Monitoring Reverse Top-k Queries Over Mobile Devices

Akrivi Vlachou*
Dept. of Informatics
AUEB
Athens, Greece
avlachou@aueb.gr

Christos Doulkeridis
Dept. of Informatics
AUEB
Athens, Greece
cdoulk@aueb.gr

Kjetil Nørvåg
Dept. of Computer Science
NTNU
Trondheim, Norway
Kjetil.Norvag@idi.ntnu.no

ABSTRACT

In typical mobile applications, mobile users seek points of interest in their vicinity (e.g., nearby restaurants) that best match their preferences. We assume a set of points of interest described by a combination of static and dynamic attributes, and a set of mobile users m_i, each associated with a weighting vector w_i, which expresses m_i's preferences over the aforementioned attribute set. The best points of interest for each mobile user correspond to the results of a top-k query, defined by the weighting vector w_i, which is performed over the combined set of static and dynamic attributes. The dynamic attribute is the current distance between the mobile user and the point of interest. Under these assumptions, the potential customers of a given point of interest q are the mobile users whose weighting vectors w_i belong to the reverse top-k set of q. In this paper, we define the *distance-based reverse top-k query* suitable for mobile environments. The problem we target is given a query point q, how to efficiently monitor q's distance-based reverse top-k result set. To address this problem, we introduce novel algorithms that enable efficient monitoring of distance-based reverse top-k result sets over mobile devices. Our experimental evaluation demonstrates the efficiency of our techniques for a wide variety of diverse setups.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Query processing*

General Terms

Algorithms, Experimentation, Performance

Keywords

Reverse top-k query, mobile systems

*Akrivi Vlachou was supported by the Greek State Scholarship Foundation (IKY).

MobiDE '11 June 12, 2011 Athens, Greece

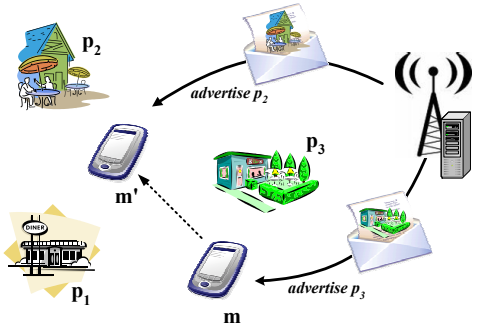


Figure 1: Motivating example.

1. INTRODUCTION

Location-based queries are widely employed to retrieve useful information based on the user's geographical position. For example, a tourist that walks around a city may seek points of interest (e.g., restaurants) in her vicinity that satisfy her preferences (e.g., cheap and highly-rated). A top-k query defined by the user preferences can be employed to return a ranked set of the k best points of interest that are nearby her current location. The provision of location-based services is facilitated by the technological advances in handheld devices and wireless communications, since they enable the widespread usage of mobile devices that can report their current geographic position.

In the application scenario that we target, location-based services are provided to mobile users as well as to points of interest, through a centralized server. First, mobile users register their preferences to the server, and wish to be informed about the k best points of interest in their vicinity. In this setting, any mobile device can directly communicate with the server, reports its position periodically, and poses queries for interesting locations nearby. Second, points of interest (e.g., restaurants) that match the preferences of users at the current time wish to monitor and detect such users. This would enable direct, real-time advertising of the point of interest to users, as shown in the example of Figure 1, or dynamic improvement of its features (e.g., price) in order to increase its attractiveness when the interest of users is low.

Although the problem of monitoring top-k queries of mobile users has been studied before [6], we address the problem from the perspective of the point of interest. Given a query point of interest q, our objective is to monitor at any time those mobile users that have q in their top-k result set.

Essentially, this is expressed by means of the reverse top-k query.

In more detail, we assume a set of mobile devices m_i, each associated with a weighting vector w_i that represents the user preferences over a set of static and dynamic attributes that characterize each point of interest. Static attributes may include the price or the rating of a point of interest; the dynamic attribute is the distance to the point of interest, that keeps changing as the user moves. For example, the score of a point of interest p for a mobile device m can be given by $f(m,p) = 0.2 \cdot p.price + 0.5 \cdot p.rating + 0.3 \cdot d(m,p)$, where $d(m,p)$ denotes the distance of m to p, and w=[0.2, 0.5, 0.3] is the weighting vector of m. Intuitively, the existence of the dynamic attribute complicates query processing because the dataset of points of interest (i.e., defined by distance, price and rating) differs for each mobile user. Furthermore, given a specific mobile user m, the dataset changes each time m moves, as the distance of m to any point of interest changes.

To this end, in this paper, we define the *distance-based reverse top-k query* for monitoring the reverse top-k result set of a given point of interest, as mobile users move in its vicinity. We identify various properties of distance-based reverse top-k queries that enable efficient monitoring of the result set. Assuming that the distance-based reverse top-k set is available for the current positions of mobile users, our premise is to update the distance-based reverse top-k result set when the users move, without re-computing it from scratch. To the best of our knowledge this is the first paper that studies efficient monitoring of distance-based reverse top-k queries.

The main contributions of our work are:

- We define the distance-based reverse top-k query and analyze its main properties.
- We present two novel algorithms that exploit the properties of distance-based reverse top-k queries, in order to monitor efficiently the reverse top-k result set when the mobile devices move.
- We conduct a thorough experimental evaluation that demonstrates the efficiency of our approach. Our algorithms consistently outperform the naive approach in all examined setups by a few orders of magnitudes.

The remaining of this paper is organized as follows: Section 2 reviews the related work. Then, in Section 3, we provide the necessary definitions and present the problem statement. In Section 4, we present our algorithms for monitoring distance-based reverse top-k queries. The experimental evaluation is presented in Section 5. Finally, we conclude the paper in Section 6.

