Call to Order: A Hierarchical Browsing Approach to Eliciting Users' Preference

Feng Zhao[1], Gautam Das[2], Kian-Lee Tan[1], Anthony K. H. Tung[1]

[1]School of Computing
National U. of Singapore
{zhaofeng,tankl,atung}@comp.nus.edu.sg

[2]CSE Department
UT Arlington
gdas@uta.edu

ABSTRACT

Computing preference queries has received a lot of attention in the database community. It is common that the user is unsure of his/her preference, so care must be taken to elicit the preference of the user correctly. In this paper, we propose to elicit the preferred ordering of a user by utilizing skyline objects as the representatives of the possible ordering. We introduce the notion of *order-based representative skylines* which selects representatives based on the orderings that they represent. To further facilitate preference exploration, a hierarchical clustering algorithm is applied to compute a denogram on the skyline objects. By coupling the hierarchical clustering with visualization techniques, we allow users to refine their preference weight settings by browsing the hierarchy. Extensive experiments were conducted and the results validate the feasibility and the efficiency of our approach.

Categories and Subject Descriptors

H.2.8 [**Databases Management**]: Database applications

General Terms

Algorithm, Performance

Keywords

Preference Query, Skyline Query, Sampling, Visualization

1. INTRODUCTION

Computing preference queries has been a well studied problem in the database community [16, 6, 15, 22]. Preferences, treated as soft constraints, are utilized in multi-criteria decision situations to identify the preferred results of manageable size [16]. Among various possible problem settings, a common one [15, 22] assumes that a monotonic ranking (or preference) function $P(\cdot)$ is provided and the user will specify his/her preference by setting a set of weights $w = \{w_1, w_2, \ldots, w_d\}$ which are used within the preference function to rank the importance of data objects. Each of the weight w_i represents the importance of an attribute A_i describing the objects and thus w_1, ..., w_d describe the importance of d attributes A_1,..., A_d. In such a problem setting, it is also assumed that the order of preference for the domain values of each attribute are known. As such, if the user is able to specify the settings of the weights correctly, then the objects will be ranked in the correct order of his/her preference and then the problem becomes one of retrieving the objects efficiently based on the order. However, if the user is unsure of his/her preference (which is typically the case), it is crucial to interact with the user to obtain a correct set of weights that represent his/her preference. Designing an effective mechanism to elicit the preference of the user is exactly what we set to do in this paper.

To elicit an user's preference, a common approach is to present the user with a set of objects, and based on his/her choice of the objects, we can potentially infer the correct weights. To ensure that all possible choices are well covered, the set of objects being presented must be carefully selected. More often than not, this involves clustering the objects into different groups and a representative from each group will be presented to the user. By stating the preference for a particular representative, he/she implicitly provides an approximate setting for the set of weights and also indicates that he/she prefers the group that is associated with the representative. Further refinement can then be made by repeating the procedure on the selected group and selecting more representatives from the group. However, such an approach will bring about a dilemma. In a typical clustering operation, an appropriate similarity function will be required to determine the similarity between the objects. Such a similarity function will usually be determined by weighting the importance of the attributes based on the user's input. The user unfortunately is relying on the clustering results to help him/her determine the importance of these attributes in the preference function!

In view of this, much research has been done on the problem of skyline computation [3, 7, 25, 17, 24, 18]. An object p dominates another object q if p is better or equal to q in all attributes and at least better than q in one. The skylines

SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA.

of a set of objects are those that are not dominated by any other objects in the set. Based on this definition, it can be shown that the set of skyline objects for a dataset is insensitive to (1) the weight assigned to each attribute and (2) the preference function being adopted. More importantly, given any monotonic preference function, it is guaranteed that the top one will always be a skyline object. More formally, let $\pi_w(D)$ denote the preferred ordering of a set of objects given weight setting w and $\pi_w(D)[i]$ denote the i^{th} object in this ordering, then $\pi_w(D)[1]$ must be a skyline object. In this sense, we will refer to $\pi_w(D)[1]$ as a **representative** of $\pi_w(D)$ and thus every possible ordering based on different weight settings will be represented by one of the skyline objects.

Since the set of skyline objects is insensitive to the setting of the weights and gives full coverage as representatives of $\pi_w(D)$, it thus makes sense to present the skylines to the user for selection and infer the weight setting that represents the user's preference based on his/her selection[1]. However, it has been shown in [25] that the expected number of skyline objects is $\Theta(\ln^{d-1} n/(d-1)!)$ for a random dataset where d is the dimensionality of the data. The large number of skyline objects for high dimensional dataset is ironical since this is the situation in which users have most difficulty determining their preferences and comparing products. Various efforts have been made [21, 28] to overcome this problem by selecting k representatives from a large set of skylines. While we will discuss these later in the related work section, it suffices to point out here that none of these works try to bring the preference function and its ordering of the objects back into the picture.

