, 3\rangle, \langle k_5, 3\rangle$
o_7:$\langle k_1, 3\rangle, \langle k_2, 2\rangle, \langle k_3, 3\rangle, \langle k_4, 2\rangle$
o_8:$\langle k_1, 1\rangle, \langle k_4, 3\rangle$
o_9:$\langle k_1, 9\rangle, \langle k_2, 1\rangle, \langle k_3, 8\rangle, \langle k_5, 3\rangle$
o_{10}:$\langle k_1, 2\rangle, \langle k_2, 1\rangle, \langle k_4, 1\rangle$
o_{11}:$\langle k_2, 2\rangle, \langle k_3, 4\rangle, \langle k_4, 1\rangle, \langle k_5, 3\rangle$
o_{12}:$\langle k_3, 1\rangle, \langle k_4, 2\rangle, \langle k_5, 2\rangle$
o_{13}:$\langle k_2, 2\rangle, \langle k_3, 3\rangle, \langle k_4, 3\rangle, \langle k_5, 2\rangle$
o_{14}:$\langle k_1, 4\rangle, \langle k_3, 4\rangle, \langle k_5, 2\rangle$
o_{15}:$\langle k_1, 9\rangle, \langle k_2, 1\rangle, \langle k_3, 6\rangle, \langle k_4, 4\rangle$
o_{16}:$\langle k_2, 2\rangle, \langle k_4, 1\rangle$

Figure 1: Dataset(k_1:Chinese,k_2:Franch,k_3:restarant, k_4:seafood,k_5:pastry, numbers in brackets are tf)

where $q' = (q'_s, q_t, k)$ and q'_s is any location.

For example, in Figure 1, the shadow region in the center is the safe region for $q_t = \{\texttt{Chinese}, \texttt{restarant}\}$ and $k = 1$.

Then based on the answer set $\mathcal{A}$ and the safe region $\mathcal{R}$, we define the answer of a moving top-k spatial keyword query.

Definition 3. *(Moving Top-k Spatial Keyword Query) Given an object set O and a moving top-k spatial keyword query $q = (q_s, q_t, k)$. As the query is moving, for each new location q'_s, its answer is $\langle \mathcal{A}, \mathcal{R} \rangle$, where $\mathcal{A}$ is the top-k result set of $q' = (q'_s, q_t, k)$ and $\mathcal{R}$ is the corresponding safe region.*

In the client-server model, for a query q, the client first checks whether $q_s \in \mathcal{R}$. If yes, $\mathcal{A}$ is the answer of query q; otherwise the client submits the query q to the server which returns the answer set and the safe region of the query. In our example in Figure 1, the server will return the answer set $\mathcal{A} = \{o_4\}$ and a safe region $\mathcal{R}$ (shaded in the Figure). Next the client updates its location and checks whether it is still in $\mathcal{R}$. If yes, $\{o_4\}$ is still the answer; otherwise the answer changes and the client issues a new query to the server.

Existing studies [2] focus on top-k spatial keyword queries and they propose efficient algorithms to compute the answer set $\mathcal{A}$. In this paper we emphasize on how to compute the safe region $\mathcal{R}$ and address the research challenges as discussed in Section 1.

2.2 Ranking Function

Given a query q and an object o, to compute their ranking score $\texttt{score}(q, o)$, we combine their spatial proximity between o_s and q_s, denoted by $\texttt{dist}(o_s, q_s)$, and their textual relevancy between o_t and q_t, denoted by $\texttt{text}(o_t, q_t)$. The ranking function is defined as follows.

$$\texttt{score}(q, o) = \alpha \cdot \texttt{dist}(q_s, o_s) + (1 - \alpha) \cdot (1 - \texttt{text}(q_t, o_t)) \quad (1)$$

where α is a tuning parameter to trade-off the importance between the spatial distance and textual relevancy. Notice that in the ranking function, we normalize $\texttt{dist}(q_s, o_s)$ and $\texttt{text}(q_t, o_t)$ to $[0, 1]$ using their possible maximum values. In the paper, we use the Euclidean distance (function `dist`) to compute the spatial distance (`dist`) between two locations, and adopt the well-known TFIDF function to evaluate the textual relevancy (`text`) as follows.

$$\texttt{text}(q_t, o_t) = \sum_{t \in q_t} (\mathsf{tf}(t, o_t) \times \mathsf{idf}(t)) \quad (2)$$

where $\mathsf{tf}(t, o_t)$ is the term frequency of term t in o_t and $\mathsf{idf}(t)$ is the inverse document frequency of term t, i.e., the

ratio of the number of objects in O to that of objects whose textual descriptions contain t. For ease of presentation, if the context is clear, we use $\texttt{text}(q_t, o_t)$ and $\texttt{text}(q, o)$, and $\texttt{dist}(q_t, o_t)$ and $\texttt{dist}(q, o)$ interchangeably.

For the example in Figure 1, assume $\alpha = \frac{2}{3}$. $\texttt{dist}(q, o_1) = 0.49$, $\texttt{text}(q, o_1) = 0.1$, and $\texttt{score}(q, o_1) = 0.63$; $\texttt{dist}(q, o_4) = 0.21$, $\texttt{text}(q, o_4) = 0.75$ and $\texttt{score}(q, o_4) = 0.22$; $\texttt{dist}(q, o_9) = 0.27$, $\texttt{text}(q, o_9) = 0.85$ and $\texttt{score}(q, o_9) = 0.23$. The top-1 answer is $\{o_4\}$ and the top-2 answer is $\{o_4, o_9\}$.

Notice that our ranking function is widely adopted in existing studies [2, 11]. Although Wu et al. [17] studied the moving top-k spatial keyword query problem, they used a very ad-hoc ranking function as defined below.

$$\texttt{rank}(q, o) = \frac{\texttt{dist}(q, o)}{\texttt{text}(q, o)}. \quad (3)$$

Obviously our ranking function is more general. We will discuss how to support their ranking function and experimentally compare with their method in Section 6.

3. REPRESENTATION MODEL FOR TOP-1 QUERIES

In this section, we first introduce a concept *dominant region* in Section 3.1, and then based on the definition we discuss how to represent the safe region in Section 3.2.

3.1 Dominant Region

Given a query q and two objects, o^* and o, the *dominant region* of o^* to o is a region such that if q is in the region, o^* is a better answer than o, as defined below.

DEFINITION 4. *Given a query $q = (q_s, q_t, k)$, the dominant region of o^* to o is:*

$$D_{o^*,o} = \{q'_s | \texttt{score}(q', o^*) \leq \texttt{score}(q', o)\} \quad (4)$$

where $q' = (q'_s, q_t, k)$.

For example, in Figure 1, the dominant region of o_4 to o_6 is the region outside the dashed line. That is if a query q is in the region, o_4 is a better answer than o_6.

Next we deduce how to represent the dominant region. We first introduce two notations.

$$\begin{aligned} \Delta d &= \texttt{dist}(q', o^*) - \texttt{dist}(q', o), \\ \Delta t &= \frac{1-\alpha}{\alpha}\left(\texttt{text}(q', o^*) - \texttt{text}(q', o)\right). \end{aligned} \quad (5)$$

Based on Equation 1, $\texttt{score}(q', o^*) \leq \texttt{score}(q', o)$ if and only if $\Delta d \leq \Delta t$. Thus we have $D_{o^*,o} = \{q'_s | \Delta d \leq \Delta t\}$, according to Definition 4. Based on the relationship between Δt and $\texttt{dist}(o^*, o)$, we can determine the shape of the dominant region as follows (also shown in Table 1 and Figure 2).

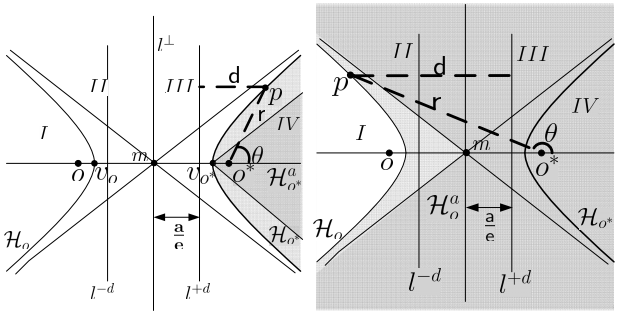
Case 1: $\Delta t \geq \texttt{dist}(o^*, o)$. Based on the triangle inequality, $\Delta d = \texttt{dist}(q', o^*) - \texttt{dist}(q', o) \leq \texttt{dist}(o^*, o)$. If $\Delta t \geq \texttt{dist}(o^*, o)$, $\Delta d \leq \Delta t$ is always true, thus the dominant region is the whole plane.