2. RELATED WORK

Top-k queries have been proposed for the retrieval of a limited set of the k highest ranked data objects based on a user-defined preference function [2, 3, 8]. Reverse top-k queries [10] focus on the reverse problem; given a query point q, identify the preference functions whose top-k result set contains q. Numerous applications can benefit from reverse top-k queries, including identifying the most influential data objects [11].

Symbol	Description
S	Dataset of points of interest
$\|S\|$	Cardinality of S
p, q	Points of interest
p^x,p^y	The spatial location of p
d_i	The i-th dimension describing p
$p[i]$	Value of dimension d_i of p
M	Set of mobile devices/users
$\|M\|$	Cardinality of M
m	A mobile device/user
m^x,m^y	The spatial location of m
f	Preference function associated with m
w	n-dimensional weighting vector of m
$d()$	Distance function
$\mathcal{P}^{p_i}(q, r_i, m)$	Priority area of q based on p_i for device m
$\mathcal{P}(q, r, m)$	Global priority area of q for device m
$\mathcal{S}(q, R, m)$	Safe area of q for device m

Table 1: Overview of symbols.

Location-based query processing has attracted significant attention lately. In [6], techniques are presented for processing the location-based top-k query that involves both spatial and non-spatial attributes over data that is broadcast in wireless networks. This query retrieves the k geographical data objects with highest scores for a given mobile user. In contrast, we are interested in monitoring the reverse top-k result set of a given point of interest, as users move around. Moreover, our setup is different as we do not assume that data objects are broadcast over the network. Monitoring of other types of continuous queries, such as k-nearest neighbor [14] or skyline queries [5, 9] over moving objects is also related to our work.

Furthermore, top-k monitoring over data streams [7, 13] bears some similarity with our problem. However, the important difference is that in our case the dataset is continuously updated due to the mobility of devices.

Our work also differs from approaches that consider data placed on mobile devices. Huang et al. [4] assume a setting with mobile devices communicating via ad-hoc networks and study skyline queries that involve spatial constraints. Their techniques aim to reduce both the communication cost and the execution time on each single device. Similarly, Vlachou et al. [12] study bandwidth-constrained skyline computation over mobile devices that maintain local datasets. In contrast, in our application scenario, the mobile devices only pose queries for points of interest, while the query processing takes place on a centralized server that keeps track of their movement.

3. PROBLEM STATEMENT

In this section, we first provide the necessary definitions, and then we formalize the problem statement.

3.1 Definitions

Consider a dataset S (with cardinality $|S|$) of points of interest (e.g., restaurants) described by a set of $n-1$ dimensions $\{d_1, \ldots, d_{n-1}\}$. Each dimension represents a numerical scoring attribute, such as price or rating of a restaurant. The

points of interest are maintained as database objects[1], and each object can be represented as a multidimensional point $p \in S$, such that $p = \{p[1], \ldots, p[n-1]\}$, where $p[i]$ is a value on dimension d_i. Thus, the values $p[i]$ are numerical non-negative scores that evaluate the corresponding attributes of database objects. We further assume that smaller score values are preferable, without loss of generality. In addition, each point of interest p has a spatial location denoted as (p^x, p^y). An overview of the basic symbols used in this paper can be found in Table 1.

Let M denote the set of mobile devices, where each device $m \in M$ corresponds to a mobile user and is also associated with a spatial location (m^x, m^y) that records its current position. Let $d()$ denote the distance function of any two spatial locations, e.g., $d(m, p)$ denotes the distance of mobile device m to a point of interest p. Without lost of generality, in this paper, we assume the distance between a mobile device and a point of interest is expressed by the Euclidean distance. However, any other distance function can be employed, as long as it obeys the triangular inequality.

Furthermore, each device is characterized by a user-defined scoring function f that aggregates the individual scores of an object into an overall score. The most important and commonly used case of scoring functions is the weighted sum function, also called linear. Each dimension d_i has an associated query-dependent weight $w[i]$ indicating d_i's relative importance for the specific user. We define the *device-specific score* of an object p for a device m as follows.

DEFINITION 1. (Device-specific score of object p for m)*: The score $f(m, p)$ of object p for device m is defined as a weighted sum of the $n-1$ individual scores and the distance $d(m, p)$ between p and m: $f(m, p) = \sum_{i=1}^{n-1} w[i] \cdot p[i] + w[n] \cdot d(m, p)$, where $w[i] \geq 0$ $(1 \leq i \leq n)$ and $\sum_{i=1}^{n} w[i] = 1$.*

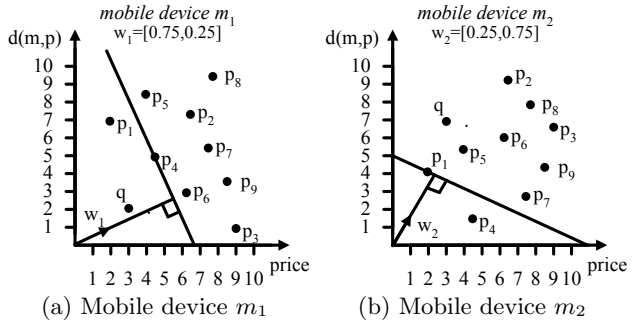
Based on this scoring function, we can also define the concept of distance-based top-k queries from the aspect of a mobile device in the afore-described context. In this query type, the user is interested not only to minimize the scores $p[i]$ of any object p, but in addition the distance $d(m, p)$ of the current location of m to p. Notice that, since the weights represent the *relative* importance between different attributes, the assumption $\sum_{i=1}^{n} w[i] = 1$ does not influence the definition of distance-based top-k queries.

