Identifying the Most Influential Data Objects with Reverse Top-k Queries

Akrivi Vlachou[1], Christos Doulkeridis[1], Kjetil Nørvåg[1] and Yannis Kotidis[2]
[1]Norwegian University of Science and Technology (NTNU), Trondheim, Norway
[2]Athens University of Economics and Business (AUEB), Athens, Greece
{vlachou,cdoulk,noervaag}@idi.ntnu.no, kotidis@aueb.gr

ABSTRACT

Top-k queries are widely applied for retrieving a ranked set of the k most interesting objects based on the individual user preferences. As an example, in online marketplaces, customers (users) typically seek a ranked set of products (objects) that satisfy their needs. Reversing top-k queries leads to a query type that instead returns the set of customers that find a product appealing (it belongs to the top-k result set of their preferences). In this paper, we address the challenging problem of processing queries that identify the top-m most *influential products* to customers, where influence is defined as the cardinality of the reverse top-k result set. This definition of influence is useful for market analysis, since it is directly related to the number of customers that value a particular product and, consequently, to its visibility and impact in the market. Existing techniques require processing a reverse top-k query for each object in the database, which is prohibitively expensive even for databases of moderate size. In contrast, we propose two algorithms, SB and BB, for identifying the most influential objects: SB restricts the candidate set of objects that need to be examined, while BB is a branch-and-bound algorithm that retrieves the result incrementally. Furthermore, we propose meaningful variations of the query for most influential objects that are supported by our algorithms. Our experiments demonstrate the efficiency of our algorithms both for synthetic and real-life datasets.

1. INTRODUCTION

Top-k queries are widely applied for retrieving the k most interesting objects based on the individual user preferences. Clearly, an object (product) that is highly ranked by many users (customers) has obviously a wider visibility and impact in the market. Thus, an intuitive definition of the influence of a product in the market is the number of customers that consider it appealing (the product belongs to their top-k results) based on their preferences. Identifying the most *influential objects* from a given database of products is important for market analysis and decision-making and is beneficial for several real-life applications.

Optimizing direct marketing. Consider a mobile phone company that aims to introduce a set of new phones in the market. After collecting customer feedback regarding their individual preferences, with respect to favored features of devices, the company is interested in finding the subset of phones that are top-ranked according to customer preferences, in order to perform direct marketing to its customers in a timely fashion. This subset of products forms the most influential products of the company.

Real estate market. A real estate agency maintains a database of houses that are available for sale in the market, including its own houses. In addition, the agency regularly collects the current preferences of the buyers for favored features of houses, as determined by the demand in the market. In order to facilitate a beneficial strategy for the promotion of its own houses, the agency needs to identify the most influential houses in the market. This set of houses have a higher probability to be sold, therefore the agents need to prioritize showing such houses to buyers. Moreover, when such houses are sold and are not in the market anymore, it is imperative to find the new influential houses to show to buyers.

Given a product, reverse top-k queries [9] have been recently proposed to retrieve the customers to whom the product belongs to their top-k result set. In this paper, we capitalize on reverse top-k queries to define the *influence score* of a product as the cardinality of its reverse top-k result set. Then, the problem of identifying the most influential products in the market can be formulated as a query that selects the m products with highest influence score. Unfortunately, efficient processing of this novel query type is not currently supported by existing techniques, as obtaining the most influential products requires computing a reverse top-k query for each product in the database, which is prohibitively expensive even for databases of moderate size.

The contributions of this paper are:

- We formulate the problem of identifying the most influential data objects based on reverse top-k queries. Each object is characterized by its influence score, which equals to the cardinality of the reverse top-k query result set (Section 4).
- We propose two algorithms for identifying the most influential data objects: the first (SB) exploits the properties of the skyband set [7] to restrict the number of candidate result objects (Section 5), whereas the second (BB) is a branch-and-bound algorithm that employs upper bounds on influence score and result sharing to report the result incrementally (Section 6). In addition, we provide theoretical proofs for the correctness and efficiency of our algorithms.
- We introduce meaningful variants of the query for most influential objects that are useful in practice, and we show that they can be supported by our algorithms with minor modifications (Section 7).
- We demonstrate the efficiency of our algorithms using both synthetic and real-life datasets (Section 8).

Section 2 reviews related work, while the preliminaries are presented in Section 3. Finally, in Section 9, we conclude the paper.

2. RELATED WORK

Top-k have been studied for retrieving a limited set of k data object based on a user-defined preference function [2, 8]. Reverse top-k queries [9] have been recently proposed for assessing the impact of a potential product in the market, based on the number of users that deem this product as one of their top-k products according to their preferences. However, identifying a ranked set of the top-m most influential products is extremely costly, as it requires computing a reverse top-k query for each product in the database, and then rank the products based on their result cardinality. In this paper, we propose more efficient algorithms for solving the problem, including an incremental branch-and-bound algorithm.

Several novel queries have been proposed lately that focus on the aspect of the product manufacturer, rather than the customer. DADA [5] aims to help manufactures position their products in the market, based on dominance relationship analysis. Creating *competitive products* has been recently studied in [10]. Nevertheless in these approaches, user preferences are expressed as data points that represent preferable products, whereas reverse top-k queries examine user preferences in terms of weighting vectors. Miah *et al.* [6] propose an algorithm that selects the *subset of attributes* that increases the *visibility* of a new product. In [13], *promotion analysis* for business intelligence is studied, which discovers and ranks for a given object the subspaces in which it is highly ranked (also called *interesting subspaces*). A related problem is that of finding the top-k most interesting regions for promotion of an object [12].

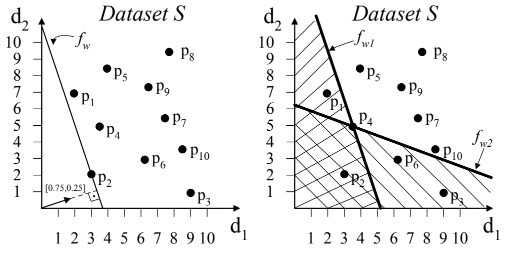
Given a query object q, reverse nearest neighbor queries [4] aim to retrieve those objects that have q as their nearest neighbor based on a similarity function. These objects are said to be influenced by q. Finding the region in space that maximizes the size of bichromatic reverse nearest neighbor is studied in [11]. The reverse skyline set of a query object q can be used to discover objects influenced by q based on dominance relationships [3]. The notion of influence has also been used by other approaches, in order to identify the best objects in a dataset. In spatial databases, an interesting problem is to compute the top-t most influential *spatial sites* [14], where the influence of an object is defined as the total weight of its reverse nearest neighbors. We argue that in the context of studying the impact of products in the market, our definition of influence is meaningful and intuitive, since it inherently integrates the most important factor for assessing the impact, namely the visibility of the product based on user preferences (in terms of top-k queries).

