Finding the Most Diverse Products using Preference Queries

Orestis Gkorgkas[1], Akrivi Vlachou[2], Christos Doulkeridis[3] and Kjetil Nørvåg[1]
[1]Norwegian University of Science and Technology (NTNU), Trondheim, Norway
[2]Institute for the Management of Information Systems, R.C. "Athena", Athens, Greece
[3]University of Piraeus, Piraeus, Greece
{orestis,vlachou,noervaag}@idi.ntnu.no, cdoulk@unipi.gr

ABSTRACT

In this paper, given a product database and a set of customer preferences, we address the problem of discovering a bounded set of r diverse products that attract the interests of different customers. This problem finds numerous applications in electronic marketplaces, e.g., for selecting the products that are placed in the home page of an online shop. Existing approaches to tackle this problem fall short because they ignore customer preferences, and instead rely solely on products' attributes. We model this problem as a diversity problem, where each product is represented by its reverse top-k result set, and seek r products that maximize their diversity value. Since the problem is NP-hard, we employ a greedy algorithm that takes as input the reverse top-k result sets of all candidate products. To further improve performance, we also design a more efficient approximate algorithm that does not require the computation of all reverse top-k sets. Our experimental evaluation demonstrates the performance of the proposed algorithms and quality of the selected diverse products.

1. INTRODUCTION

Top-k queries [17] help customers select a ranked set of k products that best match their preferences out of an overwhelmingly large collection of products. For a specific customer, her preferences are expressed by means of a top-k query, and highly ranked products in the top-k result are more attractive to the customer. Thus, from the perspective of product sellers, the visibility and the potential market of a product relates to the top-k queries for which the product is highly ranked. Towards this direction, reverse top-k queries [20] retrieve the set of user preferences for which a product appears in their top-k lists. Reverse top-k queries are very important for estimating the impact of the product on the market, as the cardinality of the result set defines an *influence score* [22] for the product, i.e., the number of customers that value a particular product.

In this paper, we study the problem of finding the r most diverse products based on the user preferences. The goal is to find a set of products that are attractive to a wide range of customers with different preferences. For instance, consider an electronic marketplace that wishes to advertise r products on its front page aiming to attract as many new customers as possible. Advertising diverse products that are attractive to different existing customers increases the probability that a new customer finds one of those products attractive. The strategy of advertising the r most influential products [22], i.e., the r products that attract the highest total number of customers, does not necessarily lead to a set of diverse products and may fail to attract many new customers, since such products may be attractive to customers with similar preferences.

User Preferences:

User	$w[1]$	$w[2]$	$w[3]$	**Top-k**
Bob	0.1	0.2	0.7	p_1
Tom	0.1	0.3	0.6	p_1
Jack	0.3	0.1	0.6	p_2
Max	0.8	0.1	0.1	p_3

Products:

Product	$p[1]$	$p[2]$	$p[3]$	**Reverse top-k**
p_1	1	2	6	Bob,Tom
p_2	2	1	6	Jack
p_3	6	5	2	Max

Table 1: Example of product database and user preferences.

Consider for example the set of user preferences and products depicted in Table 1, where maximum values in product attributes are preferable. Assume that the goal is to advertise two products for attracting new customers. Our proposed method selects the $r = 2$ most diverse products based on user preferences, which in our example is the set $\{p_1, p_3\}$. This set is more probable to attract more new customers because p_1 and p_3 satisfy more diverse preferences. For example, a customer with similar preferences to Jack is highly probable to be attracted also to p_1, even though it is not the best option for her on the market. This is because both p_1 and p_2 satisfy users that have high preference for the third dimension (expressed with a high weight $w[3]$). On the other hand, p_3 satisfies users that have totally diverse preferences compared to p_1 and p_2, namely users such as Max that prefer the first dimension.

In this paper, we introduce the problem of finding the r *most diverse* products based on user preferences. The

user preferences are captured by the reverse top-k set of each product. We model this problem as a *dispersion* problem [15] using as distance function the dissimilarity of the reverse top-k sets. In this sense, the set of r objects with the maximum diversity is returned to the user. Consequently, the selected objects are appealing to many different customers with dissimilar user preferences. Different from our work, existing solutions for identifying diverse objects rely solely on product attributes and largely overlook user preferences [18]. On the other hand, approaches that identify r objects with high total number of customers [12, 22], often fail to discover truly diverse products that can be appealing to new customers with different preferences than those of the existing ones.

To summarize the contributions of this paper are:

- We study the novel problem of finding the r most diverse products based on user preferences. We model this problem as a *dispersion* problem and define an appropriate distance function that captures the dissimilarity of products based on their reverse top-k sets.
- As dispersion problems are known to be NP-hard [8], we use a greedy algorithm that retrieves r diverse products, after computing the reverse top-k sets of the products efficiently.
- To improve the performance of our algorithm, we propose an alternative algorithm that progressively computes an approximation of the reverse top-k sets of a limited set of candidate products and retrieves a set of r products of high diversity.
- We present maintenance techniques for updating the r most diverse products in the case of dynamic data in a cost-efficient way. In addition, we generalize our approach to support any set-based similarity function.
- We demonstrate the efficiency and achieved diversity of our algorithms using both synthetic and real-life data sets.

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 provides the necessary preliminaries, while in Section 4 we formally define the r-Diversity problem. Thereafter, in Section 5, we present a greedy algorithm applied on the reverse top-k sets. In Section 6, we provide a more efficient algorithm that iteratively computes an approximation of the reverse top-k sets and refines the set of most diverse products. Section 7 addresses the case of dynamic data, while Section 8 generalizes our approach for set-based similarity functions. The experimental evaluation is presented in Section 9 and we conclude in Section 10.

2. RELATED WORK

Reverse top-k queries. Vlachou *et al.* first introduced the reverse top-k query in [20, 21]. Two versions of the reverse top-k query were presented, namely monochromatic and bichromatic. Based on the geometrical properties of the monochromatic reverse top-k query, an algorithm for the two dimensional case was proposed. For computing bichromatic reverse top-k queries, an algorithm (called *RTA*) was proposed that exploits the fact that similar queries share common results, in order to avoid evaluating the top-k queries for all user preferences. Thereafter, several papers have studied the problem of efficient reverse top-k computation. An efficient algorithm for the two-dimensional monochromatic reverse top-k that relies on a novel index was proposed in [4]. In [9], efficient evaluation of multiple top-k queries is studied, which in turn enables the computation of the reverse top-k set of a query point. The proposed method avoids evaluating the top-k queries one-by-one by grouping similar queries and evaluate them in a batch. This approach is suitable for processing many reverse top-k queries at once. An approach for processing a large number of continuous top-k queries has appeared in [27]. The proposed framework can be employed to process reverse top-k queries efficiently, however it requires to build an index over the k-th ranked objects of each query that results in high pre-processing cost. Recently, a novel branch-and-bound algorithm for reverse top-k queries has appeared in [23], where both the object data sets and the preferences set are indexed using an R-tree.

Product impact and visibility. Several papers have proposed methods that aim to quantify the impact of products in the market. DADA [11] aims to help manufactures position their products in the market, based on three types of dominance relationship analysis queries. Creating competitive products has been studied in [24]. Customer identification and product positioning has been recently studied in [2], where the attractiveness of a product is defined based on the concept of reverse skyline query. 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.* [14] study a different problem, namely how to select the subset of attributes that increases the visibility of a new product. Product promotion is studied in [25, 26], where the aim is to find the most interesting regions for promotion of a product. Only a few papers have proposed methods for retrieving interesting products by using the reverse top-k queries. In [22], the influence of a product is defined as the size of its reverse top-k set. Then, an algorithm was presented to efficiently retrieve the m most influential products. Discovering k products with maximum number of customers has been studied in [12], where the number of customers is estimated as the size of the reverse top-k set. The problems studied in [12, 22] differ from the diversity problem studied in this paper. Both approaches focus on maximizing the number of existing customers and ignore the similarity of the retrieved reverse top-k sets. These approaches fail to take into account the fact that attracting new customers requires promoting products that are attractive to customers with diverse preferences. Koh *et al.* [10] consider as products packages consisting of multiple components. They study the problem of creating and selecting packages from an existing pool of components such that the number of potential customers is maximized. Similarly to the aforementioned approaches the number of potential customers is estimated using reverse top-k sets, yet they do not study the diversity of the result set.