DEFINITION 2. (Distance-based top-k query)*: Given a positive integer k and the weighting vector w of device m, the result set $TOP_k(w)$ of the distance-based top-k query for device m retrieves a set of objects such that $TOP_k(w) \subseteq S$, $|TOP_k(w)| = k$ and $\forall p_i, p_j : p_i \in TOP_k(w)$, $p_j \in S - TOP_k(w)$ it holds that $f(m, p_i) \leq f(m, p_j)$.*

A delicate situation arises when two (or more) objects share the same score for the k-th position. In this case, for simplicity reasons, we assume that it suffices to report any one of them as k-th result.

Geometrically, in the Euclidean space a linear top-k query can be represented by a vector w. Consider the dataset S depicted in Figure 2 defined by one static (price) and one dynamic attribute (distance). In the n-dimensional space, the hyperplane which is perpendicular to vector w and contains a point p defines the score of point p, and all points

[1]In this paper, we will use the term object to refer to a point of interest.



(a) Mobile device m_1 (b) Mobile device m_2

Figure 2: Distance-based top-k queries.

lying on the same hyperplane have the same score based on w. The rank of a point p based on a weighting vector w is equal to the number of the points enclosed in the half-space defined by the perpendicular hyperplane and the origin of the data space.

In Figure 2(a), assuming that q does not belong to the dataset, p_4 is the top-2 object for a mobile user m_1 with w_1=[0.75,0.25], since data objects p_1 and p_4 are enclosed in the corresponding half-space. Figure 2(b) depicts the distance-based top-k query that corresponds to another mobile device m_2. Even though the price of the objects remains the same, the values of the dynamic attribute $d(m, q)$ are different compared to Figure 2(a), since the locations of m_1 and m_2 are different. Essentially, the distance-based top-k query of m_2 is processed over a different dataset than m_1. Furthermore, for m_1 low price is more preferable than small distance (i.e., $w_1[1]$ is higher than $w_1[2]$), while m_2 prefers nearby restaurants rather than cheap ones.

3.2 Problem Formulation

In this paper, we are interested in a dynamic variant of reverse top-k queries [10], namely distance-based reverse top-k queries. Given a query object q, the distance-based reverse top-k query identifies all mobile devices (i.e., weighting vectors) for which q belongs to the distance-based top-k result set. The formal definition of the distance-based reverse top-k query follows. Notice that this definition corresponds to the bichromatic version of the reverse top-k query (cf. [10]), which assumes that a set of user preferences W exists (in this case W is the set of weighting vectors for all mobile devices).

DEFINITION 3. (Distance-based reverse top-k query)*: Assume a query object q and a positive number k, as well as a set S of objects and a set M of mobile devices m_i each associated with a weighting vector $w_i \in W$. The distance-based reverse top-k result set ($RTOP_k(q)$) of q contains a mobile device m_i, if and only if $\exists p \in TOP_k(w_i)$ such that $f(m_i, q) \leq f(m_i, p)$.*

In the general case, query point q may or may not belong to the dataset S. Without loss of generality we assume that $q \notin S$. For the sake of brevity, in the rest of this paper we denote a query point $q \in TOP_k(w_i)$, instead of $\exists p \in TOP_k(w_i)$ such that $f(m_i, q) \leq f(m_i, p)$.

In Figure 2, we observe that m_1 belongs to the reverse top-2 result set (i.e., $RTOP_2(q)$), since no data point has a smaller score value based on w_1. On the other hand, m_2

does not belong to $RTOP_2(q)$ since it has a higher score than p_4 and p_1.

In the following, we formulate the problem of monitoring distance-based reverse top-k queries over mobile devices.

PROBLEM 1. (Monitoring distance-based reverse top-k query): *Given a query object q, a positive number k, a set of objects $p \in S$ and a set of devices $m \in M$, maintain $RTOP_k(q)$ while the devices move.*

A major difference of the distance-based reverse top-k query to the traditional reverse top-k query is that for two identical weighting vectors w_1 and w_2 ($w_1 \equiv w_2$) belonging to different devices m_1 and m_2 respectively, it is possible that $m_1 \in RTOP_k(q)$ while $m_2 \notin RTOP_k(q)$. The reason is that the devices may be located at different positions, thereby the dataset is different for each device, as depicted in Figure 2. The dynamic nature of this problem, makes existing techniques inappropriate for processing distance-based reverse top-k queries. The only applicable algorithm for computing the distance-based reverse top-k query evaluates a distance-based top-k query for each weighting vector w_i of each device m_i ($i \in [1, |M|]$). Instead, we provide efficient techniques for maintaining the reverse top-k result set up-to-date when the devices move, assuming that the reverse top-k result set of a past point of time is available.

4. MONITORING ALGORITHMS

A straightforward way to monitor the results of the distance-based reverse top-k query $RTOP_k(q)$ for a given point of interest q is to compute the top-k query based on w_i for each mobile device m_i ($i \in [1, |M|]$) every time the location of m_i changes. Then, the distance-based reverse top-k result set can be either augmented with m_i (if $q \in TOP_k(w_i)$), or m_i can be removed from it (if $q \notin TOP_k(w_i)$). We refer to this plain algorithm as *Naive*. The most obvious disadvantage of *Naive* is redundant processing of top-k queries, even when the movement of the mobile device does not affect the set $RTOP_k(q)$ of query point q. To avoid this shortcoming, we introduce an algorithm that processes only the top-k queries of those mobile devices that may affect the distance-based reverse top-k result set of q.