3. PRELIMINARIES

Consider a data space D defined by a set of n dimensions $\{d_1, \ldots, d_n\}$ and a set S of database objects on D with cardinality $|S|$. Each dimension represents a numerical scoring attribute. A database object can be represented as a point $p \in S$, such that p = $\{p[1], \ldots, p[n]\}$, where $p[i]$ is a value on dimension d_i. Without loss of generality, we assume that the values $p[i]$ are numerical non-negative scores and that smaller score values are preferable.

3.1 Top-k Queries

Top-k queries are defined based on a 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 query. The aggregated score $f_w(p)$ for



(a) Top-k query (b) Reverse top-k query

Figure 1: Top-k and reverse top-k query.

data point p is defined as a weighted sum of the individual scores: $f_w(p) = \sum_{i=1}^{n} w[i] \times p[i]$, where $w[i] \geq 0$ ($1 \leq i \leq n$) and $\exists j$ such that $w[j] > 0$, and $\sum_{i=1}^{n} w[i] = 1$.[1]

DEFINITION 1. (Top-k query)*: Given a positive integer k and a user-defined weighting vector w, the result set $TOP_k(w)$ of the top-k query is a set of points 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 $f_w(p_i) \leq f_w(p_j)$.*

Geometrically, in the Euclidean space a linear top-k query can be represented by a vector w. Consider the dataset S depicted in Figure 1(a), and the query w=[0.75,0.25]. In the n-dimensional space, the hyperplane (line in 2d) which is perpendicular to vector w and contains a point p defines the score of point p and all points lying on the same perpendicular 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 that contains the origin of the data space. In Figure 1(a), p_2 is the top-1 object for this query, since no other data object is enclosed in the corresponding half-space.

3.2 Reverse Top-k Queries

Given a query object q, the reverse top-k query identifies all weighting vectors for which q belongs to the top-k result set. The formal definition of the reverse top-k query follows.

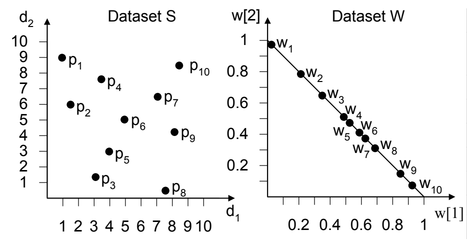
DEFINITION 2. (Reverse top-k query [9])*: Given a point q, a positive number k and two datasets S and W, where S represents data points and W is a dataset of weighting vectors, a weighting vector $w_i \in W$ belongs to the reverse top-k ($RTOP_k(q)$) result set of q, if and only if $\exists p \in TOP_k(w_i)$ such that $f_{w_i}(q) \leq f_{w_i}(p)$.*

In Figure 1(b), a dataset S is depicted together with two different weighting vectors w_1 and w_2. Assume that a reverse top-3 query is posed, while the query object is p_4. Then, w_1 belongs to the reverse top-3 query result set, since only 2 objects are contained in the underlined half-space by w_1. However, w_2 does not belong to the reverse top-3 query result set, since there exist 3 objects in the underlined half-space by w_2.

4. PROBLEM STATEMENT

We commence by defining the *influence score* of a data object p. Given a dataset and a set of weighting vectors, the definition of influence score only requires setting a single value k that determines the scope of the reverse top-k queries that are taken into account for identifying influential data objects.

[1]Since the weights represent the *relative* importance between different dimensions, the assumption $\sum_{i=1}^{n} w[i] = 1$ does not influence the definition of top-k queries.



	p_i	$RTOP_2(p)$	$f_I^k()$
1	p_3	$\{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$	8
2	p_2	$\{w_6, w_7, w_8, w_9, w_{10}\}$	5
3	p_8	$\{w_1, w_2, w_3\}$	3
4	p_5	$\{w_4, w_5\}$	2
5	p_1	$\{w_9, w_{10}\}$	2
6	p_4	{ }	0
		...	
10	p_{10}	{ }	0

Figure 2: Most influential data objects.

DEFINITION 3. (Influence Score)*: Given a positive integer k, a dataset S and a set of preferences (weighting vectors) W the influence score $f_I^k(p)$ of a data object p is defined as the cardinality of the reverse top-k query result set at point p: $f_I^k(p) = |RTOP_k(p)|$.*

Based on the definition of influence score, we provide next the formal problem statement.

DEFINITION 4. (Top-m Most Influential Data Objects)*: Given a positive integer k, a dataset S and a set of preferences (weighting vectors) W, the result set $ITOP_k^m$ of the top-m influential query is a ranked set of objects such that $ITOP_k^m \subseteq S$, $|ITOP_k^m| = m$ and $\forall p_i, p_j : p_i \in ITOP_k^m$, $p_j \in S - ITOP_k^m$ it holds that $f_I^k(p_i) \geq f_I^k(p_j)$.*

Consider the example of Figure 2, where a set S of data objects and a set of weighting vectors W are depicted. Assume that we are interested in computing the top-m most influential objects $ITOP_k^m$ for $m = 2$ and $k = 2$. In Figure 2, the result set of the reverse top-2 query for each data object p_i is shown. If the objects p_i are ranked according to the cardinality of the reverse top-2 result set ($RTOP_2(p)$), it is clear that the 2 most influential data objects are p_3 and then p_2. A delicate situation arises when two (or more) objects share the same score for the m-th position. Similar to the definition of top-k queries, it suffices to report any one of them as m-th result.

In the sequel, we present a set of useful theoretical properties related to the problem of identifying the most influential data objects that facilitate the study of the problem and provide some intuition. All proofs can be found in the Appendix.

First, for clarity reasons, we report the notions of object *dominance* and *skyline set* [1] and then, we present some important properties of the influence score. A point p dominates q ($p \prec q$), if $\forall d_i \in D$: $p[i] \leq q[i]$, and on at least one dimension $d_j \in D$: $p[j] < q[j]$. The set of points $p_i \in S$ that are not dominated by any other point $p_j \in S$ belong to the skyline set $SKY(S)$ of S.

LEMMA 1. *If a point p dominates another point q ($p \prec q$), then for any value of k, the reverse top-k result set of p is a superset of the reverse top-k result set of q:*

$$RTOP_k(p) \supseteq RTOP_k(q)$$

COROLLARY 1. *If a point p dominates another point q, then for any value of k, the influence score of p is at least as high as the influence score of q:*

$$f_I^k(p) \geq f_I^k(q)$$

We can now provide the following lemma that discloses the relationship between the most influential data object and the objects belonging to the skyline set.

Algorithm 1 Skyband-based algorithm SB

```
1:  Input: $S, W, m, k$
2:  Output: $RES$
3:  $Q \leftarrow \emptyset$
4:  $mSB(S) \leftarrow$ computeSkyband($S, m$)
5:  for all ($p_i \in mSB(S)$) do
6:      $f_I^k(p_i) \leftarrow$ RTA($S, W, p_i, k$)
7:      $Q$.push($p_i$)
8:  end for
9:  for ($j = 1 \ldots m$) do
10:     $RES \leftarrow RES \cup Q$.pop()
11: end for
12: return $RES$
```

LEMMA 2. *Given a set S of data objects, the skyline set $SKY(S)$ always contains the most influential data object p: $\exists p \in SKY(S)$, such that $p \in ITOP_k^1$.*

In our example (Figure 2), the most influential object is p_3 that belongs to the skyline set of S ($SKY(S) = \{p_1, p_2, p_3, p_8\}$).