Diversity in databases. Many approaches have been proposed for retrieving a set of diverse objects. Angel *et al.* [1] study the problem of retrieving k documents relevant to a query q, but are also diverse with each other. The diversity is computed based on document similarity metrics. Drosou *et al.* [7] study the problem of finding the k most

Symbol	Description
$\mathbb{R}^m$	m-dimensional dataspace
S	Set of data objects
D	Subset of S ($D \subseteq S$)
p,q	Data objects/products ($p, q \in S$)
W	Set of weighting vectors
$\mathbf{w}$	A weighting vector ($\mathbf{w} \in W$)
$f_{\mathbf{w}}$	Preference function associated with $\mathbf{w}$
k	Value of top-k
$TOP_k(\mathbf{w})$	Top-k data objects based on $\mathbf{w}$
$RTOP_k(p)$	Reverse top-k result set for object p
$\mathbf{c}_p$	Centroid of vectors in set $RTOP_k(p)$
$d(p, q)$	Cosine distance between centroids $\mathbf{c}_p, \mathbf{c}_q$
$d(\mathbf{u}, \mathbf{v})$	Cosine distance between vectors $\mathbf{u}, \mathbf{v}$
$div(D)$	Diversity value of a set of objects D
D^*	Optimal solution of the r-Diversity problem
$D_r(S)$	Approximate solution of the r-Diversity problem

Table 2: Overview of symbols.

diverse objects in a continuous data stream. DivDB, a system that provides query result diversification, was presented in [19]. Result diversification based on dissimilarity is studied also in [6]. Estimating the diversity of a set of points that fulfill a special property has been studied mainly for selecting representative skyline points. For instance the diversity of two skyline points can be defined as the distance between them [16] or by using their sets of dominated points [13, 18]. More specifically, in [16] the authors define the set of representative skyline to be a set of k objects that maximize the minimum Euclidean distance between any two of the k points. In [13], the representative skyline points are defined based on the distinct number of dominated points. Valkanas *et al.* [18] estimate the diversity of two skyline points by calculating the Jaccard distance of their respective sets of dominated points. The main difference to our work is that the definitions of diversity in the above approaches rely on the attribute values only and cannot exploit the existing user preferences.

3. PRELIMINARIES

Given a space $\mathbb{R}^m$, we assume that we have a set of data objects S where each object $p \in S$ can be represented as an m-dimensional point $p = \{p[1], \ldots, p[m]\}$ where $p[i] \in \mathbb{R}^+$. Each point p can be regarded as an object of a database and each dimension of the point as a specific numerical attribute. Without loss of generality, we assume that larger values are preferable.

Given a scoring function $f : S \to \mathbb{R}^+$,a *top-k* query returns the k objects of S with the best score. The scoring functions used more often, are linear functions of the form $f(p) = \sum_{i=1}^{m} w[i]p[i]$ where $w[i] \geq 0$. Such functions can be represented by an m-dimensional *weighting vector* $\mathbf{w} = \{w[1], \ldots, w[m]\}$. In such cases we denote the function that results from $\mathbf{w}$ as $f_{\mathbf{w}}$. When $\mathbf{w}$ represents the preferences of a user over the objects in S we call this vector *preference vector* or simply *preference*.

DEFINITION 1. ***(Top-k query [20])*** *Given a data set $S \subseteq \mathbb{R}^m$ and a vector $\mathbf{w} \in \mathbb{R}^m$ the result set $TOP_k(\mathbf{w})$ of the top-k query is a set of points such that $TOP_k(\mathbf{w}) \subseteq S$, $|TOP_k(\mathbf{w})| = k$ and $\forall p_1, p_2 : p_1 \in TOP_k(\mathbf{w}), p_2 \in S - TOP_k(\mathbf{w})$ it holds that $f_{\mathbf{w}}(p_1) \geq f_{\mathbf{w}}(p_2)$*

If we have a set of preferences $W \subseteq \mathbb{R}^m$ over a set of products $S \subseteq \mathbb{R}^m$ then for a given product q, we say that the result set of a *reverse top-k* query is a set $RTOP_k(q)$ which consists of all the preference vectors $\mathbf{w}$ for which it holds that $q \in TOP_k(\mathbf{w})$.

DEFINITION 2. ***(Reverse top-k query [20])*** *Given a set of points S, a point p, and a set of vectors W we say that a vector $\mathbf{w}$ belongs in the reverse top-k set $RTOP_k(p)$ of point p, if and only if $\exists q \in TOP_k(\mathbf{w})$ such that $f_{\mathbf{w}}(p) \geq f_{\mathbf{w}}(q)$.*

We can also define the *influence score* of a data object p as the cardinality of the set $RTOP_k(p)$. Table 2 provides an overview of the most basic symbols used in this paper.

4. PROBLEM STATEMENT

Let p and q denote two products (data objects) from a product database S. Also, given a set W of customer preferences (weighting vectors) and an integer k, let $RTOP_k(p) \subseteq W$ and $RTOP_k(q) \subseteq W$ denote the reverse top-k sets of p and q respectively. We also define a distance function $d : S \times S \to \mathbb{R}^+$ as:

$$d(p, q) = f_d(RTOP_k(p), RTOP_k(q))$$

that determines the dissimilarity of any two objects p and q based on their corresponding reverse top-k sets. Notice that this is a radically different approach from existing initiatives that define the distance of two objects based on the objects' attributes only.

The problem of selecting the r most diverse products from a given set S can be viewed as a *dispersion* problem [7, 8, 15, 18], where the aim is to find r objects such that an objective function of their distance d is optimized. The *dispersion sum problem* maximizes the sum of pairwise distances between the r selected products and it has been proved that it is NP-hard by reduction from the clique problem [8].

PROBLEM 1. ***(r-Diversity Problem)***. *Given a set of data objects S and a distance function d measuring the dissimilarity between two objects, the r-Diversity problem is to identify a subset $D^* \subseteq S$ such that:*

$$D^* = \arg\max_{\substack{D \subseteq S \\ |D|=r}} \sum_{\substack{p,q \in D \\ p \neq q}} d(p, q)$$

The remaining challenge is to define an appropriate function f_d that captures the dissimilarity of the reverse top-k result sets. Hence, the function f_d takes as input two sets of weighting vectors and computes their dissimilarity. We employ a function that relies on the concept of a *centroid* of a set of vectors.

DEFINITION 3. ***(centroid of RTOPk)***. *Given a set of data objects S, a set of weighting vectors W, and an object $p \in S$ such that $RTOP_k(p) \neq \emptyset$, we define as the* centroid *of p the vector:*

$$\mathbf{c}_p = \frac{1}{|RTOP_k(p)|} \sum_{\mathbf{w} \in RTOP_k(p)} \mathbf{w}$$

Since each $RTOP_k$ set corresponds to exactly one data point, the respective centroid corresponds to exactly one data point as well. Therefore each data point can be mapped to exactly one centroid vector and vice-versa.

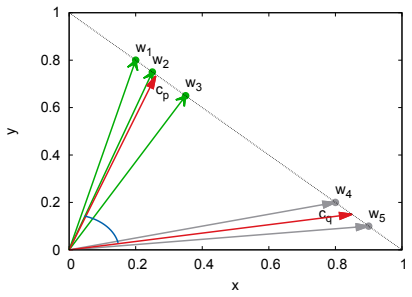


Figure 1: Example of dissimilarity function.

DEFINITION 4. ***(Dissimilarity function f_d).*** *Given a set of data objects S, a set of weighting vectors W, two objects $p, q \in S$, and their respective centroids $\mathbf{c}_p$ and $\mathbf{c}_q$, the distance of p and q is defined based on the cosine similarity of the centroids:*

$$f_d(RTOP_k(p), RTOP_k(q)) = 1 - cos(\mathbf{c}_p, \mathbf{c}_q)$$

The advantage of using the centroid $\mathbf{c}_p$ instead of the actual set of vectors $RTOP_k(p)$ is that the centroid is a compact and accurate representation of the set, which in turn allows efficient processing of the dissimilarity function, compared to other dissimilarity metrics that operate on sets of arbitrary size. As a result, we use $d(p, q) = 1 - cos(\mathbf{c}_p, \mathbf{c}_q)$ as distance function in this paper[1].

EXAMPLE 1. *Figure 1 shows an example of the reverse top-k sets $RTOP_k(p) = \{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ and $RTOP_k(q) = \{\mathbf{w}_4, \mathbf{w}_5\}$, which belong to products p and q respectively. In the Euclidean space, a linear top-k query can be represented by a vector $\mathbf{w}$ [20, 21]. The magnitude of the query vector does not influence the query result, as long as the direction remains the same, therefore without loss of generality we assume that $\sum_{i=1}^{m} w[i] = 1$. In the 2-dimensional space, all weighting vectors belong to the line as depicted in Figure 1. Moreover, top-k queries defined by similar weighting vectors $\mathbf{w}$ are expected to produce similar result sets [20, 21]. Thus, the weighting vectors of the reverse top-k set of p are expected to lie nearby on the line. Furthermore, for a hypothetical weighting vector which lies on the line between $\mathbf{w}_1$ and $\mathbf{w}_3$, it is expected that p is highly ranked, and therefore it is highly probable that this vector would belong to the reverse top-k set of p.*

The centroid of the weighting vectors captures the above intuitions, and the angle between two centroids represents the dissimilarity of the weighting vectors. Obviously, different functions for set dissimilarity (hence also for measuring distance) are supported by our approach, including (for instance) the Jaccard similarity of the reverse top-k sets. Nevertheless, the Jaccard similarity fails to capture the locality of the weighting vectors.