Case 2: $\Delta t < -\texttt{dist}(o^*, o)$. Based on the triangle inequality, $-\texttt{dist}(o^*, o) \leq \Delta d = \texttt{dist}(q', o^*) - \texttt{dist}(q', o)$. If $\Delta t < -\texttt{dist}(o^*, o)$, $\Delta d \leq \Delta t$ is always false, thus the dominant region is empty.

Case 3: $\Delta t = -\texttt{dist}(o^*, o)$. Based on Case 2, only the points on the half-line starting from o^* and with direction from o to o^* (denoted by $\mathcal{H}_{o^*,\overrightarrow{oo^*}}$) satisfy $-\texttt{dist}(o^*, o) \leq \Delta d \leq \Delta t = -\texttt{dist}(o^*, o)$, thus the dominant region is half-line $\mathcal{H}_{o^*,\overrightarrow{oo^*}}$.

Table 1: Dominant Region

Cases	Dominant Region
$\Delta t \geq \texttt{dist}(o^*, o)$	whole plane
$\Delta t = -\texttt{dist}(o^*, o)$	half-line $\mathcal{H}_{o^*,\overrightarrow{oo^*}}$
$\Delta t < -\texttt{dist}(o^*, o)$	empty
$\Delta t = 0$	half-plane ($III \cup IV$ in Figure 2)
$0 < \Delta t < \texttt{dist}(o^*, o)$	outside $\mathcal{H}_o$ ($II \cup III \cup IV$ in Figure 2)
$-\texttt{dist}(o^*, o) < \Delta t < 0$	inside $\mathcal{H}_{o^*}$ (IV in Figure 2)



(a) $-\texttt{dist}(o^*, o) < \Delta t < 0$ (b) $0 < \Delta t < \texttt{dist}(o^*, o)$

Figure 2: Dominant Region

Case 4: $\Delta t = 0$. Obviously the locus of $\Delta d = 0$ is the perpendicular bisector of segment $\overline{o^*o}$ which partitions the space into two half planes. The dominant region for $\Delta d \leq \Delta t$ is the half plane that contains o^*.

Case 5: $0 < \Delta t < \texttt{dist}(o^*, o)$. The locus of points satisfying $|\Delta d| = |\texttt{dist}(q, o^*) - \texttt{dist}(q, o)| = |\Delta t|$ is a hyperbola with o^* and o as its two focuses as proved in Lemma 1. Let $\mathcal{H}$ denote the hyperbola satisfying $|\Delta d| = |\Delta t|$, $\mathcal{H}_{o^*}$ denote the branch nearby the focus o^* (i.e., satisfying $\Delta d = -\Delta t$), and $\mathcal{H}_o$ denote the branch nearby the focus o (i.e., satisfying $\Delta d = \Delta t$). $\mathcal{L}^{\perp}$ is the perpendicular bisector of segment $\overline{o^*o}$. $\mathcal{L}^{\perp}$, $\mathcal{H}_{o^*}$, $\mathcal{H}_o$ divide the plane into four regions I, II, III, IV, as shown in Figure 2. Obviously in this case, the dominant region for $\Delta d \leq \Delta t$ is the region outside $\mathcal{H}_o$ (i.e., region $II \cup III \cup IV$ in Figure 2).

Case 6: $-\texttt{dist}(o^*, o) < \Delta t < 0$. Similar to Case 5, the dominant region for $\Delta d \leq \Delta t$ is the region inside $\mathcal{H}_{o^*}$ (i.e., region IV in Figure 2).

LEMMA 1. *If $0 < \Delta t < \texttt{dist}(o^*, o)$ or $-\texttt{dist}(o^*, o) < \Delta t < 0$, the locus of points satisfying $|\Delta d| = |\Delta t|$ is a hyperbola with o^* and o as its two focuses. The locus of points satisfying $\Delta d = \Delta t$ is the branch $\mathcal{H}_o$ and that for $\Delta d = -\Delta t$ is the branch $\mathcal{H}_{o^*}$.*

PROOF. Due to space constraints, we omit all proofs. □

As illustrated in Figure 2, $\mathcal{H}$ is the hyperbola derived from o^* and o. m is the the center of $\mathcal{H}$, i.e., the midpoint of segment $\overline{o^*o}$. Next we give the basic parameters and their values of hyperbola $\mathcal{H}$.

Focuses : o^* and o.

Vertices : The two nearest points located at the two branches: v^*_o in $\mathcal{H}_{o^*}$ and v_o in $\mathcal{H}_o$.

Semi-focal Length : The distance from one focus to the center, denoted by $\texttt{c}$. $\texttt{c} = \frac{dist(o^*,o)}{2}$.

Semi-major Length : The distance from one vertex to the center, denoted by $\texttt{a}$. $\texttt{a} = \frac{1}{2}|\Delta t|$.

Eccentricity : The parameter determines the shape of a curve (for all conic curves), denoted by $\texttt{e}$. $\texttt{e} = \frac{\texttt{c}}{\texttt{a}}$.

3.2 Safe Region

In this section, we discuss how to represent the safe region. Given a query q, suppose its top-1 answer is o^*. Based on the definition of the dominant region, obviously the safe region is the intersection of the dominant region of o^* to all other objects, i.e., $\mathcal{R} = \cap_{o \neq o^*} D_{o^*,o}$ as formalized in Lemma 2.

LEMMA 2. *Given a query q, suppose its top-1 answer is o^*. $\mathcal{R} = \cap_{o \neq o^*} D_{o^*,o}$.*

In this way, after finding the top-1 answer o^*, we compute the dominant region of o^* to each object $o \in O - \{o^*\}$. Then we compute the intersection of these dominant regions. Notice that the safe region is always not empty and q_s is in $\mathcal{R}$ as formalized in Lemma 3.

LEMMA 3. *For a query q, the safe region $\mathcal{R}$ is always not empty and q_s is in $\mathcal{R}$.*

Some readers may find that in Case 3 ($\Delta t < -\texttt{dist}(o^*, o)$), the dominant region is empty. Interestingly, we can prove that for the best answer o^*, there is no such case, that is, $\Delta t \geq -\texttt{dist}(o^*, o)$.

LEMMA 4. *Given a query q and its best answer o^*, for any object $o \in O - \{o^*\}$, $\Delta t \geq -\texttt{dist}(o^*, o)$.*

Next we discuss how to represent the safe region. Recall the different shapes of the dominant regions. It is easy to represent planes, half-planes, and half-lines. However it is not easy to represent a hyperbola. A hyperbola is a type of conic section. In a Cartesian coordinate system, it is usually represented by a second-degree polynomial or a matrix. However it is very expensive to use such methods and it is also inefficient to compute the intersection. To address this issue, we introduce two alternative methods.

Using Polygons to Approximate a Hyperbola: Since the distance between a hyperbola and its asymptotes tends to 0 when they approach infinity, we can use asymptotes to approximate hyperbola.

For the dominant region inside $\mathcal{H}_{o^*}$ (the shaded region in Figure 2(a)), consider the two half-lines starting from the vertex v_{o^*} with directions the same as the two asymptotes. Let $\mathcal{H}^a_{o^*}$ denote the region inside the two half-lines (the shaded polygon region in Figure 2(a)). Obviously $\mathcal{H}^a_{o^*}$ is in the dominant region and we use it to approximate the region inside $\mathcal{H}_{o^*}$. Similarly for the dominant region outside $\mathcal{H}_o$ (the shaded region in Figure 2(b)), consider the two half-lines starting from the center m with directions the same as the two asymptotes. Let $\mathcal{H}^a_o$ denote the region outside the two half-lines (the shaded polygon region in Figure 2(b)). Obviously $\mathcal{H}^a_o$ is in the dominant region and we use it to approximate the region outside $\mathcal{H}_o$.

Using Polar Coordinates to Denote Hyperbola: We propose to use the polar coordinate to represent a hyperbola. Consider two objects o and o^*. Suppose the corresponding dominant region $D_{o^*,o}$ is a hyperbola, denoted by $\mathcal{H}$. Let $\mathcal{H}_{o^*}(\mathcal{H}_o)$ denote the branch nearby o^* (o). Let l^{+d} and l^{-d} denote the two directrices. We construct a coordinate system where the origin is the midpoint between o and o^*, and the x-axis is the line passing o and o^*, the y-axis is the perpendicular bisector of segment between o^* and o. The distance from $l^{+d}(l^{-d})$ to y-axis is $\frac{\mathsf{a}}{\mathsf{e}}(-\frac{\mathsf{a}}{\mathsf{e}})$.