5. SKYBAND-BASED ALGORITHM

A straightforward approach to identify the most influential data objects is to issue a reverse top-k query for each data object $p \in S$ in order to compute its influence score $f_I^k(p)$, and then rank the objects. This is clearly not efficient in practice, as it requires computing a reverse top-k query for each object in the database. Notice that processing of reverse top-k queries is costly, since each query requires several top-k query evaluations. To address this shortcoming, we propose a more efficient algorithm that exploits the notion of *skyband set*.

So far, we have identified the relationship that exists between the most influential data object and the objects belonging to the skyline set. Generalizing this observation for the m most influential data objects requires an extended set of objects that contains the skyline set, namely the *m-skyband set* [7].

DEFINITION 5. *Given a dataset S, the m-skyband set $mSB(S)$ contains all data objects that are dominated by at most $m - 1$ other objects.*

Lemma 2 can be generalized for top-m most influential objects by using the m-skyband concept.

COROLLARY 2. *Given a dataset S, the m-skyband of S always contains the top-m most influential objects p_i ($i = 1 \ldots m$), i.e., $p_i \in mSB(S)$.*

Algorithm. Capitalizing on these observations, we propose Algorithm 1 (SB) that produces the most influential data objects of a dataset S. SB first computes the m-skyband ($mSB(S)$) of S (line 4) and then processes a reverse top-k query[2] (line 6) for each object $p_i \in mSB(S)$. In this way, the influence score of all objects p_i is computed, and the objects are kept in a priority queue Q (line 7), which is sorted by influence score. Finally, the m most influential objects are retrieved from Q (line 10) and they can be returned to the user. Notice that any algorithm for skyband computation can be employed for implementing the function *computeSkyband()*. In our implementation, we employ the algorithm proposed in [7] that uses an R-tree index on S.

The performance of the SB algorithm is sensitive to the cardinality of the skyband set. Especially for high values of m, the

[2]A reverse top-k query is processed using the RTA algorithm proposed in [9].

size of the skyband set increases significantly and incurs a high number of reverse top-k queries. Moreover, SB is not incremental, as it needs to process a reverse top-k query for all objects in $mSB(S)$ before reporting any result. In addition, SB requires as input the value m, since the algorithm encapsulates the computation of $mSB(S)$. Therefore, SB is restricted as it cannot compute the m+1 most influential data object on demand, without computing the $(m$+1$)SB(S)$. In the following, we propose an incremental algorithm that does not require processing all objects in the skyband set, thus alleviating the shortcomings of SB.

6. BRANCH-AND-BOUND ALGORITHM

In this section, we present the incremental branch-and-bound algorithm (BB) for identifying the most influential data objects. BB owes its efficiency to two features: upper bounds on influence score and result sharing. First, BB calculates an *upper bound* for the influence score of candidate objects based on already processed objects. This allows BB to prioritize processing of promising objects and at the same time to postpone processing of objects that are not likely to be the next most influential object. In turn, the number of reverse top-k evaluations is minimized. Furthermore, the score bound enables BB to return the most influential objects incrementally. This occurs when the score of the current object is higher than the upper bound of all candidate objects.

Secondly, the set of candidate weighting vectors for a reverse top-k query is taken as input by the reverse top-k algorithm and affects the overall performance of computing the most influential objects. BB manages to restrict the size of the input of each reverse top-k query, by using previously computed results (sets of weighting vectors) of data objects. Therefore, BB employs *result sharing* to avoid costly re-computations and boost the performance of influence score computation.

6.1 Upper Bound of Influence Scores

We provide a bound for the influence score of an object p, by using the notion of the *dynamic skyline set* [7]. A point p_i *dynamically dominates* p'_i based on point q, if $\forall d_j \in D$: $|p_i[j] - q[j]| \leq |p'_i[j] - q[j]|$, and on at least one dimension $d_j \in D$: $|p_i[j] - q[j]| < |p'_i[j] - q[j]|$. The set of points that are not dynamically dominated based on q form the dynamic skyline set of q.

DEFINITION 6. *(Constrained Dynamic Skyline Set): Given a set of points P and a point q, we denote as $P_c \subseteq P$ the set of all points p_i, such that $\forall d_j \in D$: $q[j] \geq p_i[j]$. A point $p_i \in P_c$ belongs to the constrained dynamic skyline set $CDS(q)$ of point q, if it is not dynamically dominated with respect to q by any other point p'_i in P_c.*

$CDS(q)$ is equivalent to applying a dynamic skyline query on the objects enclosed in the window query defined by q and the origin of the data space. Figure 3 depicts an example for the constrained dynamic skyline set of point q, given the set of points p_i. The window query is defined by point q, as shown in the figure. From the set of points in P_c (p_1,p_2,...,p_5), $CDS(q)$ contains only p_3,p_4,p_5.

Upper Bound on Influence Score. Using the notion of $CDS(q)$, we define the upper bound on influence score as follows.

THEOREM 1. *Given a point q and a non-empty set of constrained dynamic skyline points $CDS(q) = \{p_i\}$, it holds that:*

$$RTOP_k(q) \subseteq \bigcap_{\forall p_i \in CDS(q)} RTOP_k(p_i)$$

DEFINITION 7. *(Upper Bound): Given a point q and a non-empty set of constrained dynamic skyline points $CDS(q) = \{p_i\}$, the upper bound $U_I(q)$ of q's influence score ($f_I^k(q) \leq U_I(q)$) is:*

$$U_I(q) = |\bigcap_{\forall p_i \in CDS(q)} RTOP_k(p_i)|$$

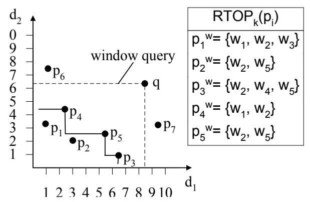


Figure 3: Upper bound of influence score of q.

Henceforth, we use p^w (i.e., the exponent w) to refer to the set of candidate weighting vectors for the reverse top-k result set of a point p. For example, in Figure 3, $CDS(q)$ consists of objects $\{p_3, p_4, p_5\}$. Therefore, the list q^w is computed as the intersection of lists p_i^w for $i =$ 3-5 and only contains the weighting vector w_2. The upper bound of the score of q is set equal to the cardinality $|q^w| = 1$.

Theorem 1 enables the calculation of an upper bound of the influence score of a point q based on the reverse top-k result sets of other points that dominate q and have been already evaluated. The following theorem shows an important property of the proposed upper bound, namely that it suffices to examine the points in $p_i \in CDS(q)$ and ignore all other points, since no point $p \in P - CDS(q)$ can further improve the upper bound.