4.1 The DRT Algorithm

Let m denote a mobile device located at a position where the distance-based top-k result set $TOP_k(w)$ has been computed. Furthermore, let m' the device located at a new position. We use $RTOP_k(q)$ and $RTOP'_k(q)$ to denote the distance-based reverse top-k result set of q at the two locations respectively. The following lemma shows that there exists an upper bound on the difference of the device-specific score between the two locations.

LEMMA 1. *Given a distance-based reverse top-k query q and a device m that moves to position m', the upper bound in the change of the device-specific score of q is $\Delta f = w[n] \cdot d(m, m')$.*

PROOF. The device-specific score of q at the new position of m' is $f(m', q) = \sum_{i=1}^{n-1} w[i] \cdot q[i] + w[n] \cdot d(m', q)$. Based on the triangular inequality, it holds that $d(m, q) - d(m, m') \leq d(m', q) \leq d(m, q) + d(m, m')$. Therefore, we derive that $\sum_{i=1}^{n-1} w[i] \cdot p[i] + w[n] \cdot (d(m, q) - d(m, m')) \leq f(m', q) \leq \sum_{i=1}^{n-1} w[i] \cdot p[i] + w[n] \cdot (d(m, q) + d(m, m'))$. This results in $f(m, q) - w[n] \cdot d(m, m') \leq f(m', q) \leq f(m, q) + w[n] \cdot d(m, m')$. Hence, the change of the device-specific score is at most $\Delta f = |f(m', q) - f(m, q)| = w[n] \cdot d(m, m')$. □

Algorithm 1 *DRT*.

```
1: Input: Query q, k, RTOP_k(q)
2: Output: Updated RTOP_k(q)
3: for (i ∈ [1, |M|]) do
4:   if (m_i ∈ RTOP_k(q) and f(m'_i, q) > f_k − 2 · w_i[n] · d(m_i, m'_i)) then
5:     compute TOP_k(w_i)
6:     if (q ∉ TOP_k(w_i)) then
7:       RTOP_k(q) ← RTOP_k(q) − {m_i}
8:     end if
9:   else
10:    if (m_i ∉ RTOP_k(q) and f(m'_i, q) ≤ f_k + 2 · w_i[n] · d(m_i, m'_i)) then
11:      compute TOP_k(w_i)
12:      if (q ∈ TOP_k(w_i)) then
13:        RTOP_k(q) ← RTOP_k(q) ∪ {m_i}
14:      end if
15:    end if
16:  end if
17: end for
18: return RTOP_k(q)
```

Let f_k denote the score of the k-th point in m's top-k result set. Based on Lemma 1, the change of the difference in the scores of q and any other point p due to the movement of m to m' is at most $2 \cdot w[n] \cdot d(m, m')$. Therefore, we derive that if w belongs to $RTOP_k(q)$, then w will also belong to $RTOP'_k(q)$, if $f(m', q) \leq f_k - 2 \cdot w[n] \cdot d(m, m')$. On the other hand, if w does not belong to $RTOP_k(q)$, then w cannot belong to $RTOP'_k(q)$, if $f(m', q) > f_k + 2 \cdot w[n] \cdot d(m, m')$. In any of the above cases, we avoid the computation of the top-k query defined by w. These properties guide the design of Algorithm 1 for monitoring distance-based reverse top-k queries, which is referred to as *DRT* (Distance-based Reverse Top-k algorithm).

Algorithm 1 avoids redundant top-k evaluations in the following two cases. In the first case (line 4), assume that w_i belongs to $RTOP_k(q)$ and the score $f(m_i, q)$ is much better than the score f_k of the k-th result. If the distance $d(m_i, q)$ of a device m_i increases only slightly, i.e., $d(m_i, q) \approx d(m'_i, q)$, then m_i essentially remains at the same area. Thus, the score $f(m'_i, q)$ cannot deteriorate enough to cause w_i's removal from $RTOP_k(q)$. *DRT* detects this situation and avoids the top-k computation. The second case (line 10) involves devices m_i whose w_i does not belong to $RTOP_k(q)$. When the distance $d(m_i, q)$ of a device m_i to q decreases only slightly, again this change may not be sufficient to make q a top-k result for w_i. *DRT* first checks whether the movement is sufficient to alter the $RTOP_k(q)$ set, thereby rendering the top-k computation unnecessary when this is the case. In contrast, *Naive* will perform the top-k computation in vain in both the aforementioned cases.

4.2 The DRT* Algorithm

Although *DRT* drastically improves the performance of query processing compared to *Naive*, we aim to reduce the number of required top-k computations even further.

Given query point q and a mobile device m, the main idea

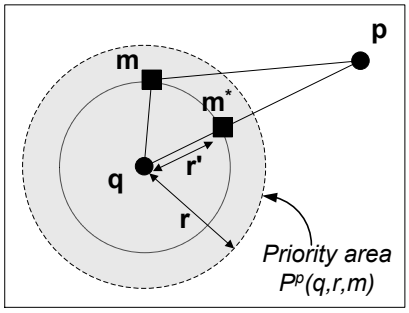


Figure 3: Priority area $\mathcal{P}^p(q,r,m)$ of q based on device m and object p.

is to define a *safe area* around q (for each mobile device) with the following interesting property: as long as the device m is located in the safe area, q belongs to the top-k result of m, or equivalently, $w \in RTOP_k(q)$. Put differently, we can avoid any top-k computation for m with respect to q, as long as m does not cross the boundary of the safe area. The safe area denoted as $\mathcal{S}(q,R,m)$ is essentially the area defined by a circle centered at q with radius R. Notice that the radius R of the safe area $\mathcal{S}(q,R,m)$ depends only on w, and is independent of the actual movement of m. Therefore it can be computed once for each device m and remains valid as long as the preference function w does not change.