Consider any point p in branch $\mathcal{H}_{o^*}$. Let $\mathsf{d} = \texttt{dist}(p, l^{+d})$ denote the distance from p to directrix l^{+d} and $\mathsf{r} = \texttt{dist}(p, o)$. We have $\frac{\mathsf{r}}{\mathsf{d}} = \frac{\texttt{dist}(p,o)}{\texttt{dist}(p,l^{+d})} = \mathsf{e}$, where $\mathsf{e} = \frac{\mathsf{c}}{\mathsf{a}}$ is the eccentricity as discussed in Section 3.1. Let θ denote the angle between $\overrightarrow{o^*p}$ and x-axis. We have $p.x = \mathsf{r} * \cos\theta + \mathsf{c}$. As $\mathsf{d} = p.x - \frac{\mathsf{a}}{\mathsf{e}}$, we have $\mathsf{d} = \mathsf{r} * \cos\theta + \mathsf{c} - \frac{\mathsf{a}}{\mathsf{e}}$. Thus $\frac{\mathsf{r}}{\mathsf{r}*\cos\theta+\mathsf{c}-\frac{\mathsf{a}}{\mathsf{e}}} = \mathsf{e}$. We can deduce that

$$\mathsf{r} = f(\theta) = \frac{\mathsf{a}(\mathsf{e}^2 - 1)}{1 - \cos\theta * \mathsf{e}}.$$

where $-\arccos(\frac{1}{\mathsf{e}}) \leq \theta \leq \arccos(\frac{1}{\mathsf{e}})$.

In this way, we can use this polar coordinate representation to denote $\mathcal{H}_{o^*}$. Similarly for the other branch $\mathcal{H}_o$,

$$\mathsf{r}' = f'(\theta) = \frac{\mathsf{a}(1 - \mathsf{e}^2)}{1 + \cos\theta * \mathsf{e}},$$

where $\pi - \arccos(\frac{1}{\mathsf{e}}) \leq \theta \leq \pi + \arccos(\frac{1}{\mathsf{e}})$.

Similarly we can also use the polar coordinate to represent the plane, half-planes, and half-lines. The key to use polar coordinate to represent plane, half-planes or half-lines is to use it to represent lines. Considering line s, we draw its perpendicular $s^\perp$ from o^*. s and $s^\perp$ intersect at f. For any point p on s, the distance from o^* to p can be represented as $\mathsf{r} = \frac{\texttt{dist}(o^*,f)}{\cos\theta}$, where θ is the angle between lines o^*p and o^*f. Note that we need to transfer the coordinate. In the polar coordinate, the new origin is o^*. For all other objects, the origin is always o^*. We need to prove that o^* is in $\mathcal{A}$.

LEMMA 5. *Given a query q, if o^* is the best answer, the location of o^* must be in the safe region $\mathcal{R}$.*

To compute the intersection among multiple polar equations, we use a piecewise function in a polar coordinate with the origin of o^*. The safe region can be represented by

$$f^1(\theta)[\theta_1 \leq \theta < \theta_2], f^2(\theta)[\theta_2 \leq \theta < \theta_3], \cdots, f^m(\theta)[\theta_m \leq \theta < \theta_1].$$

4. COMPUTATION OF Safe Region

In this section we study how to calculate the safe region.

4.1 Framework

A naive method to compute the safe region is to first calculate the dominant regions of o^* to all other objects and then compute their intersection. To improve the performance, we propose an effective pruning technique. The basic idea is as follows. Consider two objects o_i and o_j. If $D_{o^*,o_i} \subseteq D_{o^*,o_j}$, we can prune D_{o^*,o_j} since $\mathcal{R} \subseteq D_{o^*,o_i} \cap D_{o^*,o_j} = D_{o^*,o_i}$. Notice that it is usually hard to determine whether $D_{o^*,o_i} \subseteq D_{o^*,o_j}$. To utilize this idea, we propose an alternative method. Consider a region $\mathcal{R}' \supseteq \mathcal{R}$. For any object o, if $\mathcal{R}' \subseteq D_{o^*,o}$, we can prune object o since $\mathcal{R} \subseteq \mathcal{R}' \subseteq D_{o^*,o}$ and object o will not affect the safe region.

To achieve our goal, there are two challenges. The first one is how to find such $\mathcal{R}'$. The second one is how to check whether $\mathcal{R}' \subseteq D_{o^*,o}$. To address the first challenge, we propose an incremental computation method. First, we initialize the safe region $\mathcal{R}'$ as the minimum bounding rectangle of all points. Next for each object o, we compute $D_{o^*,o}$. If $\mathcal{R}' \subseteq D_{o^*,o}$, we prune object o; otherwise we update $\mathcal{R}' = \mathcal{R}' \cap D_{o^*,o}$.

Next we discuss how to check $\mathcal{R}' \subseteq D_{o^*,o}$. We introduce two functions. As $o^* \in \mathcal{R}'$, let $d_{max} = \texttt{MaxBD}(o^*, \mathcal{R}')$ denote the maximum distance from o^* to *any point* in $\mathcal{R}'$. Given

an object o, let $d_{min}(o) = \texttt{MinBD}(o^*, D_{o^*,o})$ denote the minimum distance from o^* to the *boundary* of $D_{o^*,o}$. Notice that if $d_{min}(o) \geq d_{max}$, we have $\mathcal{R}' \subseteq D_{o^*,o}$ as formalized in Lemma 6, and thus we can prune object o.

LEMMA 6. *Given a region $\mathcal{R}' \supseteq \mathcal{R}$ and object o, if $d_{min}(o) \geq d_{max}$, $\mathcal{R}' \subseteq D_{o^*,o}$.*

Given an object o, next we discuss how to compute the function $d_{min}(o) = \texttt{MinBD}(o^*, D_{o^*,o})$.

Case 1: $\Delta t \geq \texttt{dist}(o^*, o)$. $D_{o^*,o}$ is the whole plane, and $\texttt{MinBD}(o^*, D_{o^*,o})$ is the minimal distance from o^* to the boundary of the plane;

Case 2: $0 < \Delta t < \texttt{dist}(o^*, o)$. $d_{min}(o)$ is the distance from o^* to the left vertex and $\texttt{MinBD}(o^*, D_{o^*,o}) = \texttt{c} + \texttt{a} = \frac{\texttt{dist}(o^*,o)+|\Delta t|}{2} = \frac{\texttt{dist}(o^*,o)+\Delta t}{2}$;

Case 3: $\Delta t = 0$. $D_{o^*,o}$ is a half-plane, and $\texttt{MinBD}(o^*, D_{o^*,o})$ is $\frac{\texttt{dist}(o^*,o)}{2}$;

Case 4: $-\texttt{dist}(o^*, o) < \Delta t < 0$. $d_{min}(o)$ is the distance from o^* to the right vertex and $\texttt{MinBD}(o^*, D_{o^*,o}) = \texttt{c} - \texttt{a} = \frac{\texttt{dist}(o^*,o)-|\Delta t|}{2} = \frac{\texttt{dist}(o^*,o)+\Delta t}{2}$;

Case 5: $\Delta t = -\texttt{dist}(o^*, o)$. $\texttt{MinBD}(o^*, D_{o^*,o}) = 0$.

Case 6: $\Delta t < -\texttt{dist}(o^*, o)$. There will be no such case as formalized in Lemma 4.

As shown in Figure 1, $\texttt{MinBD}(o_4, D_{o_4,o_3}) = \frac{\texttt{dist}(o_4,o_3)+\Delta t}{2} = 0.153$, $\texttt{MinBD}(o_4, D_{o_4,o_{15}}) = \frac{\texttt{dist}(o_4,o_{15})}{2} = 0.164$.

Here we discuss how to compute $\texttt{MaxBD}\ (o^*, \mathcal{R}')$. Let $\mathcal{I} = \{I_1, I_2, \cdots, I_m\}$ denote the intersections in $\mathcal{R}'$ among dominant regions $D_{o^*,o}$ for each object o that has been added into $\mathcal{R}'$ (also including the intersections with the boundary of the whole plane and the four vertexes of the plane if they are in the region). We have $\texttt{MaxBD}(o^*, \mathcal{R}') = \max_{I_t \in \mathcal{I}} \texttt{dist}(o^*, I_t)$ as stated in Lemma 7. We can easily compute the intersections using the polar coordinate.