THEOREM 2. *A point $p \in P - CDS(q)$ can not refine the upper bound $U_I(q)$.*

Based on Theorem 2, $CDS(q)$ is the minimum set of points that need to be examined in order to derive the upper bound $U_I(q)$. For example in Figure 3, point $p_1 \in P_c - CDS(q)$ dominates point $p_4 \in CDS(q)$ and the intersection of the lists $p_1^w \bigcap p_4^w$ is equal to p_4^w. Thus, including p_1 in the computation of $U_I(q)$ does not further reduce the upper bound but increases the computation overhead for determining the intersection.

Algorithm 2 computeBound()

```
1:  Input: point $q$
2:  Output: $U_I(q)$
3:  compute $CDS(q)$
4:  if ($CDS(q) \neq \emptyset$) then
5:      $q^w \leftarrow \bigcap_{\forall p_i \in CDS(q)} RTOP_k(p_i)$
6:      $U_I(q) \leftarrow |q^w|$
7:  else
8:      $U_I(q) \leftarrow |W|$
9:  end if
10: return $q$
```

Algorithm. Computing the upper bound of the influence score of a point q is performed using the pseudocode in Algorithm 2. Initially, the constrained dynamic skyline points $CDS(q)$ of q are retrieved (line 3). In case this set is empty, we cannot compute a tighter upper bound and the bound of the score is the number of all weighting vectors in W (line 8). However, when the set is non-empty (line 4), then the intersection of the reverse top-k result sets of points in $CDS(q)$ provides the list q^w of candidate weighting vectors of q (line 5) and defines the upper bound of the influence score of q (line 6).

The remaining challenge is to efficiently implement the upper bound computation of the influence score of a point. This is equivalent to finding the $CDS(q)$ of a point q given all points p_i for which the $RTOP_k$ has already been computed. For this purpose, we insert all those points p_i in a main-memory R-tree. Additionally, their computed lists of weighting vectors are also maintained.

Algorithm 3 Branch-and-bound algorithm BB

```
1:  Input: $S, W, m, k$
2:  Output: $RES$
3:  Initialize main-memory R-tree $R$
4:  $Q \leftarrow$ load MBRs in root of R-tree on $S$
5:  while ($RES$.size $< m$) do
6:      $c \leftarrow Q$.pop()
7:      if ($c$ is data object) then
8:          if ($f_I^k(c)$ is computed) then
9:              $RES \leftarrow RES \cup c$
10:         else
11:             $U_I(c.q) \leftarrow$ computeBound($c.q$)
12:             if ($U_I(c.q)$ is not improved) then
13:                 $f_I^k(c) \leftarrow$ RTA($S, c^w, c, k$)
14:                 $R$.insert($c$)
15:             end if
16:             $Q$.push($c$)
17:         end if
18:     else
19:         $U_I(c.l) \leftarrow$ computeBound($c.l$)
20:         if ($U_I(c.l)$ is not improved) then
21:             $c_i \leftarrow$ expand($c$)
22:             for all ($c_i$) do
23:                 $U_I(c_i.l) \leftarrow$ computeBound($c_i.l$)
24:                 $Q$.push($c_i$)
25:             end for
26:         else
27:             $Q$.push($c$)
28:         end if
29:     end if
30: end while
```

Then, $CDS(q)$ can be efficiently retrieved by using an adaptation of BBS [7] on the main-memory R-tree.

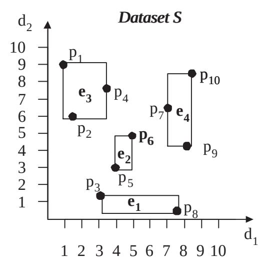
6.2 BB Algorithm

Assume that dataset S is indexed by a multidimensional indexing structure, such as an R-tree. A multidimensional bounding rectangle (MBR) e_i in the R-tree is represented by its lower left corner l_i and its upper right corner u_i. We define the *influence score of an MBR* e_i as the influence score of its lower left corner $f_I^k(e_i) = f_I^k(l_i)$.

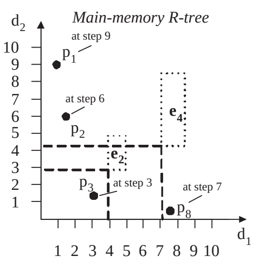
COROLLARY 3. *The influence score $f_I^k(e_i)$ of an MBR e_i is always an upper bound of all enclosed data objects.*

Intuitively, BB accesses MBRs based their influence score and places them in a priority queue Q, sorted by descending influence score. In the case of equal influence scores, the queue is sorted by ascending distance of the MBR's lower corner (or data object) to the origin of the data space. BB expands MBRs that are found at the top of Q and inserts their children MBRs in Q.

The pseudocode describing BB is presented in Algorithm 3. Initially, the root of the R-tree that indexes S is accessed (line 4). Each root entry (MBR) is inserted in Q with estimated influence score equal to the number of weighting vectors $|W|$. In each repetition, the most promising entry c is accessed from the top of Q (line 6). We use the following notation for an entry c of Q: $f_I^k(c)$ denotes its influence score, $U_I(c)$ is its upper bound, and c^w is the set of candidate weighting vectors for the result of the reverse top-k query. If c is a data object and the influence score is already computed (i.e., it corresponds to a real score, rather than an upper bound of the score), then c is safely returned as the next most influential object (line 9). Otherwise (if c is a data object), the upper bound of c's score is re-computed (function *computeBound()*) (line 11), in order to derive a tighter bound, due to recently computed results of other data objects. If a tighter bound cannot be achieved, then c's



(a) Dataset S



(b) Main-memory R-tree

Step	Action	Priority queue (Q)
1	insert root	$e_1(10), e_2(10), e_3(10), e_4(10)$
2	expand e_1	$p_3(10), e_2(10), e_3(10), p_8(10), e_4(10)$
3	calculation of p_3	$e_2(10), e_3(10), p_8(10), e_4(10), p_3(8)$
4	re-estimation of e_2	$e_3(10), p_8(10), e_4(10), p_3(8), e_2(8)$
5	expand e_3	$p_2(10), p_8(10), e_4(10), p_1(10), p_3(8), e_2(8), p_4(8)$
6	calculation of p_2	$p_8(10), e_4(10), p_1(10), p_3(8), e_2(8), p_4(8), p_2(5)$
7	calculation of p_8	$e_4(10), p_1(10), p_3(8), e_2(8), p_4(8), p_2(5), p_8(3)$
8	re-estimation of e_4	$p_1(10), p_3(8), e_2(8), e_4(8), p_4(8), p_2(5), p_8(3)$
9	calculation of p_1	$p_3(8), e_2(8), e_4(8), p_4(8), p_2(5), p_8(3), p_1(2)$
10	add p_3 to the result	$e_2(8), e_4(8), p_4(8), p_2(5), p_8(3), p_1(2)$

(c) Contents of priority queue Q

Figure 4: Example of execution of algorithm BB.

real influence score is computed by invoking the RTA algorithm for reverse top-k computation (line 13) and c is inserted in the main-memory R-tree (line 14), in order to facilitate more accurate score estimation of other entries. Then, c is pushed again in Q (line 16), either with a real score or with a tighter score bound. In case c is an MBR (line 18), its upper score bound is computed (line 19), and if it has not changed, then it is expanded and for its children MBRs c_i their upper bound scores are computed (line 23) and they are added to the queue (line 24). If a tighter bound on c's score is derived, c is added again to the queue (line 27).