In the following, we first define the concept of the priority area, which leads to the definition of the safe area. The *priority area* $\mathcal{P}^p(q,r,m)$ of q with respect to a device m and a point of interest p (other than q) is essentially a region in space, such that when m is located in the priority area, q is higher ranked than p in m's distance-based top-k query. Similarly to the safe area, the priority area is defined as a circle centered at q with radius r. Theorem 1 defines an appropriate radius r, so that the aforementioned property of the priority area holds.

THEOREM 1. *Assume a query point q, a device m, another point of interest p and the priority area $\mathcal{P}^p(q,r,m)$ with respect to p and m. If the radius r equals:*

$$r = \frac{1}{2\cdot w[n]}(w[n]\cdot d(q,p) + \sum_{j=1}^{n-1} w[j]\cdot(p[j]-q[j]))$$

then as long as $d(m,q) \le r$, q is ranked higher than p with respect to m, i.e., $f(m,q) \le f(m,p)$.

PROOF. Assume that the device m is located in the priority area, i.e., $d(m,q) \le r$. Then, it holds that:

$$d(m,q) \le r$$
$$\Rightarrow 2\cdot w[n]\cdot d(m,q) \le w[n]\cdot d(q,p) + \sum_{j=1}^{n-1} w[j]\cdot(p[j]-q[j])$$
$$\Rightarrow \sum_{j=1}^{n-1} w[j]\cdot q[j] + w[n]\cdot d(m,q) \le$$
$$\sum_{j=1}^{n-1} w[j]\cdot p[j] + w[n]\cdot d(q,p) - w[n]\cdot d(m,q) \quad (1)$$

Let m^* denote the mobile device located on the line defined by q and p and at distance r' from q, i.e., $d(m^*,q) = d(m,q) = r'$ (2), as depicted in Figure 3. In this position, it holds that: $d(q,p) = d(m^*,p) + d(m^*,q) \Rightarrow d(q,p) - d(m^*,q) = d(m^*,p)$ (3). By combining inequality (1) with equality (2) we derive that $\sum_{j=1}^{n-1} w[j]\cdot q[j] + w[n]\cdot d(m,q) \le \sum_{j=1}^{n-1} w[j]\cdot p[j] + w[n]\cdot d(q,p) - w[n]\cdot d(m^*,q)$, which based on equality (3) leads to $\sum_{j=1}^{n-1} w[j]\cdot q[j] + w[n]\cdot d(m,q) \le \sum_{j=1}^{n-1} w[j]\cdot p[j] + w[n]\cdot d(m^*,p) \Rightarrow f(m,q) \le f(m^*,p)$ (4).

Furthermore, it holds that $f(m^*,p) \le f(m,p)$ (5), since $d(m,p)+d(m,q) \ge d(q,p) = d(m^*,q)+d(m^*,p) \Rightarrow d(m,p) \ge d(m^*,p)$. From inequalities (4) and (5) we derive that as long as m is located in the priority area: $f(m,q) \le f(m,p)$. □

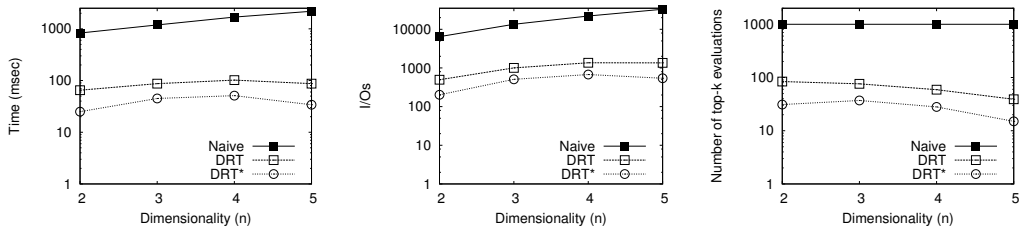
Algorithm 2 *DRT*.*

```
1: Input: Query q, k, RTOP_k(q)
2: Output: Updated RTOP_k(q)
3: S(q, R_i, m_i) is the safe area of q based on m_i
4: for (i ∈ [1, |M|]) do
5:   if (R_i ≥ 0) then
6:     if (m_i ∈ RTOP_k(q)) then
7:       if (d(m'_i, q) > R_i and (f(m'_i, q) > f_k − 2 · w_i[n] ·
         d(m_i, m'_i))) then
8:         compute TOP_k(w_i)
9:         if (q ∉ TOP_k(w_i)) then
10:          RTOP_k(q) ← RTOP_k(q) − {m_i}
11:        end if
12:      end if
13:    else if (m_i ∉ RTOP_k(q)) then
14:      if (d(m'_i, q) ≤ R_i) then
15:        RTOP_k(q) ← RTOP_k(q) ∪ {m_i}
16:      else
17:        if (f(m'_i, q) ≤ f_k + 2 · w_i[n] · d(m_i, m'_i)) then
18:          compute TOP_k(w_i)
19:          if (q ∈ TOP_k(w_i)) then
20:            RTOP_k(q) ← RTOP_k(q) ∪ {m_i}
21:          end if
22:        end if
23:      end if
24:    end if
25:  end if
26: end for
27: return RTOP_k(q)
```

Obviously, different objects p_i define different priority areas $\mathcal{P}^{p_i}(q,r_i,m)$ for q and device m. The differentiating factor of these priority areas is the radius r_i which varies for different p_i. As we are interested in defining a *global priority area* $\mathcal{P}(q,r,m)$ for query point q and device m, and not on an individual point basis, we introduce Lemma 2, which defines a global priority area for reverse top-1 queries. Conceptually, the individual priority areas $\mathcal{P}^{p_i}(q,r_i,m)$ define the locations of m where it is certain that q is ranked higher than p_i, while the global priority area $\mathcal{P}(q,r,m)$ defines the locations where it is certain that no point p_i is ranked higher than q.