LEMMA 7. *Given a region $\mathcal{R}'$ computed in the incremental algorithm, we have*

$$d_{max} = \textit{MaxBD}(o^*, \mathcal{R}') = \max_{I_t \in \mathcal{I}} \textit{dist}(o^*, I_t).$$

To facilitate the pruning, we access objects in ascending order sorted by d_{min}(o). If $d_{min}(o) \geq d_{max}$, we can prune all objects after object o. Based on this idea, we introduce our framework to compute the safe region and the pseudo-code is shown in Figure 1. We first initialize $\mathcal{R}$ as the whole plane and d_{max} (line 3-4). Then we sort the objects by $d_{min}(o)$ and create a priority queue (line 5). We dequeue and get the object o with the minimal value. If $d_{min}(o) \geq d_{max}$, the algorithm terminates (line 8); otherwise we update $\mathcal{R}$ and d_{max} (line 10-11).

For example, in Figure 1, to get the safe region of the query, we initialize $\mathcal{R}'$ as the whole plane and d_{max} as the distance from o_4 to the top-right corner of $\mathcal{R}$, i.e., d_{max} = 0.84. Then we calculate $\texttt{MinBD}\ (o, D_{o_4,o})$ for every $o \in O - \{o_4\}$ and insert o into the priority queue $\mathcal{Q}$ in order. The first element in $\mathcal{Q}$ is o_7 with $d_{min}(o_7) = 0.12 < d_{max}$. We dequeue it and update $\mathcal{R}' = \mathcal{R}' \cap D_{o_4,o_7}$. d_{max} is still 0.84 as the top-right corner is still the farthest point to o_4. Next we process objects $o_3, o_{15}, o_{14}, o_9, o_{13}$ in the same way and get $d_{max} = 0.27$. For the next object o_{10}, since $d_{min}(o_{10}) = 0.28 > d_{max}$, the algorithm terminates. Thus we only compute the dominant regions for 6 objects and prune the other 9 objects.

Algorithm 1: SENSE-NoIndex(O, q)

Input: O: A collection of objects
q: A query
Output: Safe Region $\mathcal{R}$

```
1  begin
2      o* = the best answer for query q;
3      R = the whole plane bounding objects o ∈ O ;
4      d_max = MaxBD(o*, R) ;
5      Sort object o in O based on MinBD(o*, D_{o*,o}), and
       build a priority queue Q with the sorted objects;
6      while Q is not empty do
7          o = Q.dequeue () ;
8          if d_min(o) ≥ d_max then break;
9          else
10             R = R ∩ D_{o*,o} ;
11             d_max = MaxBD(o*, R) ;
12 end
```

Figure 3: Sense (No-Index) Algorithm (Safe-region construction for moving spatial keyword queries)

4.2 Using Indexes to Improve Our Method

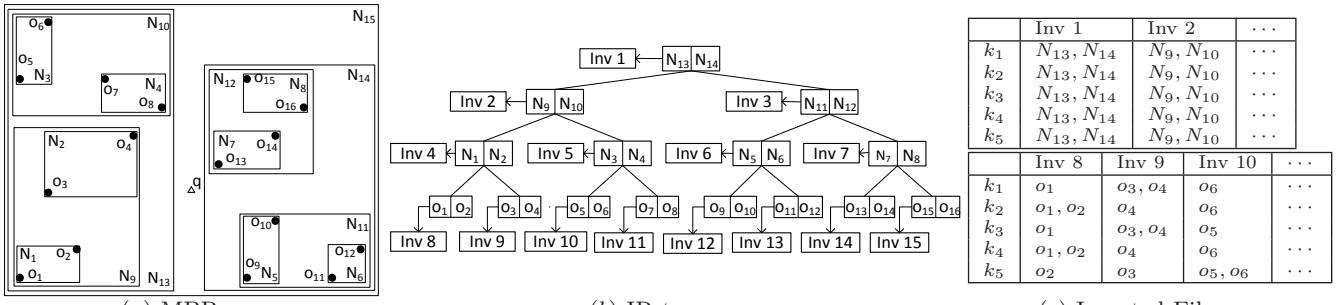
It is usually time-consuming to compute d_{min}(o) for all objects since there may be large numbers of objects. To address this issue, we utilize spatial structures to alleviate the problem. Without loss of generality, we use the IR-tree [2] as an example, which is the state-of-the-art index to answer top-k spatial keyword queries. Our method can be easily extended to support other indexing structures.

IR-tree incorporates inverted indexes into R-tree nodes. For each leaf node, in addition to keeping a set of objects in the minimum bounding rectangle (MBR) of this node, for each keyword contained in these objects, it also maintains an inverted list of objects in the node that contain the keyword. For each internal node, besides keeping a set of objects under this node, for each keyword, it also maintains an inverted list which keeps a list of its children which contain the keyword. Figure 4 shows the IR-tree structure.

The basic idea to use the spatial index to do pruning is as follows. Each IR-tree node n contains a group of objects under this node. We can estimate the lower bound of the dominant regions of objects under this node. If the estimated dominant region covers $\mathcal{R}'$, we can prune the whole subtree. As the IR-tree uses a hierarchical structure, we can prune many unnecessary nodes. Next we discuss the details.

Given a node n, we define the minimal bound distance for node n and object o^*, denoted by $\texttt{MinBD}(o^*, n)$. If $\texttt{MinBD}(o^*, n) \leq \min\{\texttt{MinBD}(o^*, D_{o^*,o}) | o \text{ is an object under node } n\}$, we have *if $\textit{MaxBD}(o^*, \mathcal{R}) \leq \textit{MinBD}(o^*, n)$, we can prune node n.* This is because for each object o under node n, we have $\texttt{MaxBD}(o^*, \mathcal{R}) \leq \texttt{MinBD}(o^*, D_{o^*,o})$ as $\mathcal{R}$ is inside $D_{o^*,o}$. Then we discuss how to define the function $\texttt{MinBD}(o^*, n)$.

Let $\textsf{mindist}(o^*, n)$ and $\textsf{maxdist}(o^*, n)$ respectively denote the minimal and maximal distance from node o^* to the MBR of node n. Let n_t denote the set of terms under this node. Notice that for each term t in n_t, its term frequency is the largest term frequency of objects under this node and its inverse document frequency is still $\textsf{idf}(t)$. Let $\Delta t_{min} = \frac{1-\alpha}{\alpha}(\texttt{text}(q', o^*) - \texttt{text}(q', n_t))$. For each object o under node n, we have



(a) MBRs (b) IR-tree

	Inv 1	Inv 2	···
k_1	N_{13}, N_{14}	N_9, N_{10}	···
k_2	N_{13}, N_{14}	N_9, N_{10}	···
k_3	N_{13}, N_{14}	N_9, N_{10}	···
k_4	N_{13}, N_{14}	N_9, N_{10}	···
k_5	N_{13}, N_{14}	N_9, N_{10}	···

	Inv 8	Inv 9	Inv 10	···
k_1	o_1	o_3, o_4	o_6	···
k_2	o_1, o_2	o_4	o_6	···
k_3	o_1	o_3, o_4	o_5	···
k_4	o_1, o_2	o_4	o_6	···
k_5	o_2	o_3	o_5, o_6	···

(c) Inverted Files

Figure 4: IR-tree

(1) $\mathsf{mindist}(o^*, n) \leq \mathtt{dist}(o^*, o) \leq \mathsf{maxdist}(o^*, n)$; and

(2) $\Delta t \geq \Delta t_{min}$ where $\Delta t = \frac{1-\alpha}{\alpha}(\mathtt{text}(q', o^*) - \mathtt{text}(q', o))$.

Based on these two notations, we can define $\mathtt{MinBD}(o^*, n)$ as follows.

Case 1: $\Delta t_{min} \geq \mathsf{maxdist}(o^*, n)$. $\mathtt{MinBD}(o^*, n)$ is the minimal distance from o^* to the boundary of the whole plane, denoted by D_{min}.

Case 2: $0 < \Delta t_{min} < \mathsf{maxdist}(o^*, n)$.

$$\mathtt{MinBD}(o^*, n) = \min(\frac{\mathsf{mindist}(o^*, n) + \Delta t_{min}}{2}, D_{min}).$$

Case 3: $\Delta t_{min} = 0$. $\mathtt{MinBD}(o^*, n) = \min(\frac{\mathsf{mindist}(o^*, n)}{2}, D_{min})$.

Case 4: $-\mathsf{mindist}(o^*, n) < \Delta t_{min} < 0$.

$$\mathtt{MinBD}(o^*, n) = \min(\max(0, \frac{\mathsf{mindist}(o^*, n) + \Delta t_{min}}{2}), D_{min}).$$

Case 5: $\Delta t_{min} < -\mathsf{mindist}(o^*, n)$. $\mathtt{MinBD}(o^*, n) = 0$.