The algorithm continues in the same spirit, as long as either: 1) the user requests the next most influential object, or 2) the requested m most influential objects have not been returned to the user.

Notice that there exists a special case when the influence score of q is equal to zero and there is obviously no need to compute $U_I(q)$. This occurs when more than k-1 objects p_i exist with strictly better values, i.e., $\forall j : p_i[j] < q[j]$, because it can be proved that for any weighting vector w (any top-k query) q does not belong to the top-k result set. This test is performed during the bound computation, but is omitted in the pseudocode for sake of simplicity.

Example. Consider a 2-dimensional dataset S that consists of objects $p_1, p_2, \ldots, p_{10}$. The dataset is indexed by an R-tree and its MBRs e_i are depicted in Figure 4(a). The scores of the objects p_j are shown in Figure 2. BB starts by loading in the queue Q the root of the R-tree that consists of MBRs e_1, e_2, e_3, e_4. The elements of the queue are sorted in descending order based on their score, as illustrated in Figure 4(c) (in case of ties, sorting is based on ascending distance from the origin). Then, the first element (MBR) e_1 of the queue is expanded and its enclosed objects (p_3 and p_8) are added to Q. In the next step, the score of p_3 that is equal to 8 is computed, p_3 is inserted in the main-memory R-tree R (shown at Figure 4(b)), and p_3 is re-inserted in Q. Then, the next element e_2 is accessed, and its score is bound to 8 using R, so e_2 is re-inserted in Q. As we cannot provide a tighter bound for the score of the next element e_3, it is expanded, and its enclosed objects (p_1, p_2, p_4) are inserted in Q. Notice that p_4's score can be bound using R to 8. In the next two steps, the scores of p_2 and p_8 are computed respectively and they are inserted in R and re-inserted in Q. Then, a tighter bound of the score of e_4 can be provided, using R. The next element is p_1, its score that is equal to 2 is computed and it is inserted in R and re-inserted in Q. At the following step, p_3 is found at the top of Q, and since its real score has been computed, p_3 is returned to the user as the most influential object.

Analysis of BB algorithm. The following theorems prove the correctness and the efficiency of the BB algorithm.

THEOREM 3. *(Correctness) Algorithm BB always returns the correct result set.*

PROOF. Let p_i be the data object that is returned as the next most influential object with score $f_I^k(p_i)$. The proof is by contradiction. Assume that there exists an object p_j with higher influence score $f_I^k(p_j) > f_I^k(p_i)$ that has not been returned by BB yet. This means that either p_j or an MBR that encloses p_j is located in the priority queue Q. We focus on the case that p_j is located in Q, as the other case follows the same reasoning. Then, based on the sorting of Q, either: (a) p_j has smaller influence score than p_i ($f_I^k(p_j) < f_I^k(p_i)$), or (b) p_j has equal influence score to p_i ($f_I^k(p_j) = f_I^k(p_i)$) and higher distance from the origin of the space. In both cases $f_I^k(p_j) \leq f_I^k(p_i)$, which leads to a contradiction. For the remaining case that an MBR $e(l, u)$ that encloses p_j is located in Q, we have based on the same reasoning that $f_I^k(e) \leq f_I^k(p_i)$. Moreover, from Corollary 3, we know that $f_I^k(p_j) \leq f_I^k(e)$, thus we derive that $f_I^k(p_j) \leq f_I^k(p_i)$, which is again a contradiction. □

Furthermore, BB minimizes the number of reverse top-k evaluations, since it evaluates queries only if the upper bound cannot be improved by any point of the dataset. Thus, the reverse top-k evaluation cannot be avoided.

THEOREM 4. *(Minimum number of $RTOP_k$ evaluations) BB minimizes the number of $RTOP_k$ evaluations.*

PROOF. Let p_i denote the element at the top of the queue Q that will be evaluated next. Let p_j denote any other element in Q. Two situations may occur: (a) p_j has smaller influence score than p_i ($f_I^k(p_j) < f_I^k(p_i)$), or (b) p_j has equal influence score to p_i ($f_I^k(p_j) = f_I^k(p_i)$) and higher distance from the origin of the space. In both cases $f_I^k(p_j) \leq f_I^k(p_i)$. Based on Corollary 1, p_j cannot dominate p_i. This means that no element in Q can refine the upper bound of p_i, thus the reverse top-k query for p_i cannot be avoided. □

7. INFLUENCE SCORE VARIANTS

In this section, we introduce variants of the top-m influential queries, which are meaningful for many applications and are supported by the BB algorithm.

Threshold-based influential objects. In several applications, users are interested in retrieval of objects with influence score higher than a user-specified threshold τ. For example, a product may be considered as important if the result set of its reverse top-k query includes 10% of all customers (i.e., 10%$|W|$). We refer to this query, as τ-influential query. As an analogy, the top-m influential query corresponds to a top-k query on the influence score, while the τ-influential query corresponds to a range query on the influence score.

DEFINITION 8. (τ-Influential Data Objects)*: Given a threshold τ, a dataset S and a set of preferences (weighting vectors) W, the result set τ-$ITOP_k$ of the τ-influential query is a ranked set of objects such that τ-$ITOP_k \subseteq S$, $\forall p_i : p_i \in \tau$-$ITOP_k$, it holds that $f_I^k(p_i) \geq \tau$.*

SB cannot support τ-influential queries, since the number of returned objects (m) is necessary as an input for computing the skyband set. In contrast, BB is easily adapted to support τ-influential queries. The only necessary modification is related to the termination condition; BB continues until the next retrieved data object has an influence score higher than τ (line 5). Additionally, the size of the queue can be reduced by adding to Q only objects p with upper bound $U_I(p)$ of influence score higher than or equal to τ.

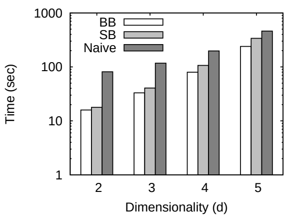
Profit-aware influence score. Assuming that each weighting vector w_i corresponds to a customer, the influence score corresponds to the number of customers that are interested in a product based on the top-k results set of their queries. Often, each customer is associated with a profit pr_i that expresses the average/expected profit of this customer, for example the number of orders or the total amount of cost of the orders. The definition of the influence score can be extended to include also the notion of the profit related to the customer.