Furthermore, we define the *diversity* $div(D)$ of a set of objects $D \subseteq S$. Notice that the set D^* with the highest diversity value $div(D^*)$ among all r-sets of points in S, is the optimal solution for Problem 1. The diversity value $div(D)$ is normalized in [0,1].

[1]In a slight abuse of notation, we also use $d(\mathbf{u}, \mathbf{v}) = 1 - cos(\mathbf{u}, \mathbf{v})$ to denote the cosine distance between any two vectors $\mathbf{u}$ and $\mathbf{w}$.

DEFINITION 5. ***(Diversity value)*** *Given a set of points S, a subset $D \subseteq S$ of size r, and set of vectors W, we define as* diversity *of D:*

$$div(D) = \frac{2}{r(r-1)} \sum_{\substack{p,q \in D \\ p \neq q}} (1 - cos(\mathbf{c}_p, \mathbf{c}_q))$$

5. ALGORITHMS WITH CENTROID COMPUTATION

The process of discovering r diverse products $D_r(S)$ from a set of products S consists of two main steps: (1) identifying a set C of *candidate centroids* that correspond to candidate products for inclusion in the most diverse products (Section 5.1), and (2) selecting r of these candidates as the most diverse products (Section 5.2).

Each candidate centroid in $\mathbf{c}_p \in C$ corresponds to one product $p \in S$ and is the centroid vector of the $RTOP_k(p)$ set of p. More formally $C = \{\mathbf{c}_p | p \in S, RTOP_k(p) \neq \emptyset, \mathbf{c}_p$ is centroid of $RTOP_k(p)\}$. Obviously, products that are not preferable for any customer are ignored.

Algorithm 1 describes the afore-described method and returns a set of r diverse products. In line 1, the candidate centroids C are computed using any of the methods that will be described in Section 5.1. As the set of centroids C may be large depending on the data distribution, a sample R of fixed size s is created by picking centroids uniformly at random (line 2). Finally, in line 3, the second step entails solving the r-Diversity problem by applying a greedy algorithm, called Diverse Product Selection Algorithm (*DPSA*), on the sampled set of centroids R, as will be described in Section 5.2.

Algorithm 1 r-Diverse Products

Input: S: set of products
W: set of weighting vectors
k: value of top-k and reverse top-k
s : size of initial sample
r : required number of diverse products
Output: $D_r(S)$: the set of r most diverse products of S

1: $C \leftarrow$ CandidateCentroids(S, W, k)
2: $R \leftarrow$random subset of C with $|R| = s$
3: $D_r(S) \leftarrow DPSA(C, R, r)$
4: **return** $D_r(S)$

5.1 Retrieving the Candidate Centroids

Different alternatives exist in order to compute the set C of candidate centroids. In the following, we present three alternative methods for determining the set C. Notice that all methods produce an identical set C of centroids.

The most straightforward method is to perform a reverse top-k query for each product p in S and compute the centroid vector of each set $RTOP_k(p)$ using Definition 3. We denote this approach *Rtopk*. Its processing cost is basically determined by the computation of $|S|$ reverse top-k queries. Since any existing algorithm for reverse top-k processing can be employed for the underlying reverse top-k computation, this method is quite generic.

An improvement of the first method is derived based on the observation that some products have empty reverse top-k sets (i.e., they do not belong to the top-k result of any

weighting vector). Hence, it is possible to avoid processing some reverse top-k sets. To achieve this, we exploit the progressive result generation of the algorithm in [22], which is able to retrieve objects in decreasing order of the sizes of their reverse top-k sets. We denote this method as *Itopk* based on the fact that the algorithm [22] has been proposed for retrieval of influential objects. As a result, we avoid processing a reverse top-k query for objects with empty reverse top-k sets, thus improving the performance of *Rtopk*.

The third method exploits the observation that it may be more efficient to process all top-k queries, instead of processing multiple reverse top-k queries. Thus, we perform a top-k query for each preference vector $w \in W$, which makes straightforward the computation of the reverse top-k sets of any data object, and hence also their respective centroids. In fact, the top-k sets do not need to be maintained until all top-k queries have been processed, but instead the centroids can be calculated progressively. For each retrieved object in the top-k result set, the centroid is updated by adding the new vector $\mathbf{w}$ to its previous centroid, while also the number of vectors per object is maintained. After finishing all top-k queries, for each centroid the coordinates are divided by the cardinality of the reverse top-k set. Since top-k queries for all vectors in W are processed, we call this method all top-k, i.e., *Atopk*. An advantage of *Atopk* is that the processing cost in terms of top-k evaluations is fixed, namely $|W|$ top-k queries, in contrast to *Rtopk* and *Itopk* where in the worst case the top-k evaluations can be up to $|W| \cdot |S|$. Thus, the efficiency of *Atopk* is influenced slightly by the cardinality of S, in contrast to *Rtopk* which computes the reverse top-k set even for products with empty reverse top-k sets.

5.2 Diverse Product Selection Algorithm

After having computed the centroid vectors of all non-empty reverse top-k sets, the next step is to find the r most diverse centroids and the products that they represent. As already mentioned, the r-Diversity problem is defined as a dispersion problem that is known to be NP-hard [8]. Thus, computing the optimal solution for the r-Diversity problem is not feasible even for relatively small data sets. Hence, we employ an algorithm that efficiently computes an approximate solution of high quality [5]. More specifically, we use a greedy algorithm, called Diverse Product Selection Algorithm (*DPSA*), that iteratively selects the next centroid that maximizes the value of the objective function. Its pseudocode is depicted in Algorithm 2.

Description. The algorithm takes as input the set of candidate centroids C, a random sample set R of the candidate centroids that is going to be used, and an integer r which is the desired number of most diverse products. It returns an approximate set $D_r(S)$ of the r most diverse products and their centroids. The sample R is typically much smaller in size than C, in order to reduce the cost of the first part of the algorithm, which is to find the two most distant vectors in R (line 2) and add them to the result set $D_r(S)$ (line 3). Then, the algorithm iteratively selects the next centroid $\mathbf{c}_q$ until r centroids have been retrieved (loop in line 4). Each time, the selected centroid is the one that maximizes the sum of distances from the already selected most diverse vectors $D_r(S)$. Notice that R is used only for the initialization of $D_r(S)$ (line 3), while the remaing centroids are selected from C.

Complexity. The selection of the two most diverse products (seeds) has a cost $O(|R|^2) = O(s^2)$. The remaining part of the algorithm has a cost of $O(r^2|C|)$ and therefore the total cost is equal to $O(s^2 + r^2|C|)$. If no sample is used ($s = |C|$) in the seed selection then the algorithm will have a cost of $O(|C|^2)$.

Algorithm 2 Diverse Product Selection Algorithm *DPSA*()

Input: C : set of candidate centroids
R : sample of C
r : required number of diverse products
Output: $D_r(S)$: the set of r most diverse products of S

1: $\mathbf{c}_{p1}, \mathbf{c}_{p2} \leftarrow \mathbf{c}_{p1}, \mathbf{c}_{p2} : \forall \mathbf{c}_{pi}, \mathbf{c}_{pj} \in R : d(\mathbf{c}_{p1}, \mathbf{c}_{p2}) \geq d(\mathbf{c}_{pi}, \mathbf{c}_{pj})$
2: $C \leftarrow C - \{p_1, p_2\}$
3: $D_r(S) \leftarrow \{p_1, p_2\}$
4: **while** $|D_r(S)| < r$ **do**
5: $\quad \mathbf{c}_q = \arg\max_{\mathbf{c}'_q \in C} \left(\sum_{p \in D_r(S)} d(\mathbf{c}'_q, \mathbf{c}_p) \right)$
6: $\quad D_r(S) \leftarrow D_r(S) \bigcup \{q\}$
7: $\quad C \leftarrow C - \{q\}$
8: **end while**
9: **return** $D_r(S)$

Implementation details. In each loop iteration of the *DPSA* algorithm (lines 4-8), the algorithm calculates the sum of distances between a centroid vector $\mathbf{c_q} \in C - D_r(S)$ and the centroid vectors in $D_r(S)$. As described above this procedure has a cost of $O(r^2|C|)$. In the case of the cosine distance we can reduce this cost to $O(r|C|)$ by exploiting the properties of the cosine function. As shown in Equation 1 the sum of distances of $\mathbf{c}_q$ to all centroids in $D_r(S)$ is equal to $|D_{r(S)}| - \frac{\mathbf{c}'_q}{|\mathbf{c}'_q|} \cdot \mathbf{c}_{D_r(S)}$. In that way it is only necessary to calculate the centroid of $D_r(S)$ before each loop iteration.

$$\begin{aligned} \sum_{p \in D_r(S)} d(\mathbf{c}'_q, \mathbf{c}_p) &= \sum_{p \in D_r(S)} 1 - cos(\mathbf{c}'_q, \mathbf{c}_p) \\ &= |D_r(S)| - \frac{\mathbf{c}'_q}{|\mathbf{c}'_q|} \cdot \sum_{p \in D_r(S)} \frac{\mathbf{c}_p}{|\mathbf{c}_p|} \\ &= |D_r(S)| - \frac{\mathbf{c}'_q}{|\mathbf{c}'_q|} \cdot \mathbf{c}_{D_r(S)} \end{aligned} \quad (1)$$

6. SELECTIVE TOP-K ALGORITHM

The main drawback of the previous algorithm is that it requires the computation of all centroids, which has a significant processing cost regardless of the employed method for candidate centroid computation. In particular, depending on the cardinality of W and S, the computation of the centroids may be time-consuming. In order to alleviate this shortcoming, in this section, we propose a method that fuses the centroid computation with the selection of diverse objects. Our goal is to efficiently compute an approximation of the centroids (by evaluating only a handful of carefully selected top-k queries), which is sufficient to produce a set of r products with high diversity.