LEMMA 8. *Given an object o^* and a node n, the above-defined function* ***MinBD*** *(o^*, n) satisfies* ***MinBD****$(o^*, n) \leq \min\{$****MinBD****$(o^*, D_{o^*,o}) | o$ is an object under node $n\}$.*

We can prove that $\mathtt{MinBD}(o^*, n) \leq \min\{\mathtt{MinBD}(o^*, D_{o^*,o}) | o$ is an object under node $n\}$. Based on this function, we can use indexes to prune nodes (groups of objects) and propose an index-based method. The pseudo-code is shown in Figure 5. We first initialize $\mathcal{R}$ and d_{max} similar to the non-index algorithm (line 3-4) and create an empty priority queue $\mathcal{Q}$ (line 5). Then we insert the root of the IR-tree in to $\mathcal{Q}$ (Line 6). We dequeue the top element n in $\mathcal{Q}$ until $\mathtt{MinBD}(o^*, n) \geq d_{max}$ (line 8). If $\mathtt{MinBD}(o^*, n) \geq d_{max}$, the algorithm terminates (line 9) since all dominant regions of remaining objects contain the safe region. If n is a node, we get each of its children n', calculate its dominant region $D_{o^*,n}$ and $d_{min}(n')$, and insert n' into $\mathcal{Q}$ with its corresponding $d_{min}(n')$ (line 10-13). If n is an object, we update $\mathcal{R}$ using the intersection between $\mathcal{R}$ and its dominant region (line 15) and also update d_{max} (line 16).

4.3 Cache-based Improvement

Note that before finding the safe region $\mathcal{R}$, we must compute the answer set $\mathcal{A}$. When computing $\mathcal{A}$, we have already traversed the IR-tree and visited some nodes and objects. To compute the safe region, we may still need to visit some of these nodes and objects. For example, in Figure 4, nodes N_{13}, N_{14}, N_{15} will be visited twice. More importantly, to compute the answer set and the safe region, we use the same keyword set. Based on these observations, we can cache some information to avoid unnecessary computation

Algorithm 2: SENSE-Index(O, q)

Input: O: A collection of objects
q: A query
Output: Safe Region $\mathcal{R}$

```
1  begin
2      o* = the best answer for query q;
3      R = the whole plane bounding all o ∈ O ;
4      d_max = MaxBD(o*, R) ;
5      Q = an empty priority queue ;
6      Build an IR-tree and insert root into Q ;
7      while Q is not empty do
8          n = Q.dequeue () ;
9          if d_min(n) ≥ d_max then break;
10         if n is a node then
11             for each child n' of n do
12                 d_min(n') = MinBD(o*, n') ;
13                 Insert n into Q with d_min(n') ;
14         else
15             R = R ∩ D_{o*,n} ;
16             d_max = MaxBD(o*, R) ;
17 end
```

Figure 5: Sense (Index) Algorithm

and facilitate the safe region computation. To this end, we cache the following information.

(1) Inverted lists: When visiting a node, the answer computation step needs to load the inverted lists of query keywords. In the safe region computation step, we may still use such inverted lists, and thus we can cache them.

(2) Keyword MBR: In order to get $\mathtt{MinBD}(o^*, n)$ for a node n, we have to use $\mathsf{mindist}(o^*, n)$ (and $\mathsf{maxdist}(o^*, n)$). The most direct way is to use the object which is nearest (and farthest) to o^*. However this method is inaccurate since some objects in the MBR may contain no query keyword and this kind of objects will not affect the safe region. To improve the accuracy, for a visited node in the answer computation step, we compute and store the MBR that contain at least one keyword, called *keyword MBR*. We use the keyword MBR to estimate the spatial information of this node.

(3) Score bounds: For a visited node in the answer computation step, for each node we can cache its virtual textual information n_t and Δt.

Thus during the answer computation step, we cache the above information. Then in the safe region computation step, we can utilize such information for pruning and thus can improve the efficiency. The cache-based method has the following two advantages. First, as IR-tree is a disk-based structure, it is expensive to access nodes multiple times from the disk and the cache-based method can reduce the number of disk accesses. Second, we can estimate the dominant region for each object more accurately.

4.4 Discussions

Checking Safe Region in the Client: In the client, we need to check whether the current location is in a safe region. As we use a piecewise function to represent a safe region, it is very efficient to do the checking as follows. Assume the current location is q'_s, $\mathcal{A} = \{o^*\}$ and $\mathcal{R} = f^i(\theta)[\theta_i \le \theta < \theta_{i+1}](1 \le i \le m, \theta_{m+1} = \theta_1)$. First, we compute θ' for q'_s which is the angle from line o^*q' to the x-axis. Suppose $\theta' \in [\theta_j, \theta_{j+1}), 1 \le j \le m$. We calculate $r = f^j(\theta')$ which is the distance from o^* to the boundary of $\mathcal{R}$ with this angle. Therefore, if $\texttt{dist}(o^*, q'_s) \le r$, q'_s is inside $\mathcal{A}$, and thus o^* is still the best answer. Otherwise, the client should send a new query to the server.

Supporting other functions: Our framework can be extended to support the ranking function in Equation 3 which was used in [17]. Different from our ranking function, the shape of the safe region of this function is composed of circles and lines. We can extend our techniques to compute the dominant region and the safe region. Notice that in their work, they only use polygons to approximate circles and they cannot find the exact safe region. However we can use the polar coordinate to efficiently compute the exact safe region. Due to space constraints, we omit the details.

5. EXTENDING OUR TECHNIQUES TO SUPPORT TOP-K QUERIES

In this section, we extend our techniques to support top-k queries. We propose an intersection-based method in Section 5.1 and an approximate method in Section 5.2.

5.1 Intersection-based Method

To compute the safe region for a top-k query, an intuitive way is to first compute the safe region for each object $o^* \in \mathcal{A}$ to the object set $O - \mathcal{A}$ (denoted by $\mathcal{R}_{o^*,O-\mathcal{A}}$) and then compute their intersection ($\mathcal{R} = \cap_{o^* \in \mathcal{A}} \mathcal{R}_{o^*,O-\mathcal{A}}$). We can compute $\mathcal{R}_{o^*,O-\mathcal{A}}$ by intersecting the dominant region of o^* to every object $o \in O - \mathcal{A}$. Thus we have $\mathcal{R} = \cap_{o^* \in \mathcal{A}}(\cap_{o \in O-\mathcal{A}} D_{o^*,o})$ as stated in Lemma 9.

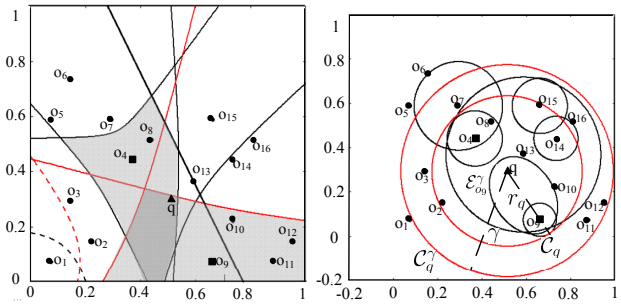
LEMMA 9. *Given a query q and its answer set $\mathcal{A}$, the safe region can be computed as:*

$$\mathcal{R} = \cap_{o^* \in \mathcal{A}}(\cap_{o \in O-\mathcal{A}} D_{o^*,o}). \quad (6)$$

For example, in Figure 6(a), assume $k = 2$. The shaded regions respectively denote $\mathcal{R}_{o_4,O-\{o_9\}}$, $\mathcal{R}_{o_9,O-\{o_4\}}$, and the safe region for $\{o_4, o_9\}$. Obviously the safe region of $\{o_4, o_9\}$ is the intersection of $\mathcal{R}_{o_4,O-\{o_9\}}$ and $\mathcal{R}_{o_9,O-\{o_4\}}$.

Based on Lemma 9, for each object o^* in $\mathcal{A}$, we first compute $\mathcal{R}_{o^*,O-\mathcal{A}}$ and then intersect $\mathcal{R}_{o^*,O-\mathcal{A}}$ for every o^* in $\mathcal{A}$. For the no-index based algorithm (Algorithm 1), we can easily compute $\mathcal{R}_{o^*,O-\mathcal{A}}$ by replacing O with $O-\mathcal{A}$. For the index-based algorithm (Algorithm 2), during the traversal of the indexing structure, if we encounter an object in $\mathcal{A}$, we just ignore the object. If we encounter a node, we use the same method, regardless of whether the node contains an object in $\mathcal{A}$. The main reason is as follows. If we prune a node n, all objects under the node cover the safe region. Thus we can use the two algorithms to support top-k queries.