DEFINITION 9. (Profit-aware Influence Score)*: Given a positive integer k, a dataset S and a set of preferences (weighting vectors) W, the profit-aware influence score $\hat{f}_I^k(p)$ of a data object p is defined as: $\hat{f}_I^k(p) = \sum_{\forall w_i \in RTOP_k(p)} pr_i$, where pr_i the profit that is associated to w_i.*

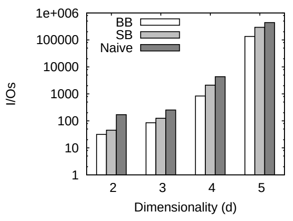
The profit-aware influence score can be used for supporting both top-m influential queries and τ-influential queries. SB and BB can support profit-aware influence queries by adjusting only the score calculation. In more details, BB needs to additionally maintain the profit that is associated with each weighting vector, and consider this value during the upper bound calculation. Then, the algorithm estimates an upper bound also for the profit-aware influence score and is able to return the correct result set.

8. EXPERIMENTAL EVALUATION

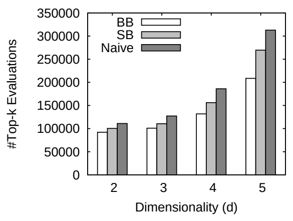
In this section, we present the results of the experimental evaluation of identifying the most influential objects. All algorithms SB, BB and Naive (which evaluates one reverse top-k query for each object in the database) are implemented in Java and the experiments run on a 3GHz Dual Core AMD processor equipped with 2GB RAM. The R-tree uses a buffer size of 100 blocks and the block size is 4KB. For the dataset S we employ both real and synthetic data collections, namely uniform (UN), correlated (CO) and anticorrelated (AC), are used. For the uniform dataset, all attribute values are generated independently using a uniform distribution. The correlated and anticorrelated datasets are generated as described in [1]. For the dataset W of the weighting vectors, two different data distributions are examined, namely uniform (UN) and clustered (CL). The clustered dataset W is generated as described in [9] and models the case that many users share similar preferences. In addition, we use two real datasets. NBA consists of 17265 5-dimensional tuples, representing a player's performance per year. The attributes are average values of: number of points scored, rebounds, assists, steals and blocks. HOUSE (Household) consists of 127930 6-dimensional tuples, representing the percentage of an American family's annual income spent on 6 types of expenditure: gas, electricity, water, heating, insurance, and property tax. We conduct a thorough sensitivity analysis varying the dimensionality (2-5d), the cardinality (10K-50K) of the dataset S, the cardinality (5K-20K) of the dataset W the value of k (10-100) and the value of m (5-20). Unless explicitly mentioned, we use the default setup of: $|S| = 10K$, $|W| = 10K$, d=3, k=10, m=10, and uniform distribution for S and W. Our metrics include: a) the



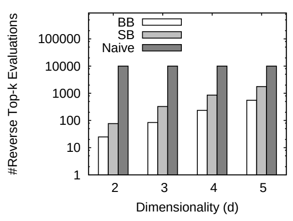
(a) Time.



(b) I/Os.



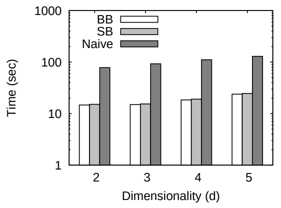
(c) Top-k evaluations.



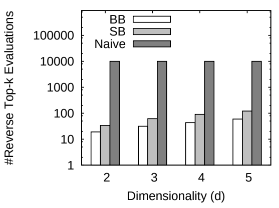
(d) Rev. top-k evaluations.

Figure 5: Comparative performance of all algorithms for UN dataset and varying dimensionality (d).

total execution time, b) the number of I/Os, c) the number of top-k evaluations, and d) the number of reverse top-k evaluations. Notice that one reverse top-k evaluation translates to multiple top-k evaluations. The experimental results with real data as well as some additional results using synthetic data can be found in Appendix B.

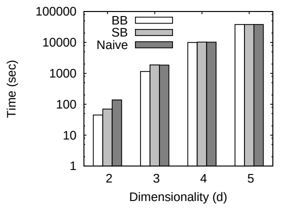


(a) Time.

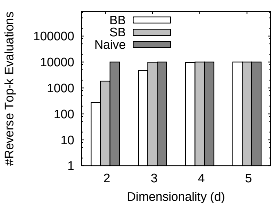


(b) Rev. top-k evaluations.

Figure 6: All algorithms for CO dataset and varying d.



(a) Time.



(b) Rev. top-k evaluations.

Figure 7: All algorithms for AC dataset and varying d.

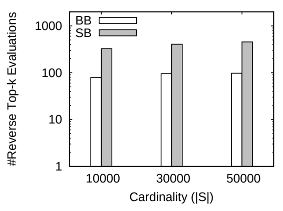
Varying dimensionality d. Figure 5 presents the comparative performance of all algorithms for uniform data distributions of S and W. BB is faster in terms of total time (Figure 5(a)) and more efficient in terms of I/Os (Figure 5(b)), usually almost by one order of magnitude. Furthermore, BB consistently requires fewer top-k evaluations than both SB and naive (Figure 5(c)). However, when the number of reverse top-k evaluations is considered (Figure 5(d)), BB attains the highest gain; BB outperforms SB by one order of magnitude and naive by 1-3 orders of magnitude. The reason that this gain is not directly reflected to metrics such as total time or I/Os is because the performance of the state-of-the-art algorithm employed for reverse top-k evaluation (RTA) is affected by the cardinality of the result set. Thus, higher ranked objects (i.e., the influential objects) have bigger result sets. As a result, the number of top-k evaluations is high and this affects the total time and the I/Os. Clearly, if we could replace RTA with a more efficient algorithm, BB would achieve even better total time and I/Os.

Our conclusions remain the same in the case of CO dataset (depicted in Figure 6), only the absolute values are smaller. In Figure 7, we show the results obtained from AC data distribution, which is the most demanding dataset for the problem of identifying the most influential objects. This is due to the fact that the number of candidate objects for inclusion in the final result set explodes, especially for increased dimensionality. BB consistently outperforms all algorithms, though the gain is smaller as the dimensionality increases. Notice that for $d > 3$, SB degenerates to the naive algorithm, because the cardinality of the skyband becomes comparable to $|S|$. This also affects the bounding of BB, since only few dominance relationships exist with increased dimensionality.

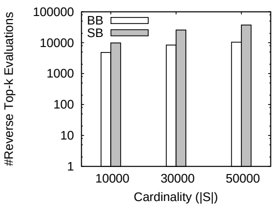
Summarizing, our algorithms outperform the naive approach in all metrics consistently, often by more than one order of magnitude, while BB is more efficient than SB. Due to this reason, we do not show the naive approach in the remaining experiments. Henceforth, for the synthetic datasets, we use as metrics the I/Os and the number of reverse top-k evaluations, as they reflect the cost and the pruning capability respectively of our algorithms. Moreover, we show only the experiments with UN and AC data distributions, as they are more demanding than the CO data distribution and the results with CO do not offer any additional insight.