6.1 Centroid Approximation

Conceptually, the proposed algorithm uses a series of iterations, where each iteration consists of three parts: (1) select

a weighting vector $\mathbf{w}_i$ in order to process the top-k query it defines, (2) compute an approximation of the centroid-set based on the results of all already processed top-k queries, and (3) select diverse products by invoking the *DPSA* algorithm (Section 5.2) with input the approximate centroid-set. In each iteration, a top-k query based on $\mathbf{w}_i$ is executed. Some objects $p \in TOP_k(\mathbf{w}_i)$ may not have been retrieved before and those are added to the centroid-set. For the remaining objects $p \in TOP_k(\mathbf{w}_i)$ the approximate centroid is updated, since $\mathbf{w}_i$ is added to their reverse top-k sets. In fact, the reverse top-k sets are not maintained, but the centroid of an object is computed progressively as in the case of *Atopk*. Thus, in each iteration the centroid-set is only an approximation of the candidate centroids C computed by Algorithm 1 because (a) C may contain more centroids as some objects may not have been retrieved yet and (b) the centroids of an object p are estimated based on a limited set of top-k queries only. However, in each iteration, the candidate-set is enriched with the results of the next top-k query. Additionally, a set of r diverse products $D_r(S)$ is computed based on the current set of centroids. Finally, the selection of the next weighting vector to be processed is based on maximizing the sum of distances to the set of centroids defined by $D_r(S)$.

The main idea of our algorithm is that the maximum cosine distance (i.e., maximum diversity) of two objects is bounded by the maximum cosine distance of any two weighting vectors. Let us assume that $\mathbf{w}_1$ and $\mathbf{w}_2$ are the two weighting vectors with the maximum cosine distance (the most diverse). Let us further assume that there exist two products p_1 and p_2 for which holds: $RTOP_k(p_1) = \{\mathbf{w}_1\}$ and $RTOP_k(p_2) = \{\mathbf{w}_2\}$. Then, it holds that for 2-Diversity problem the optimal solution is $\{p_1, p_2\}$ and their diversity equals to $1 - cos(\mathbf{w}_1, \mathbf{w}_2)$, since $\mathbf{c}_{p_i} = \mathbf{w}_i$. If more weighting vectors belong to $RTOP_k(p_1)$ then the diversity between $\{p_1, p_2\}$ decreases. Therefore, our algorithm starts by evaluating the top-k queries for the most diverse weighting vectors. In each step, the most diverse weighting vector to the current most diverse centroids is selected, as each centroid may summarize several weighting vectors.

6.2 Algorithmic Description

Algorithm 3 shows the pseudocode of the proposed algorithm that uses a limited set of top-k queries only. We call this algorithm *Selective Top-k Algorithm* and denote it with *Stopk*.

Description. The first major difference to Algorithm 1 is that the initial centroid computation is avoided. First, the algorithm computes a random sample W' (of size s) of W (line 1) and the two most dissimilar weighting vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ of W' are selected (line 2). Notice that the sample W' is produced uniformly at random, thus it follows the distribution of W and is used only for the initialization of $\widehat{\mathcal{C}}$. Applying the initialization step on W would result in a cost of $O(|W|^2)$, while with the sample it is reduced to $O(|W'|^2)$. Next, the top-k queries for $\mathbf{w}_1$ and $\mathbf{w}_2$ are processed and from the resulting merged set of products a set $\widehat{\mathcal{C}}$ of centroids is computed (line 3). Notice that $\widehat{\mathcal{C}}$ is computed based solely on the products retrieved thus far by the two top-k queries. These centroids form the candidate set for finding the most diverse products. In the following step, the Algorithm 2 is invoked with input the candidate set, and the two most diverse products are retrieved and placed in $D_r(S)$ (line 4). Note that $\widehat{\mathcal{C}}$ is much smaller than C, thus *DPSA* algorithm is applied on $\widehat{\mathcal{C}}$ and no random sample is required.

Algorithm 3 Selective Top-k Algorithm

Input: S: set of products
W: set of weighting vectors
k: value of top-k and reverse top-k
s : size of initial sample
r : required number of diverse products
steps : number of iterations (*steps* $\geq r$)
Output: $D_r(S)$: the set of r most diverse products of S

1: $W' \leftarrow$random subset of W with $|W'| = s$
2: $\mathbf{w}_1, \mathbf{w}_2 \leftarrow \mathbf{w}_1, \mathbf{w}_2 : \forall \mathbf{w}_i, \mathbf{w}_j \in W' : d(\mathbf{w}_1, \mathbf{w}_2) \geq d(\mathbf{w}_i, \mathbf{w}_j)$
3: $\widehat{\mathcal{C}} \leftarrow$ ComputeCentroids($\bigcup_{x=1,2} TOP_k(\mathbf{w}_x)$)
4: $D_r(S) \leftarrow DPSA(\widehat{\mathcal{C}}, \widehat{\mathcal{C}}, 2)$
5: $i = 2$
6: **while** $i <$ *steps* **do**
7: $i++$
8: $\mathbf{w}_i = \arg\max_{\mathbf{w} \in W} \left(\sum_{p \in D_r(S)} d(\mathbf{c}_p, \mathbf{w}) \right)$
9: $\widehat{\mathcal{C}} \leftarrow$ ComputeCentroids($\bigcup_{x=1\ldots i} TOP_k(\mathbf{w}_x)$)
10: $D_r(S) \leftarrow DPSA(\widehat{\mathcal{C}}, \widehat{\mathcal{C}}, min(i+1, r))$
11: **end while**
12: **return** $D_r(S)$

In each iteration, the most dissimilar weighting vector $\mathbf{w}_i$ to the centroid vectors $\mathbf{c}_p$ ($p \in D_r(S)$) is selected (line 8). For this $\mathbf{w}_i$, the respective top-k query is executed and the candidate list $\widehat{\mathcal{C}}$ is updated as before (line 9). Then, the *DPSA* algorithm is invoked again to produce a new set of diverse products (line 10). The same procedure is repeated for at least r steps. Notice that when the iteration counter i is smaller than r, the algorithm produces i diverse products, and only when i becomes greater than r does the algorithm return r diverse products.

In order to improve further the approximation of the centroids more iterations can be applied. The number of iterations (*steps*) is a system parameter that captures an interesting trade-off between the diversity of the result set and the processing time. Small values of *steps* increase the efficiency of the algorithm by reducing its processing time. In the experimental evaluation, we demonstrate that a small number of iterations is sufficient to produce results with diversity comparable to that of Algorithm 1, with significantly lower processing cost. Notice that in the extreme case that the number of iterations of *Stopk* is set equal to the cardinality of W and also no sampling is used ($s = |W|$), the set of diverse products and number of required top-k queries will be the identical with *Atopk*. Nevertheless, in this case, *Stopk* will have the computational overhead of applying multiple times the *DPSA* algorithm and finding the diverse weighting vectors.