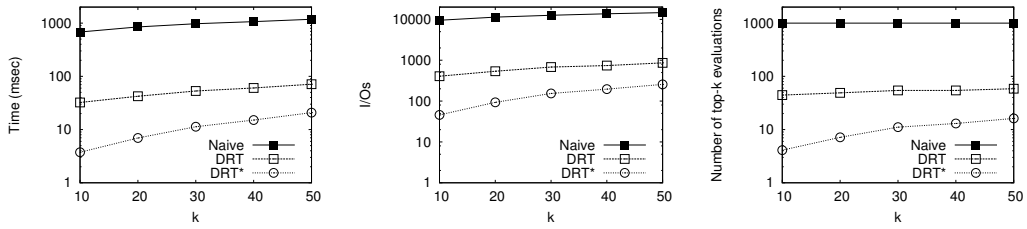
LEMMA 2. *Given a set S of points of interest and the individual priority areas $\mathcal{P}^{p_i}(q,r_i,m)$ $\forall p_i \in S$, the global priority area $\mathcal{P}(q,r,m)$ of query point q with respect to device m defined by radius: $r = \min_{i=1}^{|S|} r_i$ delineates an area where q is the top-1 object for device m.*

PROOF. By contradiction. Assume that $q \notin TOP_1(w)$ for device m and $d(m,q) \le r$. Then, there exists a point of interest p_j ($p_j \neq q$), such that $p_j \in TOP_1(w)$. We distinguish two cases: (a) $r_j \ge d(m,q)$, then the definition of $\mathcal{P}^{p_j}(q,r_j,m)$ states that q is ranked higher than p_j, which is a contradiction, since $p_j \in TOP_1(w)$, or (b) $r_j < d(m,q)$, which combined with $d(m,q) \le r$ produces $r_j < r$, which is a contradiction to $r = \min_{i=1}^{|S|} r_i$. □



(a) Time (b) I/Os (c) Number of top-k

Figure 4: Comparative performance vs. dimensionality n for uniform (UN) datasets.



(a) Time (b) I/Os (c) Number of top-k

Figure 5: Comparative performance varying k for uniform (UN) datasets.

We are now ready to introduce our main theorem, which defines a safe area $\mathcal{S}$ for reverse top-k queries with arbitrary values of k ($k \geq 1$). Similar to the global priority area, the safe area defines the locations where it is certain that fewer than k points p_i are ranked higher than q.

THEOREM 2. *Given a set S of points of interest and the priority area $\mathcal{P}^{p_i}(q, r_i, m)$ of query point q with respect to device m $\forall p_i \in S$, the radius of the safe area $\mathcal{S}(q, R, m)$ of q with respect to m is defined as the k-th smallest radius r_k of the individual priority areas, and it holds that if $d(m, q) \leq R$ then $q \in TOP_k(w)$.*

PROOF. By contradiction. Assume that $q \notin TOP_k(w)$ for device m and $d(m, q) \leq R$. Then, there exist k points of interest p_j ($p_j \neq q$), such that $p_j \in TOP_k(w)$. We distinguish two cases: (a) $\exists j : r_j \geq d(m, q)$, then the definition of $\mathcal{P}^{p_j}(q, r_j, m)$ states that q is ranked higher than p_j, which is a contradiction, since $p_j \in TOP_k(w)$ and $q \notin TOP_k(w)$, or (b) $\forall j : r_j < d(m, q)$, which combined with $d(m, q) \leq R$ produces $\forall j : r_j < R$, which is a contradiction because R is equal to the k-th smallest radius r_k. □

Based on the concept of the safe area, we propose a novel algorithm, called *DRT**, that monitors efficiently the result of the distance-based reverse top-k query. *DRT** enhances the functionality of *DRT* by exploiting the safe area of a query point to achieve more savings in computational cost.

Algorithm 2 provides the necessary pseudocode. An interesting situation arises when the radius R_i of a safe area $\mathcal{S}(q, R_i, m_i)$ of q with respect to m_i is negative (line 5). In this case, we can safely exclude device m_i from further processing, because q cannot appear in m_i's top-k result set. In addition, the property of the safe area outlined in Theorem 2 is exploited to avoid the computation of the top-k result (lines 7,14). In line 7, if a device m_i that belongs to $RTOP_k(q)$ moves inside the safe area, then we know that m_i still belongs to $RTOP_k(q)$. In line 14, if a device m_i that does not belong to $RTOP_k(q)$ moves inside the safe area, we can immediately report that w_i now belongs to the updated $RTOP_k(q)$. In both cases, the algorithm avoids the evaluation of the respective top-k query.

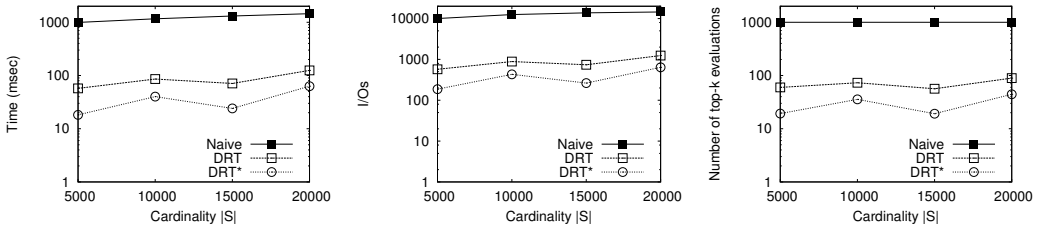
5. EXPERIMENTAL STUDY

In our experimental evaluation, we implemented the proposed query processing algorithms to study their performance, and simulated the networking aspects. The simulator was implemented in Java. We evaluated our two algorithms *DRT* and *DRT**, and also *Naive* for comparative purposes. The distance-based top-k query is processed using a modified branch-and-bound algorithm over an R-tree [8]. The R-tree employed for the underlying top-k processing uses a buffer size of 100 blocks and the block size is 8KB.