Varying cardinality of S. Figure 8 shows the effect of increased values of $|S|$ to the number of reverse top-k evaluations induced by our algorithms for UN and AC datasets. As shown in the charts, BB prunes more data objects without processing, due to the upper score bounds employed, thus achieving significant performance gains for both UN and AC. More importantly, BB maintains its advantage as $|S|$ increases. Also notice that in general the number of induced reverse top-k evaluations increases moderately with $|S|$ for each data distribution.

Clustered dataset W. In Figure 9, we use a clustered dataset W and vary the dimensionality for UN and AC data distributions for S. We observe that the number of I/Os is smaller in Figure 9(a), when compared to the uniform dataset W in Figure 5(b). This shows that our algorithms perform better in the case of clustered user preferences W, which are more realistic in real applications. In general, BB always outperforms SB regardless of the dimensionality value, even in the case of the demanding AC data distribution, although the gain is admittedly smaller.



(a) UN dataset.



(b) AC dataset.

Figure 8: BB vs. SB for varying $|S|$.

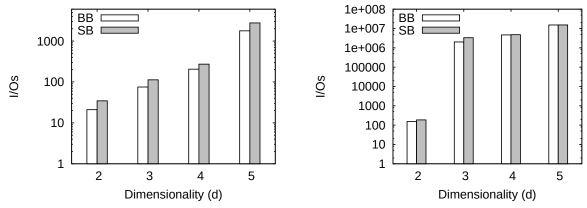

(a) UN dataset. (b) AC dataset.

Figure 9: BB vs. SB for clustered W.

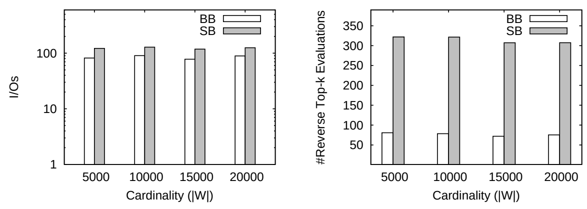

(a) I/Os. (b) Rev. top-k evaluations.

Figure 10: BB vs. SB for UN dataset and varying $|W|$.

Varying cardinality of W. We also measure the number of I/Os and reverse top-k evaluations when increasing $|W|$ in Figure 10. Although the number of top-k evaluations increases with $|W|$ (figure omitted due to lack of space), this is not reflected in the induced I/Os, which remain relatively stable. More importantly, the gain of BB in terms of pruning is sustained as $|W|$ increases, as shown in Figure 10(b). Both algorithm scale well with increased values of $|W|$ and BB is better in all cases.

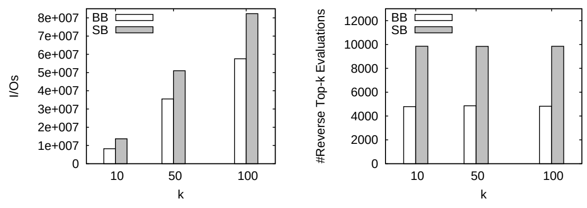

(a) I/Os. (b) Rev. top-k evaluations.

Figure 11: BB vs. SB for AC dataset and varying k.

Varying k. In Figure 11, we investigate the performance of our algorithms for increasing values of k for the AC dataset. Intuitively, when k increases, more objects are candidates for inclusion in the top-m influential objects, since more objects appear in the result sets of top-k queries. Therefore, it is expected that the cost of query processing will increase. This is verified by our experiments in terms of necessary I/Os, as shown in Figure 11(a). However, the number of reverse top-k evaluations is not affected significantly, as depicted in Figure 11(b). In all cases, BB's comparative advantage over SB is sustained.

Varying m. In Figure 12, we demonstrate the effect of m in the performance of our algorithms. BB is more stable than SB as m increases. As the m-skyband size increases with m, SB needs to evaluate more reverse top-k queries, thus its cost increases. In contrast, BB scales gracefully with m, because as m increases a higher number of objects have already been evaluated, resulting in refined upper bounds and improved performance of query processing for the remaining objects. This is an important finding that

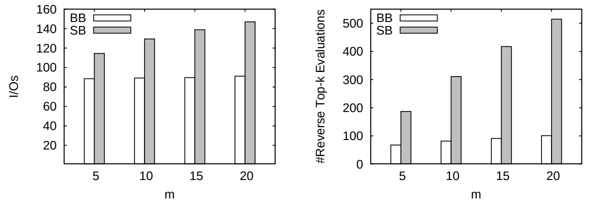

(a) I/Os. (b) Rev. top-k evaluations.

Figure 12: BB vs. SB for UN dataset and varying m.

demonstrates the benefit of BB with increasing values of m.

9. CONCLUSIONS

Given a product, the reverse top-k query returns a set of customers that find the product appealing, i.e., it belongs to their top-k results. In this paper, we address the important data analysis task of identifying the most *influential products* to customers, given a database of products. The influence of a product is defined as the cardinality of the reverse top-k result set. By exploiting the properties of reverse top-k queries, we propose two algorithms that find efficiently the most influential objects. Our first algorithm SB restricts the candidate set of objects that need to be examined based on the skyband set of the data objects. The second algorithm BB is a branch-and-bound algorithm that retrieves the result incrementally. In addition, we study meaningful variations of the query for most influential objects that are useful in practice and they can be supported by our algorithms.

10. 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] E. Dellis and B. Seeger. Efficient computation of reverse skyline queries. In *Proc. of VLDB*, pages 291–302, 2007.
[4] F. Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *Proc. of SIGMOD*, pages 201–212, 2000.
[5] C. Li, B. C. Ooi, A. K. H. Tung, and S. Wang. Dada: a data cube for dominant relationship analysis. In *Proc. of SIGMOD*, pages 659–670, 2006.
[6] M. Miah, G. Das, V. Hristidis, and H. Mannila. Standing out in a crowd: Selecting attributes for maximum visibility. In *Proc. of ICDE*, pages 356–365, 2008.
[7] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM Transactions on Database Systems*, 30(1):41–82, 2005.
[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] A. Vlachou, C. Doulkeridis, Y. Kotidis, and K. Nørvåg. Reverse top-k queries. In *Proc. of ICDE*, 2010.
[10] Q. Wan, R. C.-W. Wong, I. F. Ilyas, M. T. Özsu, and Y. Peng. Creating competitive products. *PVLDB*, 2(1):898–909, 2009.
[11] R. C.-W. Wong, M. T. Özsu, P. S. Yu, A. W.-C. Fu, and L. Liu. Efficient method for maximizing bichromatic reverse nearest neighbor. *PVLDB*, 2(1):1126–1137, 2009.
[12] T. Wu, Y. Sun, C. Li, and J. Han. Region-based online promotion analysis. In *Proc. of EDBT*, 2010.
[13] T. Wu, D. Xin, Q. Mei, and J. Han. Promotion analysis in multi-dimensional space. In *Proc. of VLDB*, 2009.
[14] T. Xia, D. Zhang, E. Kanoulas, and Y. Du. On computing top-t most influential spatial sites. In *Proc. of VLDB*, pages 946–957, 2005.