EXAMPLE 2. *Table 3 shows an example of the execution of Stopk for $r = 2$. We assume that in the initialization step vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ are selected. Furthermore (assuming $k = 3$), the top-3 results for the selected vectors are depicted. After the initialization step, the sets of approximate centroids $\widehat{\mathcal{C}}$ contains 5 centroids (namely $\mathbf{c}_{p_1}$, ..., $\mathbf{c}_{p_5}$),*

Initialization step:
$TOP_k(\mathbf{w}_1) = p_1, p_2, p_3$, $TOP_k(\mathbf{w}_2) = p_2, p_4, p_5$ $\mathbf{c}_{p_1} = \mathbf{w}_1$, $\mathbf{c}_{p_2} = \frac{1}{2}(\mathbf{w}_1 + \mathbf{w}_2)$, $\mathbf{c}_{p_3} = \mathbf{w}_1$, $\mathbf{c}_{p_4} = \mathbf{w}_2$, $\mathbf{c}_{p_5} = \mathbf{w}_2$
First step:
$TOP_k(\mathbf{w}_3) = p_3, p_4, p_5$ $\mathbf{c}_{p_1} = \mathbf{w}_1$, $\mathbf{c}_{p_2} = \frac{1}{2}(\mathbf{w}_1 + \mathbf{w}_2)$, $\mathbf{c}_{p_3} = \frac{1}{2}(\mathbf{w}_1 + \mathbf{w}_3)$ $\mathbf{c}_{p_4} = \frac{1}{2}(\mathbf{w}_2 + \mathbf{w}_3)$, $\mathbf{c}_{p_5} = \frac{1}{2}(\mathbf{w}_2 + \mathbf{w}_3)$
Second step:
$TOP_k(\mathbf{w}_4) = p_1, p_2, p_6$ $\mathbf{c}_{p_1} = \frac{1}{2}(\mathbf{w}_1 + \mathbf{w}_4)$, $\mathbf{c}_{p_2} = \frac{1}{3}(\mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_4)$, $\mathbf{c}_{p_3} = \frac{1}{2}(\mathbf{w}_1 + \mathbf{w}_3)$, $\mathbf{c}_{p_4} = \frac{1}{2}(\mathbf{w}_2 + \mathbf{w}_3)$, $\mathbf{c}_{p_5} = \frac{1}{2}(\mathbf{w}_2 + \mathbf{w}_3)$, $\mathbf{c}_{p_6} = \mathbf{w}_4$

Table 3: Example of *Stopk*.

which correspond to the data points that have been retrieved by at least one top-k query. Algorithm 2 is applied on $\widehat{\mathcal{C}}$ and we assume that $\mathbf{c}_{p_1}$ and $\mathbf{c}_{p_4}$ as the two most diverse vectors. In the first iteration of Stopk, the most diverse (according to $\mathbf{c}_{p_1}$ and $\mathbf{c}_{p_4}$) weighting vector of W is selected. In this step, $\mathbf{w}_3$ is selected and the approximate centroids $\widehat{\mathcal{C}}$ are updated based on $TOP_k(\mathbf{w}_3)$ as depicted in Table 3. Again, Algorithm 2 is applied on $\widehat{\mathcal{C}}$ and $\mathbf{c}_{p_1}$ and $\mathbf{c}_{p_4}$ are identified as the two most diverse vectors. Stopk continues with a second iteration by evaluating $\mathbf{w}_4$. In this step, $\mathbf{c}_{p_6}$ is added to $\widehat{\mathcal{C}}$ as it belongs to $TOP_k(\mathbf{w}_4)$. Again Algorithm 2 will be invoked and the most dissimilar weighting vector of W will be selected. The same procedure continues until steps iterations has been executed.

Complexity. To perform a cost analysis of the algorithm, we need to identify its basic cost factors. These factors include the initial computation of the two most dissimilar vectors ($O(s^2)$), the processing cost of *steps* top-k queries, the cost of determining the next most dissimilar weighting vector ($O(steps \cdot r \cdot |W|)$) (line 8), and the cost induced by invoking the *DPSA* algorithm *steps* times. The cost of processing *steps* top-k queries will always be much cheaper than Algorithm 1, which needs to process W top-k queries (in the case of *Atopk*) to perform the centroid computation. It should also be noted that the calls to the *DPSA* algorithm are much cheaper, because it operates on $\widehat{C}$ which is much smaller than C. Overall, the cost of the algorithm is $s^2 + steps \cdot r \cdot |W|$, since these are the dominant cost factors, and the complexity is linear with respect to W ($O(steps \cdot r \cdot |W|)$), when *steps* is small (r is typically small anyway).

7. MAINTENANCE

In this section, we present techniques for maintaining the diverse set of products in the case of dynamic data. In fact, the methodology of *Stopk* (Algorithm 3) can be applied to maintain the r-diverse products. We consider two cases: (1) new products are inserted in the product database, and (2) new preferences (in the form of weighting vectors) are added in the customer preference database. Both cases actually occur in online shops, when new products appear in the market and new customer preferences are extracted from social sites.

In order to support product insertions efficiently, we exploit the top-k queries that where computed during the computation of $\widehat{C}$. Let W^* be the set of weighting vectors for which the top-k queries have been computed. We maintain for each weighting vector $\mathbf{w} \in W^*$ the score of the k-th product. When a new product p_{new} is inserted in the database, we check for each query $\mathbf{w} \in W^*$ if p_{new} has a better score than the k-th product. If this does not occur for any $\mathbf{w} \in W^*$, we can safely ignore p_{new}, as it does not affect the determination of the diverse products. On the other hand, if p_{new} becomes top-k for some weighting vector $\mathbf{w}$, we compute the new centroids only for the affected products (i.e., p_{new} and the products p_i that used to be at the k-th rank, but were evicted by p_{new}) and update the set $\widehat{C}$. We then apply *DPSA* algorithm on $\widehat{C}$ and produce a new set of diverse products. Note that in the first case, we can ensure that the result set is the same as in the case where *Stopk* would be executed on the updated data set, but this does not hold for the second case. The similarity of the centroids before and after the update can be used in order to decide when the *Stopk* algorithm should be invoked to have a result set of higher quality.

In order to be able to handle new preferences effectively, during the computation of the diverse product we maintain the minimum (min) of all maximal sums of distances between a centroid and any selected weighting vector (i.e., the expression in line 8 of Algorithm 3). In the case of a new weighting vector $\mathbf{w}_{new}$, we follow the same principle as Algorithm 3 to decide whether the respective top-k query should be evaluated. If $\sum_{p \in D_r(S)} d(\mathbf{c}_p, \mathbf{w}_{new})$ is larger than min, then the top-k query for $\mathbf{w}_{new}$ is computed, the set of centroids $\widehat{C}$ updated and *DPSA* algorithm is executed on $\widehat{C}$ to produce a new set of diverse products. Intuitively, when vector $\mathbf{w}_{new}$ induces small changes to the set of centroids, we do not need to recompute the r-diverse products as $\mathbf{w}_{new}$ would not have been selected by *Stopk* in any case. Again, the retrieved diverse products are not the same as if *Stopk* would be executed on the updated weighting vector set, therefore a threshold on the similarity of the centroids before and after the update may trigger executing *Stopk* to get a set of higher quality.

8. SUPPORTING OTHER SET-BASED SIMILARITY FUNCTIONS

In general, our approach is applicable also for other functions that compute the similarity between sets of vectors. In such a case, our algorithms would not calculate centroids (which is simply a representation of a set of weighting vectors), but would instead directly operate on the reverse top-k sets of products. Following this line of thought, $\hat{C}$ would represent a set of approximate reverse top-k sets (instead of a set of centroids) and the computation of the most diverse sets becomes independent of the similarity or distance function.

In more technical terms, Algorithm 1 would not calculate centroids but would only maintain the reverse top-k sets, and Algorithm 3 would not compute centroids incrementally but would simply update the approximate reverse top-k sets of products based on the executed top-k queries. Then, Algorithm 2 can be directly applied to the reverse top-k sets.

For instance, one popular similarity function is the Jac-

Parameter	Values	
Datasets	UN, CO, AC, CL NBA, HOUSE	
Data cardinality	1K, **5K**, 10K 5K, **10K**, 30K **100K**, 200K, 500K 17265, 127930	(Diversity Quality) (Scalability Analysis) (Sensitivity Analysis) (Real Datasets)
Weight cardinality	1K, **5K**, 10K 5K, **10K**, 30K **100K**, 200K, 500K **100K**, 200K, 500K	(Diversity Quality) (Scalability Analysis) (Sensitivity Analysis) (Real Datasets)
# results(r)	3, 4, **5** 10 5, **10**, 30 5, **10**, 30	(Diversity Quality) (Scalability Analysis) (Sensitivity Analysis) (Real Datasets)
# top-k results(k)	**10**, 20 5, **10**, 30, 50 5, 10, **30**, 50 5, 10, **30**, 50	(Diversity Quality) (Scalability Analysis) (Sensitivity Analysis) (Real Datasets)
# dimensions(m)	3 3 **3**, 4, 5, 6	(Diversity Quality) (Scalability Analysis) (Sensitivity Analysis)

Table 4: Parameters (default values in bold).

card similarity, which is defined as the size of the intersection divided by the size of the union of two sets. Our approach readily supports the Jaccard similarity on reverse top-k sets, as outlined above. Notice that the advantage of the cosine similarity compared to Jaccard for the problem of finding diverse products is that it returns more fine-grained similarity values. For example, in the case of disjoint sets, the Jaccard similarity value equals to zero, and does not distinguish between the sets. Instead, the cosine similarity of the centroid vectors allows us to distinguish between them by returning a non-zero value. Moreover in the case of a set A that is a subset of another set B ($A \subseteq B$), the Jaccard similarity is equal to $\frac{|B|-|A|}{|B|}$ which can get arbitrarily close to the maximum value. In such cases, using the Jaccard similarity would not be helpful, as it could lead to selecting products which are possibly covered by others. The centroid vectors reduce this problem (they do not eliminate it) by choosing sets that are selected by distant user preferences.

9. EXPERIMENTAL EVALUATION

In this section, we present the results of the experimental evaluation. All algorithms were implemented in Java and the experiments run on 2x Intel Xeon X5650 Processors (2.66GHz). The algorithms are disk-based and the index structure used was an R-tree with a buffer size of 100 blocks and the block size is 4KB. The main parameters and values used through the experiments are presented in Table 4.