For example in Figure 6(a), consider $\mathcal{A} = \{o_4, o_9\}$. We first compute the safe region for o_4, i.e., $\mathcal{R}_{o_4,O-\{o_4,o_9\}}$. Different from finding the safe region for top-1 answer, we do not need to compute D_{o_4,o_9}. Then, based on the safe region for o_4, we traverse the IR-tree again to compute $\mathcal{R}_{o_9,O-\{o_4,o_9\}}$. The intersection between $\mathcal{R}_{o_4,O-\{o_4,o_9\}}$ and $\mathcal{R}_{o_9,O-\{o_4,o_9\}}$ is the safe region.



(a) Intersection-based (b) Approximate method

Figure 6: Supporting top-k queries

Next we discuss how to compute the intersection of regions $\mathcal{R}_{o^*,O-\mathcal{A}}$ for $o^* \in \mathcal{A}$. As each region is composed by a set of hyperbolas, it is expensive to compute the intersection. Although we use the polar coordination, different regions have different origins and we have to transfer coordinates. To alleviate this problem, we use a polygon to approximate the region as follows. For each curve of the region, if it is convex, we use the segment between its two end points to approximate it; if the curve is concave, we use tangents inside the safe region to approximate it. Notice that if we use the index-based algorithm, this method needs to access the index k times. If k is large, the performance will be poor. To address this issue, we propose an alternative method. The basic idea is to compute the intersection during the traversal of the index, we only access the index once. Before introducing our idea, we first define the dominant region of an answer set $\mathcal{A}$ to an object $o \notin \mathcal{A}$.

DEFINITION 5. *Given a query $q = (q_s, q_t, k)$, the dominant region of $\mathcal{A}$ to o ($o \notin \mathcal{A}$) is:*

$$D_{\mathcal{A},o} = \{q'_s | \forall o^* \in \mathcal{A}, \texttt{score}(q', o^*) \le \texttt{score}(q', o)\}, \quad (7)$$

where $q' = (q'_s, q_t, k)$.

Obviously $\mathcal{R} = \cap_{o \notin \mathcal{A}} D_{\mathcal{A},o}$. Based on this concept, we propose two incremental algorithms. For the no-index method, we only need to use $D_{\mathcal{A},o}$ to replace $D_{o^*,o}$. For the index-based method, besides replacing $D_{o^*,o}$ with $D_{\mathcal{A},o}$ for an object o, we need to redefine how to prune an object o (node n). As o^* may not be in $\mathcal{R}$, we need to reselect a location which must be in $\mathcal{R}$. Based on Lemma 3, q must be in $\mathcal{R}$. Thus we use $\texttt{MinBD}(q, D_{\mathcal{A},o})$ and $\texttt{MaxBD}(q, \mathcal{R})$ to do pruning. We can easily deduce a lower bound of the distance from q to the boundary of $D_{\mathcal{A},o}$ based on the triangle inequality, i.e., we set $\texttt{MinBD}(q, D_{\mathcal{A},o}) = \texttt{MinBD}(o^*, D_{\mathcal{A},o}) - \texttt{dist}(o^*, q)$. $\texttt{MaxBD}(q, \mathcal{R})$ is the maximal distance from q to the boundary of $\mathcal{R}$. Then, we can use our framework to compute the safe region. As $D_{\mathcal{A},o}$ and $\mathcal{R}$ are complex regions composed by a set of hyperbolas, it is expensive to compute the region. To address this issue, we use polygons to approximate the regions as described in Section 3.

For example, in Figure 6(a), the dashed lines represent the boundary of D_{o_4,o_1} and D_{o_9,o_1} respectively and their intersection is $D_{\mathcal{A},o_1}$. In the intersection method, we compute the safe region first for o_4 and then for o_9. Instead, our incremental method first computes $D_{\mathcal{A},o}$ for $o \notin \mathcal{A}$, and then calculates their intersection to get the safe region.

5.2 Approximate Method

The previous methods compute the *global* safe region, and in this section we compute a *local* safe region (denoted by $\tilde{\mathcal{R}}$) which is a subset of the exact *global* safe region ($\mathcal{R}$). As the client's locations are usually not far away from the current query, we can use the local safe region to approximate the safe region. This method has a big advantage that it is very efficient to compute the local safe region and outperforms the methods in Section 5.1. Next we discuss the details.

We can transfer our ranking function in Equation 1 into the following form:

$$\texttt{score}(q,o) = \alpha \times \big(\texttt{dist}(q,o) + \frac{1-\alpha}{\alpha} \cdot (1 - \texttt{text}(q,o)\big).$$

In this way we can model each object o as a circle $\mathcal{C}_o$ with center o and radius of $r_o = \frac{1-\alpha}{\alpha}(1 - \texttt{text}(q,o))$. Then,

$$\texttt{score}(q,o) = \alpha\big(\texttt{dist}(q,o) + r_o\big).$$

Let $r_q = \max\{\texttt{dist}(q,o^*) + r_{o^*} | o^* \in \mathcal{A}\}$ and $\mathcal{C}_q$ denote the circle centered at q and radius of r_q. For an object $o^* \in \mathcal{A}$, we have $\texttt{score}(q,o^*) \leq \alpha r_q$, which is equivalent to $\texttt{dist}(q,o^*) + r_{o^*} \leq r_q$, thus $o^* \in \mathcal{A}$ if and only if the circle $\mathcal{C}_{o^*}$ are inside $\mathcal{C}_q$. Similarly an object $o \in O - \mathcal{A}$ if and only if the circle $\mathcal{C}_o$ are not inside $\mathcal{C}_q$.

Then suppose the client moves to location q'. Let $\mathcal{C}_{q'}$ denote the circle centered at q' and radius of $r_q - \texttt{dist}(q,q')$. As $\mathcal{C}_{q'}$ is inside $\mathcal{C}_q$, for each object $o^* \in \mathcal{A}$, if $\mathcal{C}_{o^*}$ is inside $\mathcal{C}_{q'}$, o^* is still a top-k answer of q'. $\mathcal{C}_{o^*}$ is inside $\mathcal{C}_{q'}$ if and only if

$$\texttt{dist}(o^*,q') + r_{o^*} \leq r_q - \texttt{dist}(q,q'),$$

which can be written as

$$\texttt{dist}(o^*,q') + \texttt{dist}(q,q') \leq r_q - r_{o^*}. \quad (8)$$

Obviously the locus of such points is an ellipse with q and o^* as the two focuses, denoted by $\mathcal{E}_{o^*} = \{q'_s | \texttt{dist}(o^*,q') + \texttt{dist}(q,q') \leq r_q - r_{o^*}\}$. Obviously $\mathcal{E} = \cap_{o^* \in \mathcal{A}} \mathcal{E}_{o^*}$ is a local safe region, that is if $q' \in \mathcal{E}$, $\mathcal{A}$ is still the answer set. Since $q \in \mathcal{E}$ as stated in Lemma 10, $\mathcal{E}$ is not empty.

LEMMA 10. *Given a query q and its answer set $\mathcal{A}$, $\mathcal{E}$ is not empty and $q \in \mathcal{E}$.*

We can use the polar coordinate to represent an ellipse. As q is in each ellipse, we can take q as the origin of the polar coordinate and thus can efficiently compute the intersection of the k ellipses. Due to space constraints, we omit the details about how to use polar coordinates to compute the intersection. As the local safe region is small, we can enlarge it as follows. Based on Equation 8, if r_q increases, the eclipse $\mathcal{E}_{o^*}$ will become larger and so will region $\mathcal{E}$. Thus we can increase r_q to enlarge the safe region as follows (Figure 6(b)).

Let $\gamma = r_q + \triangle$ and $\mathcal{C}_q^\gamma$ denote the circle centered at q and radius of γ. We compute the ellipse $\mathcal{E}^\gamma_{o^*} = \{q'_s | \texttt{dist}(o^*,q') + \texttt{dist}(q,q') \leq \gamma - r_{o^*}\}$, and region $\mathcal{E}^\gamma = \cap_{o^* \in \mathcal{A}} \mathcal{E}^\gamma_{o^*}$. Obviously $\mathcal{E} \subseteq \mathcal{E}^\gamma$. Obviously $\mathcal{A}$ dominates the objects outside $\mathcal{C}_q^\gamma$. Since there may be other objects involved in $\mathcal{C}_q^\gamma - \mathcal{C}_q$, we need to determine whether $\mathcal{A}$ dominates these objects. Let $\mathcal{C}_q^\gamma - \mathcal{C}_q$ denote the set of objects whose corresponding circles inside circle $\mathcal{C}_q^\gamma$ but not inside $\mathcal{C}_q$ and $D_{\mathcal{A},\mathcal{C}_q^\gamma - \mathcal{C}_q}$ denote the dominant region of $\mathcal{A}$ to $\mathcal{C}_q^\gamma - \mathcal{C}_q$ which is $\cap_{o^* \in \mathcal{A}} D_{o^*,\mathcal{C}_q^\gamma - \mathcal{C}_q}$. Let $\tilde{\mathcal{R}}^\gamma = \mathcal{E}^\gamma \cap D_{\mathcal{A},\mathcal{C}_q^\gamma - \mathcal{C}_q}$. We can take $\tilde{\mathcal{R}}^\gamma$ as a local safe region since $\tilde{\mathcal{R}} \subseteq \mathcal{R}$ as formalized in Lemma 11.