5.1 Experimental Setup

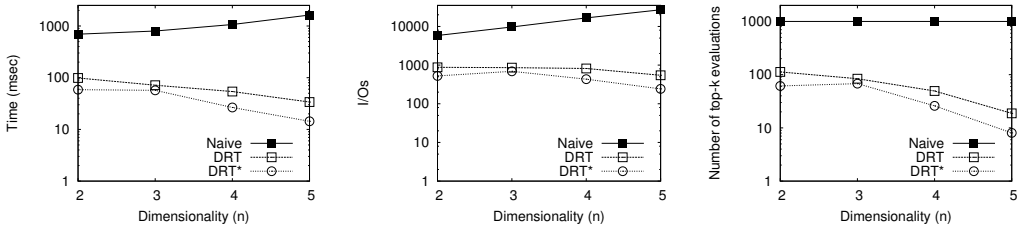
The mobility scenario is modeled using multiple discrete steps that represent the sequence of locations that devices visit as time elapses. Given a mobile device, first its direction is determined by selecting an angle uniformly at random. Then, at each step, the device travels a distance that follows a Gaussian distribution with default value of mean equal to 10% of the maximum value of the geographical coordinate, and standard deviation equal to 10% of the mean value. Notice that using smaller values of mean is less challenging, since devices would travel smaller distances at each step, and fewer mobile devices would be added or removed from the result set at each step.

We conduct experiments varying several parameters as well as data distributions to test the scalability of all algorithms. For the weighting vectors W, two different data distributions are examined, namely uniform (UN) and clustered (CL). For the locations of the points of interest and



(a) Time (b) I/Os (c) Number of top-k

Figure 6: Comparative performance vs. cardinality $|S|$ for uniform (UN) datasets.



(a) Time (b) I/Os (c) Number of top-k

Figure 7: Comparative performance vs. dimensionality n for clustered (CL) locations S and M.

the mobile devices we use uniform (UN) and clustered (CL) distribution. In both cases, for the clustered generator we generate a set of 5 Gaussian clusters with deviation equal to 0.05, as described in [10]. When both datasets are CL, the cluster centroids are common. We also vary the total number of attributes n (dimensionality) from 2 to 5, and the cardinality (S=5K-20K) of points of interest. We use synthetic data collections, namely uniform (UN), correlated (CO) and anti-correlated (AC), for studying the effect of different distributions of the static attributes of points of interest. For the uniform dataset, all attribute values are generated independently using a uniform distribution. The correlated and anti-correlated datasets are generated as described in [1]. The query points follow the data distribution of the points of interest (both in terms of location and static attributes). Finally, we vary the value k from 10 to 50.

All experiments are repeated 20 times and the average values are depicted in all charts. Unless explicitly mentioned, we use the default setup of: $|S|$=10K, $|W|$=10K, n=3, k=30, and uniform distribution for S, W, the locations of mobile devices and points of interest. Our metrics are aggregated over the entire movement of devices include: a) the total execution time, b) the number of I/Os, c) the number of top-k evaluations. Notice that the number of top-k evaluations is the main factor that influences the performance of our algorithms and it is independent of the efficiency of the particular implementation. In our experiments, we assume that at some given point of time the reverse top-k set is known and we monitor it for 1000 steps. In each step one mobile device moves, since the movements of mobile devices are independent and one movement does not influence the result sets of other mobile devices.

5.2 Experimental Results

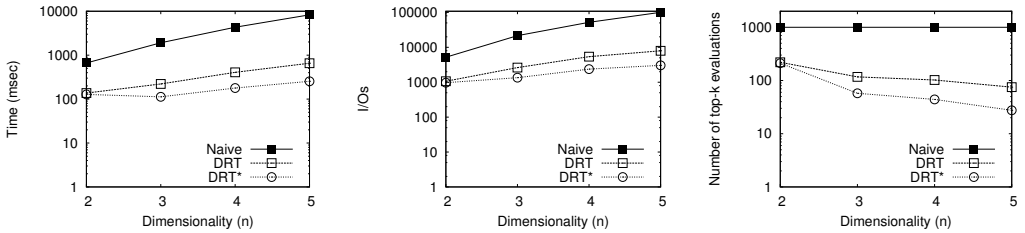
In Figure 4, we study the effect of varying the number n of attributes (dimensionality) of points of interest. Recall that n means $n-1$ static attributes and 1 dynamic attribute. The results show that *DRT* improves *Naive* by more than one order of magnitude in all metrics. *DRT** manages to improve the performance of *DRT* even further, by exploiting the concept of safe areas. Notice that the number of top-k evaluations drops for increased dimensionality. This is because a higher number of scoring attributes reduces (on average) the effect of distance on the distance-based score, leading to fewer modifications of the $RTOP_k$ set. Our algorithms discard unnecessary top-k evaluations effectively, therefore their performance improves in terms of top-k evaluations.

Then, in Figure 5, we gradually increase the value k of reverse top-k queries. It is noteworthy to mention that *DRT** is two orders of magnitude better than *Naive*. Especially in the case of queries with small k, *DRT** demonstrates its efficiency. The performance of all algorithms deteriorates with increasing k values, because more mobile devices belong to the $RTOP_k$ set, which in turn makes monitoring more challenging.