Data sets. For the data set S, we use both synthetic and real data. We examine four different synthetic data distributions, namely uniform (UN), correlated (CO), anticorrelated (AC) and clustered (CL). For the uniform data set, the data object values for all m dimensions are generated independently using a uniform distribution. The correlated and anticorrelated data sets are generated as described in [3]. The clustered data set was created as follows: first 5 cluster centroids were selected randomly. Then, each coordinate is generated on the m-dimensional space by following a normal distribution on each axis with variance $\sigma_S^2 = 0.345$, and a mean equal to the corresponding coordinate of the centroid. In addition, we use a real data set. 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. For the data set W of the weighting vectors, we used a uniform (UN) distribution.

Algorithms. We implemented the three algorithms for centroid computation (*Rtopk*, *Itopk*, and *Atopk*) coupled with the *DPSA* algorithm as described in Section 5, and the selective top-k algorithm (*Stopk*) described in Section 6. We also implemented an exact algorithm (denoted *opt*) that finds the optimal solution, but obviously cannot scale well. For reverse top-k processing, *Rtopk* uses the state-of-the-art BBR* algorithm [23], while *Itopk* uses the BB algorithm [22]. In all algorithms the data set is indexed by an R-tree and for the top-k query processing we employ a state-of-the-art branch-and-bound algorithm [17].

Metrics. The metrics under which we evaluated the implemented algorithms were: (a) execution time required by each algorithm, (b) I/O accesses, (c) achieved diversity values. We also measured the number of processed top-k queries, but in the interest of space we do not report them since they follow exactly the I/O metric. We measure only the I/O induced on the data set S, since the I/O on W are caused by sequential access and accessing data sequentially is much faster than the random accesses of S.

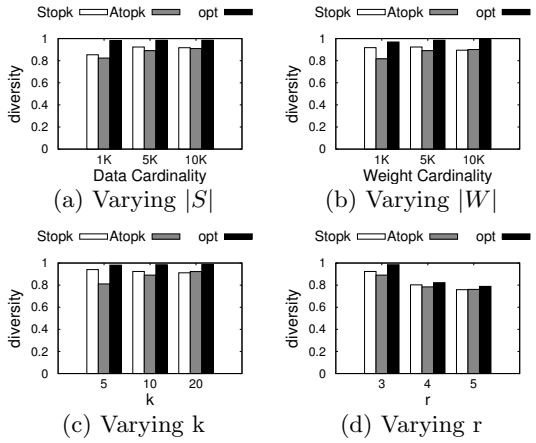
We conduct an experimental study varying the parameters of dimensionality (3-6), cardinality (1K-500K) of S, cardinality (1K-500K) of W, value of k (5-50), value of r (3-30), sample size $|W'|$ ($0.001|W|$-$0.1|W|$), and number of *steps* (100-1000). Each experiment was repeated 10 times over different instances of the data sets with the same parameters and different seed to the random generator, in order to factor out the effect of randomization. Average values are reported in all cases.

Evaluation methodology. Our evaluation was divided in three parts. In the first part (9.1), we compare the algorithms *Atopk* and *Stopk* against the exact algorithm (*opt*) in order to evaluate the quality of approximation of diversity. In the second part (9.2), we evaluate the performance of *Stopk* against the algorithms that rely on centroid computation (*Rtopk*, *Itopk*, and *Atopk*). In the last part (9.3 and 9.4), we perform a thorough sensitivity analysis of *Atopk* (which proved to perform best among the algorithms with centroid computation) against *Stopk*. We should stress that the diversity of the result set of *Stopk* is calculated using the whole set of preferences W, and not only the vectors used for the identification of the most diverse products.

9.1 Quality of Diversity

The purpose of this series of experiments is to study the loss of diversity compared to optimal solution when using our algorithms *Atopk* and *Stopk*. Thus, we compare to the optimal diversity produced by an exact algorithm (*opt*), which examines all possible $\binom{|S|}{r}$ combinations of products exhaustively to find the optimal solution. Recall that *Rtopk* and *Itopk* produce exactly the same result set as *Atopk* and therefore have also the same diversity. The default setup for this series of experiments was: m=3, $|S| = 5K$, $|W| = 5K$, $k = 10$, $r = 5$, $s = 0.1 \cdot |W|$, $steps = 100$, S:UN, and W:UN.

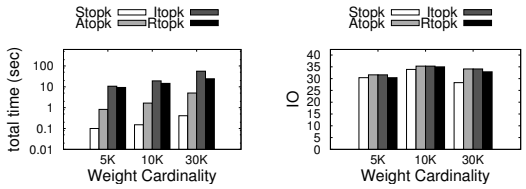
Figure 2 depicts the diversity values for varying different parameters, namely $|S|$, $|W|$, r and k. The diversity achieved by the greedy algorithm (*Atopk*) is in most cases quite close to optimal, while *Stopk* results in similar diver-



(a) Varying $|S|$ (b) Varying $|W|$

(c) Varying k (d) Varying r

Figure 2: Comparison to optimal diversity value.



(a) Time (b) I/O

Figure 3: Performance when varying $|W|$.

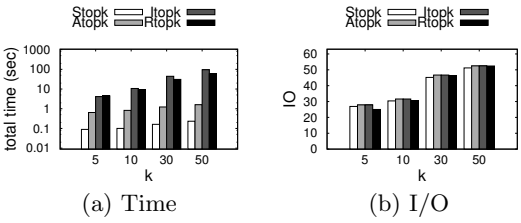
sity values. As we can observe, our approximate algorithms perform very well in comparison with the exact algorithm. In the worst case, when the size of the objects data set is only $|S|$=1000 objects (Figure 2(a)), the maximum difference in diversity is 19%. As the datasets grow in size, the diversity of the result set of the approximate algorithms approaches the optimal diversity. It is noteworthy that as the number of returned objects (r) increases, the diversity value drops. This behavior is expected as the more points we select the smaller their average distance will become.

We omit the figures comparing the performance of our algorithms to *opt*, since, as expected, our algorithms outperformed the exact approach by 1-4 orders of magnitude in terms of execution time.

9.2 Scalability Analysis

In this series of experiments we compare the performance of the algorithms with centroid computation (described in Section 5) in terms of execution time and I/O. We also include the *Stopk* algorithm in the charts for completeness. The default setup for this series of experiments is m=3, $|S| = 10K$, $|W| = 10K$, $k = 10$, $r = 10$, S:UN, W:UN, $steps = 100$, $s = 0.01 \cdot |W|$.

Varying weight cardinality $|W|$. As Figure 3 illustrates, *Atopk* and *Stopk* outperform by orders of magnitude the *Rtopk* and *Itopk* algorithms in terms of execution time. This difference is not reflected in the measured I/O, because of the use of the buffer of the R-tree. When the number of issued top-k queries is considered, both *Rtopk* and *Itopk* process at least one order of magnitude more top-k queries than *Atopk* and *Stopk*. This processing cost is responsible for their slow runtime. We note that even though both *Rtopk* and *Itopk* are more efficient than *Atopk* when a single



(a) Time (b) I/O

Figure 4: Performance when varying k.

reverse top-k query or a small number of influential points is needed, they are less efficient when they are run repeatedly multiple times. In this case, *Atopk* has a benefit and performs better. In particular, *Rtopk* has no memory of the completed executions for different queries, and therefore it computes repeatedly the top-k results of many preference vectors. On the other hand, *Itopk* performs fewer reverse top-k queries than *Rtopk*, but shares the same shortcoming for those reverse top-k queries that it processes. Therefore, it faces the same problem as the *Rtopk*, however in not such great extent. Similar conclusions are drawn when varying the data cardinality $|S|$. The performance of the algorithms is much less affected by the increase of data cardinality, as during the execution of the top-k queries very few data objects are accessed.