LEMMA 11. *For any γ, we have*
(1) $\tilde{\mathcal{R}}^\gamma = \mathcal{E}^\gamma \cap D_{\mathcal{A},\mathcal{C}_q^\gamma - \mathcal{C}_q} \subseteq \mathcal{R}$, and
(2) If γ is ∞, $\tilde{\mathcal{R}}^\gamma = \mathcal{R}$.

There are two challenges. The first one is how to compute $D_{\mathcal{A},\mathcal{C}_q^\gamma - \mathcal{C}_q}$. We can find the objects in $\mathcal{C}_q^\gamma$-$\mathcal{C}_q$ using any spatial index structures (which is the objects with distance to q between γ and r_q). Then based on these objects, we can efficiently compute $D_{\mathcal{A},\mathcal{C}_q^\gamma - \mathcal{C}_q}$ based on the non-index based algorithm. The second one is how to select γ. If γ is small, the local safe region is small, and the client queries have large probabilities outside the region. If γ is large, there will be more objects in $\mathcal{C}_q^\gamma - \mathcal{C}_q$, and it will take more time to compute $D_{\mathcal{A},\mathcal{C}_q^\gamma - \mathcal{C}_q}$. Thus it is a tradeoff to select an appropriate γ. We can determine the value based on the client moving speed (e.g., m meter/per second). If we expect the query location to be still in the safe region after s seconds, we set $\gamma = r_q + k * m * s$.

6. EXPERIMENT

We implemented our proposed techniques and compared with the state-of-the-art method MSK [17]. We used disk-based IR-tree [2] as the index to compute the answers and safe regions. We fixed the page size at 4KB. We used two real spatial data sets composed of POIs in California and Beijing. The details are listed in Table 2. We randomly generated 100 query trajectories and each query had 2-5 keywords. Each trajectory consisted 1,000 points and the distance between two consecutive points were 100 meters. All the experiments were implemented in Java and conducted on a Linux server with Intel(R) Xeon(R) 2.27GHz CPU and 4GB RAM.

Table 2: Data Sets

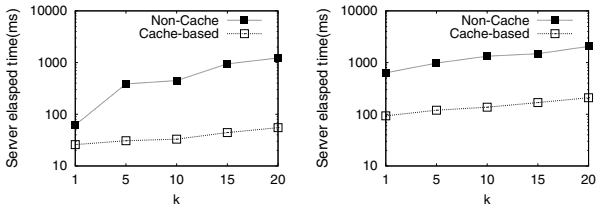
data set	# of objects	# of distinct keywords	avg # of keywords per object
California	544,906	132,552	7.39
Beijing	1,056,770	93,543	4.52

6.1 Evaluation of Cache-based Technique

In this section, we evaluate the cache-based technique. We implemented two algorithms, SENSE-Index without cache and SENSE-index with cache, where we used the incremental intersection methods. Figure 7 shows the results. We can see that the cache-based technique can significantly improve the performance. For example, on the California dataset, when $k = 20$, the cache-based time reduces the time from 1228 milliseconds to 55 milliseconds. This is because we can reduce large numbers of IOs by using cached information.

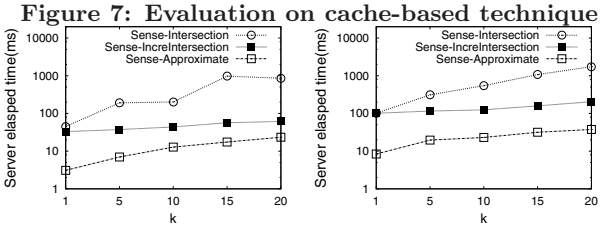
6.2 Evaluation of Computing Algorithms

We first evaluate our different methods to compute the safe region. We implemented tree algorithms, intersection based method (SENSE-INTERSECTION), incremental intersection-based method (SENSE-INCREINTERSECTION), and approximate-based method (SENSE-APPROXIMATE). Figure 8 shows the experimental results. We can see that SENSE-INCREINTERSECTION and SENSE-APPROXIMATE outperform SENSE-INTERSECTION. This is because SENSE-INTERSECTION has to traverse the IR-tree k times. If $k = 1$, SENSE-INCREINTERSECTION and SENSE-INTERSECTION achieved nearly the same performance. SENSE-APPROXIMATE was better than the other two methods as it reduces the number of dominant regions and estimates the safe regions using smaller numbers of dominant regions. As SENSE-INCREINTERSECTION is always better than SENSE-INTERSECTION, next we only compare SENSE-INCREINTERSECTION and SENSE-APPROXIMATE.



(a) California (b) Beijing

Figure 7: Evaluation on cache-based technique



(a) California (b) Beijing

Figure 8: Evaluation of computing model on server

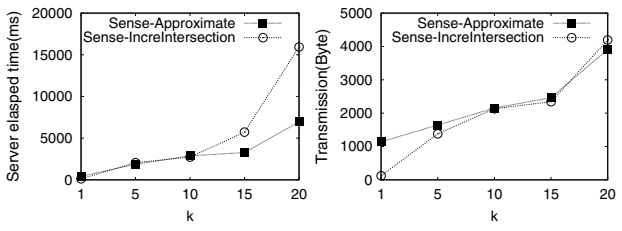
6.3 SENSE-INCREINTERSECTION vs. SENSE-APPROXIMATE

We further compare SENSE-INCREINTERSECTION and SENSE-APPROXIMATE in terms of average server elapsed time, average client elapsed time, average communication cost (bytes) and the update frequency which is the ratio of the number of queries issued to the server to the total number of queries. The client issued a query for each point in a trajectory. For SENSE-APPROXIMATE, to calculate $\mathcal{A}$ and $\mathcal{R}$, we set $s = 1000$. Figure 9 shows the results by varying k. Due to the space constraints, we only show the results on the California dataset. On the Beijing dataset, we got similar results. We can see that SENSE-APPROXIMATE took less server time as it uses an approximate method to compute the safe region. Although SENSE-APPROXIMATE involved more client time, it only took about 0.006 milliseconds which is negligible. SENSE-APPROXIMATE involved more communication overhead since it needs to intersect hyperbolas and ellipses which results in more vertexes. SENSE-APPROXIMATE had larger update frequency as it estimated the safe region which is much smaller than the real safe region.

To further compare the two methods, we evaluate the server time and communication cost in a time window (We do not compare the client time as it is negligible). In our experiments, the time windows is 10 minutes. We compared the total server elapsed time to answer the queries in the window. Figure 10 shows the results. We can see that SENSE-APPROXIMATE took less server time than SENSE-INCREINTERSECTION, especially for larger k values. This is because SENSE-APPROXIMATE estimates the safe regions and reduces the computation time. SENSE-APPROXIMATE involved nearly the same communication overhead with SENSE-INCREINTERSECTION, especially for a large k. This is because our approximate method has better approximation ratio for larger k, since it enlarges the space for larger k.

6.4 Comparison with state-of-the-art method

As the state-of-the-art method MSK [17] only supports the ad-hoc ranking function as discussed in Section 2.2, we extend SENSE-INCREINTERSECTION to support the same ranking function and compare with it. Figure 11 shows the results by varying k. We can see that our method significantly outperforms MSK in terms of server time, client time, and communication cost. The main reason is as follows. For server time, we use polar coordinates to represent the safe region, which is much more efficient than the approximate based method. For the communication cost, MSK needs to return some objects to avoid involving false negatives. That is also why the client time of MSK is much larger than SENSE: besides checking whether the current location is in safe region, the client has to examine every returned objects. SENSE and MSK nearly achieve the same update frequency as they generate similar safe region. The only difference is that MSK approximates the safe regions using polygons and our method computes the exact ones. Figure 12 shows the results by varying velocity. We can see that our method still significantly outperforms MSK in terms of the server time, client time, and communication cost.