In this paper, we propose to elicit the preferred ordering of a user by utilizing skyline objects as representatives of the possible ordering. Our approach tries to find k representative skylines that best capture the orderings that are associated with other skyline objects. This brings about two challenges:

1. Given a dataset D, let $W_p(D)$ denote the set of weight settings such that for every $w \in W_p$, $\pi_w(D)[1] = p$, $p \in D$. In this case, $W_p(D)$ is a set of weight settings in which the object p will be ranked first and such a set could potentially be of infinite memberships. As such, comparing the ordering represented by two skyline objects becomes difficult.

2. Given that $\pi_w(D)$ represents a ranking with large number of objects, comparing any instance of the rankings represented by two skyline objects will require computationally efficient solutions to be developed.

In order to overcome these problems, we propose an indirect notion of similarity between the orderings that are represented by two skyline objects p and q. We claim that the ordering of q is close to p if q has a high probability of ranking high whenever p is ranked first in the ordering. Based on this notion which we will formally define later, we make various contributions towards eliciting users' preference based on hierarchical browsing of skylines:

- We introduce the notion of **order-based representative skylines** which selects representative skylines based on the ordering that they represent. Unlike previous work, we bring the preference function back into the picture when determining representative skylines since our aim is to elicit the preference of the user based on these representatives.

- To handle the two problems that we presented earlier, we define a notion of similarity that avoids explicit comparison of the orderings that are represented by two skyline objects. Based on this similarity measure, we develop sampling techniques that allow us to efficiently and accurately estimate the similarity between any two skyline objects. The similarity measure also allows us to define a goodness measure for clustering skyline objects, and a k-partitioning clustering algorithm is developed to cluster skyline objects based on this goodness measure.

- By applying the k-partitioning algorithm recursively, we create a hierarchical clustering of the skyline objects. By coupling hierarchical clustering with visualization techniques, we enable users to refine their preference weight settings by browsing the hierarchy.

- We conducted extensive experiments, and the results show that our approach is both effective and efficient.

The rest of the paper is organized as follows. Section 2 gives our new definition of representative skylines and shows the defects of existing methods. Section 3 reviews the related work on skyline query and preference elicitation. Section 4 presents the efficient sampling algorithm, and hierarchical browsing to elicit users' preference is described in Section 5. Results of our extensive experimental study are reported in Section 6. Section 7 concludes the paper.

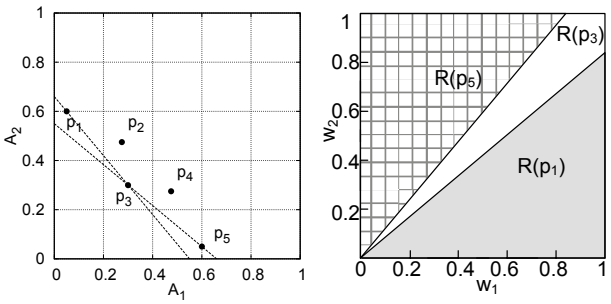
2. PRELIMINARY

2.1 Problem Definition

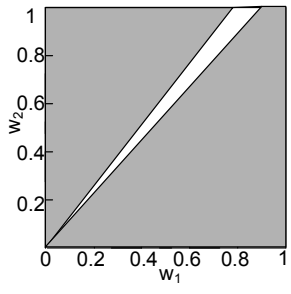
We have a database D of n objects. Each object is described by d attributes A_1, ..., A_d. We will use $p.A_i$ to refer to the value of an attribute A_i for an object p. For ease of discussion, we assume that all of these attributes are numerical attributes ranging from 0 to 1 [2] and that a smaller value indicates better score. As such, we say p dominates q if $p.A_i < q.A_i$ for at least one value of i and $p.A_i \leq q.A_i$ for $1 \leq i \leq d$. The skyline set $S \subseteq D$ consists of all objects in D which are not dominated by any other objects in D. We also have a monotonic ranking (or preference) function $P(\cdot)$ which is provided by the application domain and users will specify their preference by providing a set of weights $w = \{w_1, w_2, \ldots, w_d\}$, $0 \leq w_i \leq 1$. Given the set of weights, the user can easily define any monotonic ranking function as $P(\overrightarrow{w}, \overrightarrow{f(\cdot)}) =< \overrightarrow{w}, \overrightarrow{f(\cdot)} >$, i.e. the dot product of weight vector $\overrightarrow{w}(w_1, w_2, \ldots, w_d)$ and monotonic function vector $\overrightarrow{f(\cdot)}(f_1(\cdot), f_2(\cdot), \ldots, f_d(\cdot))$. $f_i(\cdot)$ can be any monotonic function on objects, such as linear function, product function or exponential function. As illustrated in 2-dimensional space, one simple linear ranking function $P = 0.9A_1 + 0.2A_2$ can be expressed as the dot product of the weight vector $\overrightarrow{w}(0.9, 0.2)$ and the monotonic function vector $\overrightarrow{f}(A_1, A_2)$.