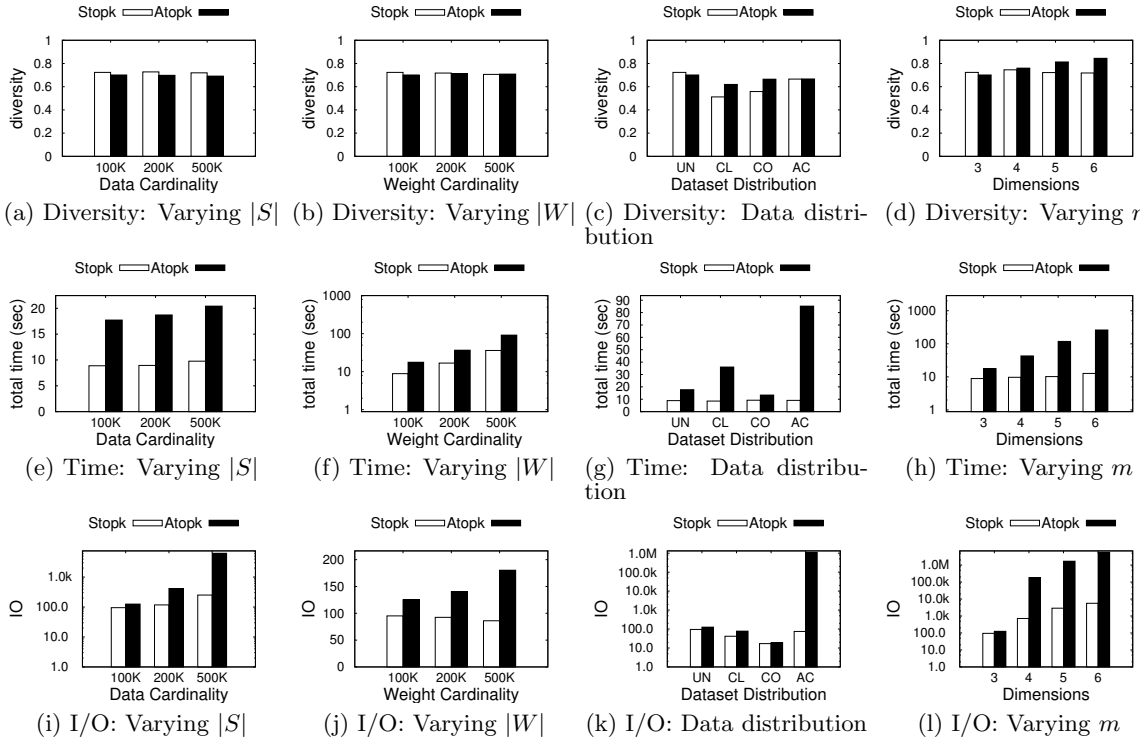
Varying k. Figure 4 illustrates the effect of varying parameter k. *Atopk* consistently outperforms *Rtopk* and *Itopk* in terms of time, while *Stopk* improves further the performance in terms of both time and I/O. *Atopk*, *Rtopk* and *Itopk* have the same performance in terms of I/O due to the R-tree buffer employed during query processing. Furthermore, for all algorithms, both time and I/O increase for increasing values of k.

Since *Atopk* consistently outperforms the two other algorithms (*Rtopk* and *Itopk*) that rely on centroid computation, we use only *Atopk* in the remaining experiments as representative of this family of algorithms.

9.3 Sensitivity Analysis

In this section, we provide a detailed sensitivity analysis by varying different parameters that influence the performance of our proposed algorithms. Due to space limitations, for some setups we omit the figures depicting the I/O since the time indicates the efficiency of the algorithms. The default setup for this series of experiments is m=3, $|S| = 100K$, $|W| = 100K$, $k = 30$, $r = 10$, S:UN, W:UN, $steps = 500$, $s = 0.01 \cdot |W|$.

Sensitivity analysis for varying $|S|$, $|W|$, Data Distribution and m. In Figure 5, we study the behavior of *Atopk* and *Stopk* for increasing cardinality of the data set S and the weighting vectors W, as well as for various data distributions (UN,CL,CO,AC) and dimensionality values m. First, we examine how the diversity values of *Atopk* and *Stopk* for the different parameters are influenced (Figures 5(a)–5(d)). We observe that *Stopk* benefits from the increased size of the data set and the preferences set. *Stopk* retrieves a set of r objects that have similar diversity compared to *Atopk*. In most cases the diversity achieved by *Stopk* is almost equal to the one achieved by *Atopk* and in some cases it is even slightly higher. This happens because *Stopk* locates the most diverse preferences and based



(a) Diversity: Varying $|S|$ (b) Diversity: Varying $|W|$ (c) Diversity: Data distribution (d) Diversity: Varying m

(e) Time: Varying $|S|$ (f) Time: Varying $|W|$ (g) Time: Data distribution (h) Time: Varying m

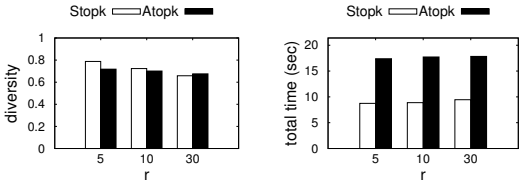
(i) I/O: Varying $|S|$ (j) I/O: Varying $|W|$ (k) I/O: Data distribution (l) I/O: Varying m

Figure 5: Sensitivity analysis for varying $|S|$, $|W|$, data distribution and m.

on them it identifies the most diverse products. *Atopk* on the other hand, bases the search for most diverse objects on the centroids of the $RTOP_k$ sets of the products. For increased dimensionality, as depicted in Figure 5(d), the diversity value of *Stopk* compared to *Atopk* is influenced more than for the other parameters. Recall that the diversity between the products was calculated for *Stopk* using all vectors in W and not only the ones used for the identification of the products.

Figures 5(e)–5(h) show the execution time of the algorithms with respect to various parameters. In Figure 5(e) we notice that none of the algorithms is influenced significantly by the data set cardinality, because top-k queries require retrieving only few data objects that are highly ranked independently of the data set cardinality. Figure 5(f) (which uses log-scale) shows that both algorithms are influenced in a similar way while varying the number of weighting vectors $|W|$, but *Stopk* is always more efficient than *Atopk*. In particular, the time increases with increasing number of weighting vectors $|W|$. In Figure 5(g), we depict the performance of both algorithms for different data distributions. In the case of anticorrelated data the execution time for *Atopk* is 4 times larger that in the case of a uniform distribution. On the other hand, the performance of *Stopk* is not affected by anticorrelated values significantly. Figure 5(h) shows that *Stopk* scales nicely for increased dimensionality, in contrast to *Atopk* whose cost increases by one order of magnitude when we go from 3 to 6 dimensions. This experiment provides strong evidence for the scalability of *Stopk* with increased number of dimensions.

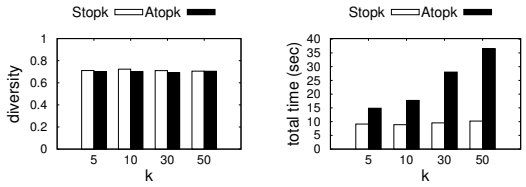
Figures 5(i)–5(l) evaluate the performance of our algorithms in terms of I/O accesses. The conclusions for the I/O accesses are similar to those for the time, except for the case of increasing the data set cardinality $|S|$. In Figure 5(i), we notice that even though the time is not increased by varying $|S|$, the I/O accesses increase. Naturally, more I/O are needed for processing a top-k query of a larger data set, but this is not reflected on the time. Due to the buffering, the computational cost of processing multiple top-k queries is more significant than the time needed to retrieve the relevant index nodes. Nevertheless, in all cases *Stopk* outperforms *Atopk* in terms of I/O. For example Figure 5(l) shows that *Stopk* is two orders of magnitude cheaper than *Atopk* in I/O accesses as we increase the dimensions.



(a) Diversity (b) Time

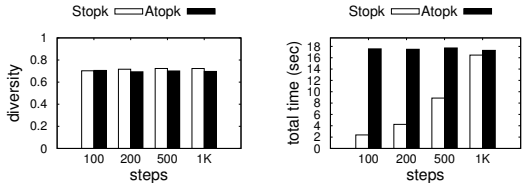
Figure 6: Varying r.

Varying r. Figure 6 examines the effect of varying the number r of retrieved products on our algorithms. First, we observe in Figure 6(a) a decreasing tendency of the diversity value as r increases for both algorithms, which is expected as also the diversity value of the optimal solution will decrease as the most diverse products are selected first. As far as the performance is considered, in Figure 6(b), the time of *Atopk* is not influenced by the increase of r, because *Atopk* computes all top-k queries independently of the size of the result set r and the computational cost of *DPSA* algorithm



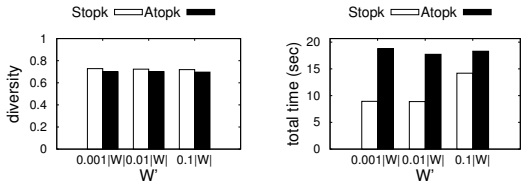
(a) Diversity (b) Time

Figure 7: Varying k.



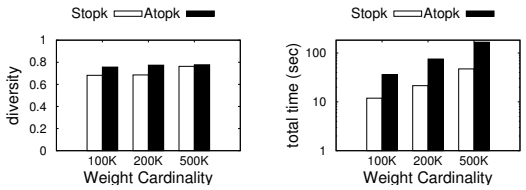
(a) Diversity (b) Time

Figure 8: Varying *steps*.



(a) Diversity (b) Time

Figure 9: Varying W'.



(a) Diversity: varying |W| (b) Time: varying $|W|$

Figure 10: House Dataset: varying |W|

is not significant compared to the cost of the top-k queries. *Stopk* is also not significantly affected, since the values of r are relatively small, and the algorithm is executed *steps* times in any case. Still, *Stopk* remains always much faster than *Atopk*.