(a) Server Time (b) Communication Cost

Figure 10: Sense-IncreIntersection vs. Sense-Approximate (for 10 minutes, California)

7. RELATED WORK

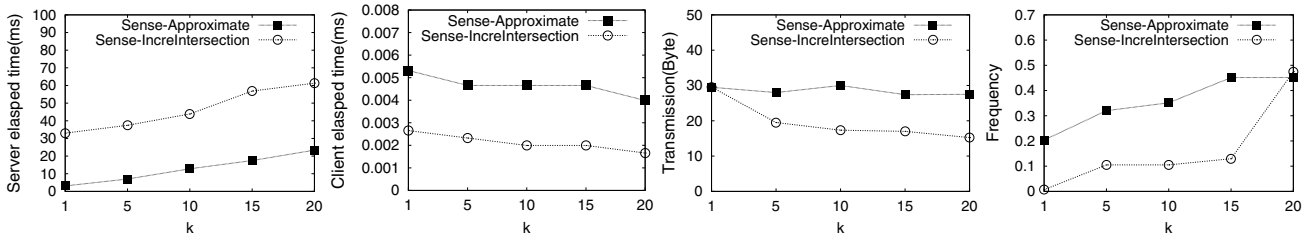
There have been many studies on spatial keyword search [9, 7, 22, 19, 18, 10, 13, 1, 3, 21]. Felipe et al. [4] addressed a k-nearest-neighbor problem which returns k objects that contain the query keywords and are near to the query location. They proposed IR^2-tree by adding signature files to R-tree nodes. Cong et al. [2] proposed the IR-tree which can support IR-based ranking functions. Zhou et al. [22] and Hariharan et al. [7] studied range-based spatial keyword search, which, given a rectangle and a set of keywords, finds all relevant answers that are located in the rectangle. Zhang et al. [19] studied collective keyword search, which finds a set of objects that contain all the query keywords. Lu et al. [10] studied reverse spatial and textual k nearest neighbor search. Li et al. [9] studied direction-aware search by considering directions. Obviously these problems are different from ours.

Wu et al. [17] studied moving spatial keyword search. However they only supported ad-hoc ranking functions (Section 2.2). Instead our method adopts a general ranking function. When we adapted our method to support their functions, our method significantly outperforms their approach in terms of both efficiency and communication cost.

Continuous queries have attracted much attention with the popularity of location-based services. For a continuous query, the query position was moving while the objects can be static [15, 14, 16] or moving [6, 8, 5]. We consider the former case. To avoid repeatedly issuing queries, the safe region based method was proposed [20, 12]. However these studies only considered spatial information and did not take into account textual descriptions.

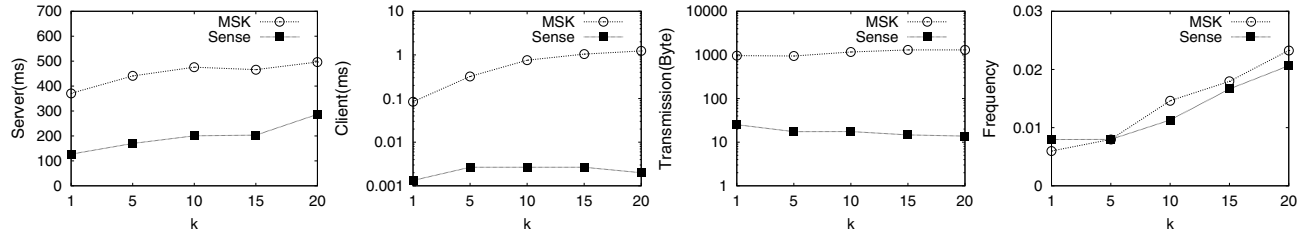
8. CONCLUSION

In this paper we have studied the moving top-k spatial keyword search problem. For each query submitted to the server, besides generating top-k answers we also constructed its safe region. We proposed to use hyperbolas to represent the safe region. To efficiently calculate the safe region, we devised effective pruning techniques and utilized index-



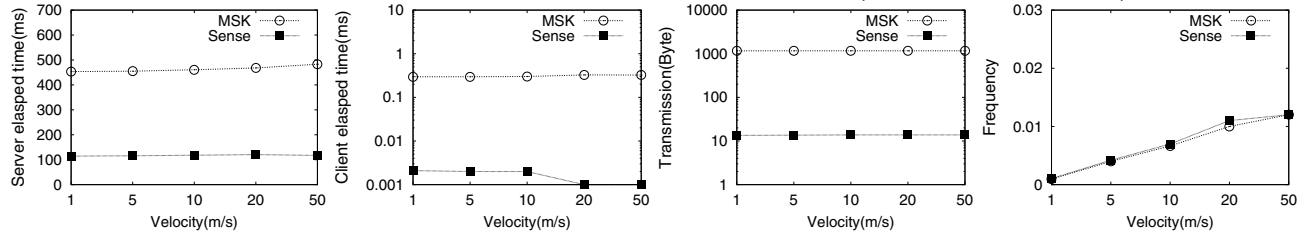
(a) Server Elapsed Time (b) Client Elapsed Time (c) Communication Cost (d) Update Frequency

Figure 9: Sense-IncreIntersection vs. Sense-Approximate (varying k, California)



(a) Server Elapsed Time (b) Client Elapsed Time (c) Communication Cost (d) Update Frequency

Figure 11: Comparison with state-of-the-art method (varying k, California)



(a) Server Elapsed Time (b) Client Elapsed Time (c) Communication Cost (d) Update Frequency

Figure 12: Comparison with state-of-the-art method (varying velocity, California)

ing structures to improve the performance. We also developed incremental algorithms to efficiently compute the safe region. We have implemented our proposed techniques and experimental results show that our method significantly outperforms state-of-the-art approaches.

Acknowledgement. This work was partly supported by the National Natural Science Foundation of China under Grant No. 61003004, National Grand Fundamental Research 973 Program of China under Grant No. 2011CB302206, and a project of Tsinghua University under Grant No. 20111081073, and the "NExT Research Center" funded by MDA, Singapore, under Grant No. WBS:R-252-300-001-490.

9. REFERENCES

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

[2] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.

[3] J. Fan, G. Li, L. Zhou, S. Chen, and J. Hu. Seal: Spatio-textual similarity search. *PVLDB*, 5(9):824–835, 2012.

[4] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[5] B. Gedik and L. Liu. Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In *EDBT*, pages 67–87, 2004.

[6] B. Gedik, K.-L. Wu, P. S. Yu, and L. Liu. Motion adaptive indexing for moving continual queries over moving objects. In *CIKM*, pages 427–436, 2004.

[7] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *SSDBM*, page 16, 2007.

[8] H. Hu, J. Xu, and D. L. Lee. A generic framework for monitoring continuous spatial queries over moving objects. In *SIGMOD Conference*, pages 479–490, 2005.

[9] G. Li, J. Xu, and J. Feng. Desks: Direction-aware spatial keyword search. In *ICDE*, pages 459–470, 2012.

[10] J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In *SIGMOD Conference*, pages 349–360, 2011.

[11] B. Martins, M. J. Silva, and L. A. Ribeiro. Indexing and ranking in geo-ir systems. In *GIR*, pages 31–34, 2005.

[12] S. Nutanong, R. Zhang, E. Tanin, and L. Kulik. The v*-diagram: a query-dependent approach to moving knn queries. *PVLDB*, 1(1):1095–1106, 2008.

[13] S. B. Roy and K. Chakrabarti. Location-aware type ahead search on spatial databases: semantics and efficiency. In *SIGMOD Conference*, pages 361–372, 2011.

[14] Z. Song and N. Roussopoulos. K-nearest neighbor search for moving query point. In *SSTD*, pages 79–96, 2001.

[15] Y. Tao and D. Papadias. Time-parameterized queries in spatio-temporal databases. In *SIGMOD Conference*, pages 334–345, 2002.

[16] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *VLDB*, pages 287–298, 2002.

[17] D. Wu, M. L. Yiu, C. S. Jensen, and G. Cong. Efficient continuously moving top-k spatial keyword query processing. In *ICDE*, pages 541–552, 2011.

[18] B. Yao, F. Li, M. Hadjieleftheriou, and K. Hou. Approximate string search in spatial databases. In *ICDE*, pages 545–556, 2010.

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

[20] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. Location-based spatial queries. In *SIGMOD Conference*, pages 443–454, 2003.

[21] R. Zhong, J. Fan, G. Li, K.L.. Tan, and L. Zhou. Location-Aware Instant Search. In *CIKM*, 2012.

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