APPENDIX

A. PROOFS

Proof of Lemma 1.

PROOF. By contradiction. Assume that $RTOP_k(p) \subset RTOP_k(q)$ and therefore there exists $w \in RTOP_k(q)$ such that $w \notin RTOP_k(p)$. We conclude that: (a) $f_w(p) \leq f_w(q)$, because $p \prec q$, thus $\forall d_i$ $p[i] \leq q[i]$ and f_w is monotone, and (b) $\exists r \in TOP_k(w)$ such that $f_w(q) \leq f_w(r)$, since $w \in RTOP_k(q)$ (Definition 2). From (a) and (b) we derive that $\exists r \in TOP_k(w)$ such that $f_w(p) \leq f_w(q) \leq f_w(r)$, thus by definition $w \in RTOP_k(p)$, which is a contradiction. $\square$

Proof of Corollary 1.

PROOF. According to Lemma 1, it holds that: $RTOP_k(p) \supseteq RTOP_k(q)$. This directly implies that: $|RTOP_k(p)| \geq |RTOP_k(q)|$, and equivalently: $f_I^k(p) \geq f_I^k(q)$. $\square$

Proof of Lemma 2.

PROOF. By contradiction. Assume that $\nexists p \in SKY(S)$ such that $p \in ITOP_k^1$. Then $\exists q \notin SKY(S)$ and $q \in ITOP_k^1$. We conclude that: (a) $\exists p \in SKY(S)$ such that $p \prec q$ and $f_I^k(p) \geq f_I^k(q)$ (Corollary 1) (b) $\forall r \in S - ITOP_k^1$ it holds that $f_I^k(q) \geq f_I^k(r)$ since $q \in ITOP_k^1$. Thus, from (a) and (b) we derive that $\forall r \in S - ITOP_k^1$ it holds $f_I^k(p) \geq f_I^k(q) \geq f_I^k(r)$. Based on Definition 4 we conclude that $p \in ITOP_k^1$, which contradicts our assumption. $\square$

Proof of Theorem 1.

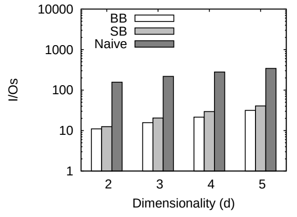
PROOF. It holds that $\forall p_i \in CDS(q)$: $p_i \prec q$ and there exists at least one $p_i \in CDS(q)$. Moreover, from Lemma 1 we derive that: $\forall p_i : RTOP_k(q) \subseteq RTOP_k(p_i)$. According to the set theory it holds that: if $A \subseteq B$ and $A \subseteq C$, then $A \subseteq B \cap C$. Consequently, it holds that $RTOP_k(q) \subseteq \bigcap_{\forall p_i \in CDS(q)} RTOP_k(p_i)$. $\square$

Proof of Theorem 2.

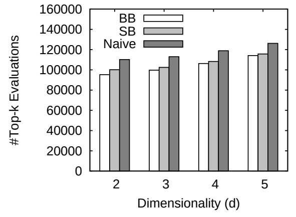
PROOF. Let us assume that there exist $p \in P - CDS(q)$ such that $\bigcap_{\forall p_i \in CDS(q) \cap \{p\}} RTOP_k(p_i) \subset \bigcap_{\forall p_i \in CDS(q)} RTOP_k(p_i)$ which leads to a smaller upper bound $U_I(q)$. If $p \notin P_c$, then $p \not\prec q$ and the Lemma 1 is violated. Thus, $p \in P_c - CDS(q)$ and based on Definition 6 $\exists p_j \in CDS(q)$ such that $p \prec p_j$. Based on Lemma 1 $RTOP_k(p_j) \subseteq RTOP_k(p)$ and therefore $\bigcap_{\forall p_i \in P} RTOP_k(p_i) \subseteq \bigcap_{\forall p_i \in CDS(q)} RTOP_k(p_i)$. Which leads to a contradiction. $\square$

Proof of Corollary 3.

PROOF. Obviously, any data object p enclosed in an MBR $e_i(l_i, u_i)$ is either equal or dominated by the lower left corner l_i. Thus, based on Corollary 1, the influence score $f_I^k(p)$ of object p is bound by the influence score of l_i: $f_I^k(p) \leq f_I^k(l_i)$. $\square$

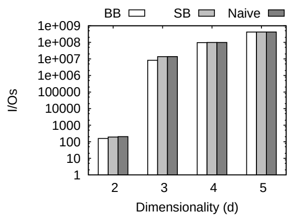


(a) I/Os.

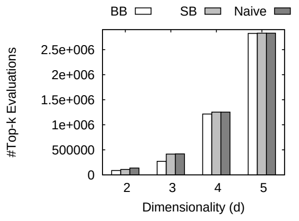


(b) Top-k evaluations.

Figure 13: All algorithms for CO dataset and varying d.

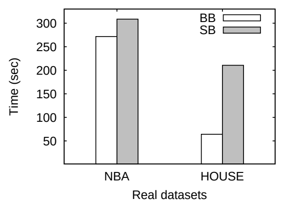


(a) I/Os.

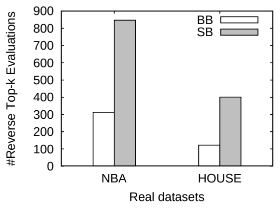


(b) Top-k evaluations.

Figure 14: All algorithms for AC dataset and varying d.



(a) Time.



(b) Rev. top-k evaluations.

Figure 15: BB vs. SB for real data.

B. ADDITIONAL EXPERIMENTAL RESULTS

In this section, we provide some additional experimental results.

Experiments with synthetic data. In Figure 13, we measure the number of I/Os and the number of required top-k evaluations for the CO dataset and for varying d. The results are complementary to those of Figure 6. Both our algorithms perform consistently better than naive. In addition, BB is more efficient than SB.

In Figure 14, we depict the same measures for the AC dataset and for varying d. These results correspond to the experimental setup of Figure 7 and they confirm the conclusions drawn for the case of the AC dataset.

Experiments with real data. In Figure 15, we show the experiments using the real datasets. BB produces the result faster than SB for both datasets. Moreover, the gain in terms of pruning is high for both datasets, as demonstrated by the number of reverse top-k evaluations. Obviously, the absolute values of the measures depend on the peculiarities of each dataset, for example the data distribution. However, we observe that the relative gain of BB compared to SB is much higher in the case of the larger dataset (HOUSE), since BB manages to prune more data points. Notice that HOUSE is the largest dataset in our experimental study in terms of cardinality and dimensionality. In general, the results with real datasets are in accordance with the conclusions drawn from the synthetic datasets.