Varying k. In Figure 7, we gradually increase the parameter k of the reverse top-k queries from 5 to 50. In Figure 7(a), we notice that the diversity value is stable as k increases, which seems counter-intuitive at first. By increasing k the size of the reverse top-k set increases for some objects and more objects have a non-empty reverse top-k set. However, this does not influence the diversity value significantly, as the most diverse centroids may not change. Figure 7(b) depicts the time obtained for different values of k. Although we witness a small deterioration in the performance of both algorithms, *Stopk* consistently outperforms *Atopk*. Processing top-k queries is more time-consuming for higher values of k and the *DPSA* algorithm gets slower with increasing k because the number of candidates for finding the diverse objects increases. We should add however, that the effect of parameter k has much smaller impact on the performance of *Stopk* because *Stopk* performs a small number of top-k queries.

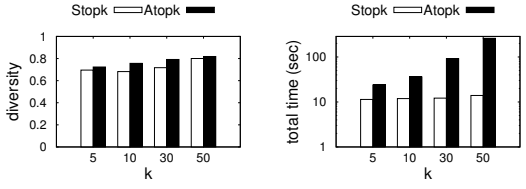
Varying steps. The *steps* parameter is an essential parameter for the *Stopk* algorithm as it balances the efficiency of the algorithm and the diversity the algorithm achieves. Recall that *Stopk* performs only *steps* top-k queries, which is only a small fraction of the $|W|$ top-k queries that *Atopk* performs. On the other hand, *Stopk* executes also *steps* times *DPSA* algorithm algorithm on a small set of approximate centroids, which is not necessary for *Atopk*. In Figure 8, we observe that the diversity achieved using very few vectors is quite close to the diversity achieved by *Atopk*. As we increase the *steps* parameter the achieved diversity increases marginally. However the execution time increases proportionally with the increase of the *steps* parameter. This experiment verifies that a small value of *steps* is sufficient to produce results of high diversity in a very efficient way.

Varying sample size $|W'|$. The size of sample of preferences from which we select the two initial centroids plays an important role in the performance of the *Stopk* algorithm. The complexity of the selection process is $O(|W'|^2)$ and therefore a large sample can have significant impact on the performance of the algorithm. However, as shown in Figure 9, the increased cost in performance is not accompanied by an increased gain in diversity. The reason behind this fact is that once the sample is large enough to offer a good representation of the whole set of preferences, further enlargement will not help significantly in finding better initial centroids.

9.4 Results on Real Data

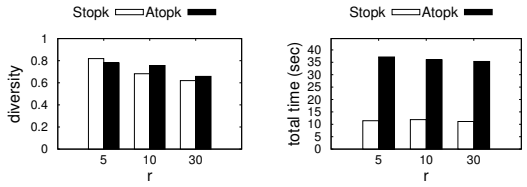
We have also performed an evaluation of our algorithm using a real data set. The conclusions drawn are overall in accordance with the conclusions made by the evaluation with synthetic data, thus verifying our findings. The default setup for this series of experiments is $|S| = 127930$, $|W| = 100K$, $k = 30$, $r = 10$, W:UN, $steps = 500$, $s = 0.01 \cdot |W|$. The size of the data set used and the high complexity of the exact algorithm (*opt*) did not allow the exact algorithm to terminate and therefore we did not include its performance results in this series of experiments.

Analysis for varying $|W|, k,$ and r. Figures 10-12 depict performance of the two algorithms. For all values of the varying parameters *Stopk* achieves diversity values close to the ones of *Atopk*. For both algorithms we notice a drop in the diversity values when r is increases which is expected as analyzed in 9.1. With respect to processing time it is evident



(a) Diversity: varying k (b) Time: varying k

Figure 11: House Dataset: varying k



(a) Diversity: varying r (b) Time: varying r

Figure 12: House Dataset: varying r

that both parameters $|W|$ and k play a significant role in the performance of *Atopk*. This does not come as a surprise as the processing cost of *Atopk* is dominated by the processing cost of the top-k queries needed for the computation of the centroids of the $RTOP_k$ sets for each product. On the contrary *Stopk* is much less affected by those parameters as it performs a limited number of top-k queries. The increase of parameter r has little effect in both algorithms. The performance difference with respect to I/O is in all cases larger than two orders of magnitude. Only exception is for $k < 10$ where *Stopk* is one order of magnitude more efficient.

10. CONCLUSIONS

In this paper, we address the important problem of selecting the r most diverse products based on customers' preferences. The reverse top-k set of each product is represented by its centroid and the distance between centroids is then expressed using cosine distance. In order to find products that are attractive to customers with dissimilar preferences, we define the r-Diversity problem as a *dispersion* problem applied on the products' reverse top-k sets. As dispersion problems are known to be NP-hard, we propose two approximate algorithms that solve the problem. The first algorithm computes the reverse top-k sets and then applies a greedy algorithm that retrieves a set of products of high diversity. The second applies the greedy algorithm on an approximation of the reverse top-k sets by evaluating only some carefully selected top-k queries. In our experimental evaluation, we study the performance of the proposed algorithms and the diversity of the retrieved products in various experimental setups. In particular, we demonstrate that our algorithms both achieve diversity values close to optimal and are very efficient in practice.

Acknowledgments

The work of A. Vlachou was supported by the Action "Supporting Postdoctoral Researchers" of the Operational Program "Education and Lifelong Learning" (Action's Beneficiary: General Secretariat for Research and Technology), and is co-financed by the European Social Fund (ESF) and the Greek State. The research of C. Doulkeridis has been co-financed by ESF and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Aristeia II, Project: ROADRUNNER.

11. REFERENCES

[1] A. Angel and N. Koudas. Efficient diversity-aware search. In *Proc. of SIGMOD*, pages 781–792, 2011.

[2] A. Arvanitis, A. Deligiannakis, and Y. Vassiliou. Efficient influence-based processing of market research queries. In *Proc. of CIKM*, pages 1193–1202, 2012.

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

[4] S. Chester, A. Thomo, S. Venkatesh, and S. Whitesides. Indexing reverse top-k queries in two dimensions. In *Proc. of DASFAA (1)*, pages 201–208, 2013.

[5] M. Drosou and E. Pitoura. Diversity over continuous data. *IEEE Data Eng. Bull.*, 32(4):49–56, 2009.

[6] M. Drosou and E. Pitoura. DisC diversity: result diversification based on dissimilarity and coverage. *PVLDB*, 6(1):13–24, 2012.

[7] M. Drosou and E. Pitoura. Dynamic diversification of continuous data. In *Proc. of EDBT*, pages 216–227, 2012.

[8] E. Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990.

[9] S. Ge, L. H. U, N. Mamoulis, and D. W. Cheung. Efficient all top-k computation - a unified solution for all top-k, reverse top-k and top-m influential queries. *TKDE*, 25(5):1015–1027, 2013.

[10] J.-L. Koh, C.-Y. Lin, and A. Chen. Finding k most favorite products based on reverse top-t queries. *The VLDB Journal*, pages 1–24, 2013.

[11] 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.

[12] C.-Y. Lin, J.-L. Koh, and A. L. P. Chen. Determining (k)-most demanding products with maximum expected number of total customers. *IEEE Trans. Knowl. Data Eng.*, 25(8):1732–1747, 2013.

[13] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In *Proc. of ICDE*, pages 86–95, 2007.

[14] 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.

[15] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

[16] Y. Tao, L. Ding, X. Lin, and J. Pei. Distance-based representative skyline. In *Proc. of ICDE*, pages 892–903, 2009.

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

[18] G. Valkanas, A. N. Papadopoulos, and D. Gunopulos. SkyDiver: a framework for skyline diversification. In *Proc. of EDBT*, pages 406–417, 2013.

[19] M. R. Vieira, H. L. Razente, M. C. N. Barioni, M. Hadjieleftheriou, D. Srivastava, C. T. Jr., and V. J. Tsotras. DivDB: a system for diversifying query results. *PVLDB*, 4(12):1395–1398, 2011.

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

[21] A. Vlachou, C. Doulkeridis, Y. Kotidis, and K. Nørvåg. Monochromatic and bichromatic reverse top-k queries. *TKDE*, 23(8):1215–1229, 2011.

[22] 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.

[23] A. Vlachou, C. Doulkeridis, K. Nørvåg, and Y. Kotidis. Branch-and-bound algorithm for reverse top-k queries. In *Proc. of SIGMOD*, pages 481–492, 2013.

[24] Q. Wan, R. C.-W. Wong, I. F. Ilyas, M. T. Özsu, and Y. Peng. Creating competitive products. *PVLDB*, 2(1):898–909, 2009.

[25] T. Wu, Y. Sun, C. Li, and J. Han. Region-based online promotion analysis. In *Proc. of EDBT*, pages 63–74, 2010.

[26] T. Wu, D. Xin, Q. Mei, and J. Han. Promotion analysis in multi-dimensional space. *PVLDB*, 2(1):109–120, 2009.

[27] A. Yu, P. K. Agarwal, and J. Yang. Processing a large number of continuous preference top-k queries. In *Proc. of SIGMOD*, pages 397–408, 2012.