The effect of increasing the cardinality of points of interest is studied in Figure 6. Again, *DRT** shows the best performance, followed by *DRT*. Notice that the comparative advantage of both algorithms over *Naive* is sustained, regardless of number of points of interest, which demonstrates the scalability of our algorithms.

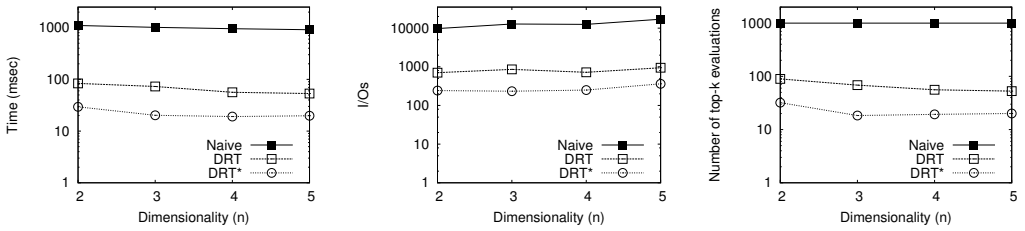
An interesting result is obtained by testing clustered distributions of points of interest and mobile devices in Figure 7. As the dimensionality grows, our proposed algorithms improve their performance. The reason is that our techniques exploit the fact that devices are located nearby points of interest, and avoid query processing in more occasions than in the case of uniform distribution. In contrast, *Naive* cannot benefit from the clustered distribution and evaluates more top-k queries, resulting in higher processing cost for increased dimensionality.

In addition, we tested increasing values of k in the same



(a) Time (b) I/Os (c) Number of top-k

Figure 8: Comparative performance vs. dimensionality n for anti-correlated (AC) scoring attributes.



(a) Time (b) I/Os (c) Number of top-k

Figure 9: Comparative performance vs. dimensionality n for correlated (CO) scoring attributes.

setup, and obtained similar results as in the case of uniform distributions (Figure 5), only the performance gap between *DRT** and *DRT* was smaller. Furthermore, we examined the case of clustered distribution of points of interest and uniform distribution of mobile devices. In all metrics, *DRT** showed the best performance consistently. These charts are omitted due to lack of space.

Figure 8 shows the results obtained when the anti-correlated dataset is employed for the static attributes. This is a hard setup for all algorithms, therefore the absolute values of the metrics are higher. Interestingly, the cost of all algorithms in terms of time and I/O increases with dimensionality, even though our algorithms require fewer top-k evaluations for higher values of dimensionality. The reason is that each top-k evaluation is more costly than in the case of uniform distribution, due to the anti-correlated dataset, and the cost of each top-k evaluation increases rapidly with increasing dimensionality. In all cases, *DRT** demonstrates the best performance.

We also tested the correlated dataset for static attributes and the results are depicted in Figure 9. In this case, the performance is rather stable with increased dimensionality, and again our algorithms are significantly better than *Naive*.

6. CONCLUSIONS

In this paper, we proposed efficient algorithms for processing distance-based reverse top-k queries over mobile devices. To this end, we defined the distance-based reverse top-k query and analyzed its properties. The technical challenge that we addressed is that the reverse top-k computation is performed over a combination of static and dynamic attributes. We proposed two novel algorithms that avoid the underlying distance-based top-k computation when possible, resulting in significant performance gains. Our experimental evaluation demonstrates the efficiency of our algorithms in all examined setups.

7. REFERENCES

[1] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proc. of ICDE*, pages 421–430, 2001.

[2] S. Chaudhuri and L. Gravano. Evaluating top-k selection queries. In *Proc. of VLDB*, pages 397–410, 1999.

[3] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proc. of PODS*, pages 102–113, 2001.

[4] Z. Huang, C. S. J. H. Lu, and B. C. Ooi. Skyline queries against mobile lightweight devices in MANETs. In *Proc. of ICDE*, page 66, 2006.

[5] Z. Huang, H. Lu, B. C. Ooi, and A. K. H. Tung. Continuous skyline queries for moving objects. *IEEE Trans. Knowl. Data Eng.*, 18(12):1645–1658, 2006.

[6] H. Jung, B. K. Cho, Y. D. Chung, and L. Liu. On processing location based top-k queries in the wireless broadcasting system. In *Proc. of SAC*, pages 585–591, 2010.

[7] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In *Proc. of SIGMOD*, pages 635–646, 2006.

[8] Y. Tao, V. Hristidis, D. Papadias, and Y. Papakonstantinou. Branch-and-bound processing of ranked queries. *Inf. Syst.*, 32(3):424–445, 2007.

[9] L. Tian, L. Wang, P. Zou, Y. Jia, and A. Li. Continuous monitoring of skyline query over highly dynamic moving objects. In *Proc. of MobiDE*, pages 59–66, 2007.

[10] A. Vlachou, C. Doulkeridis, Y. Kotidis, and K. Nørvåg. Reverse top-k queries. In *Proc. of ICDE*, pages 365–376, 2010.

[11] A. Vlachou, C. Doulkeridis, K. Nørvåg, and Y. Kotidis. Identifying the most influential data objects with reverse top-k queries. *PVLDB*, 3(1):364–372, 2010.

[12] A. Vlachou and K. Nørvåg. Bandwidth-constrained distributed skyline computation. In *Proc. of MobiDE*, pages 17–24, 2009.

[13] D. Yang, A. Shastri, E. A. Rundensteiner, and M. O. Ward. An optimal strategy for monitoring top-k queries in streaming windows. In *Proc. of EDBT*, 2011.

[14] X. Yu, K. Q. Pu, and N. Koudas. Monitoring k-nearest neighbor queries over moving objects. In *Proc. of ICDE*, pages 631–642, 2005.