[1]Note that since multiple settings of w can be represented by the same skyline object, this inference is only approximate.

[2]This can be obtained by mapping the attribute values of some application domain to a score from 0 to 1



(a) Data Space (b) Weight Space



(c) $R_4(p_2)$ on weight space

Figure 1: Example of Data Space and Weight Space

Given the above setting, our paper deals with two multi-dimensional spaces. First, we have the **data space**, which is the d-dimensional space that is formed from A_1, ..., A_d. Second, we have the **weight space**, which is another d-dimensional space formed from w_1,...,w_d, i.e. the i^{th} dimension of this space represents the weight w_i. Any point in the weight space thus corresponds to a particular setting of the weights. For any skyline object $p \in S$, we will use $R(p)$ to refer to the region in the weight space such that $\pi_w(D)[1] = p$ as long as w is within the region $R(p)$. Since the weight space is normalized to the unit range, we can treat the volume of $R(p)$ as a probability that p is the top object in any possible ordering. Figure 1 illustrates the data space and weight space for a set of skyline objects when the preference function is a simple dot linear product between the weights and the attribute values. As can be seen in Figure 1(a), p_1, ..., p_5 are all skyline objects since they do not dominate each other. However, if we look at the weight space in Figure 1(b), we can see only $R(p_1)$, $R(p_3)$ and $R(p_5)$ since based on the dot linear product preference function, p_2 and p_4 can never be ranked first regardless of the weight setting.

If we use $V(\cdot)$ as a function that calculates the volume of any given region, then the probability of p being a top object will be denoted as $V(R(p))$. To generalize this further, we will use $R_m(p)$ to denote the region in the weight space such that p is among the top-m objects when compared to other skyline objects. We are now ready to define a similarity measurement between two skyline objects p and q.

DEFINITION 2.1. $\mathcal{SIM}_m(p,q)$
Given $p, q \in S$ and m, we measure how well p can represent q by

$$\mathcal{SIM}_m(p,q) = V(R(p) \cap R_m(q))/V(R(p))$$

It is easy to see that $\mathcal{SIM}_m(p,q)$ is in fact the probability that q is within the top-m skyline objects whenever p is ranked first. Intuitively, we are saying that if p has orderings that are very similar to q, then $\mathcal{SIM}_m(p,q)$ will be high and thus p can represent q well. Note that $\mathcal{SIM}_m(p,q)$ is in fact not a metric since it is not symmetric and also does not follow triangular inequality. This however does not affect our k-partitioning clustering algorithm. Unlike most clustering applications in which members in the same cluster must be similar, our sole aim here is that the representatives of each cluster can represent its members accurately while other members in the cluster need not be similar to each other.

Given S, our aim is to select a k representative set K, such that $K \subseteq S$, $|K| = k$ and other non-representative skylines are somehow represented by K in terms of the ordering that they represent. Intuitively, K should satisfy two criteria. First, its $\bigcup_{p \in K} V(R(p))$ should cover a sufficiently large region of the weight space so that weight settings are covered as much as possible. Second, K should somehow represent other skylines that are not within K.

The first criteria is relatively easier to satisfy with the observation that $\{R(p) \cap R(q) = \emptyset, p, q \in S \wedge p \neq q\}$. Since there are no overlap between the regions, it is enough to ensure that $V(R(p))$ is sufficiently large for each $p \in K$ so that $\bigcup_{p \in K} V(R(p))$ is large. For the second criteria, we will propose a measure of goodness.

DEFINITION 2.2. $Quality(K,S)$

$$Quality(K,S) = \frac{\sum_{q \in S} \max_{p \in K} \mathcal{SIM}_m(p,q)}{|S|}$$

As can be seen, $Quality(K,S)$ is a goodness measure that is similar to those used in a k-partitioning algorithm, i.e. the average similarity between each skyline object and its best representative. Correspondingly, we define our order-based representative skyline problem as follow:

DEFINITION 2.3. *Order-based Representative Skylines*
Given S, p , m and threshold α, find a set of representative $K \subseteq S$ such that:
1. For each $p \in K$, $V(R(p)) \geq \alpha$.
2. $Quality(K,S)$ is maximized.

2.2 Problem Analysis

In order to find the order-based representative skylines, the beginning step is to calculate the volumes of $R(p)$ and $R_m(p)$ for each skyline object p. However, we will show the hardness of exact computation of $R(p)$ and $R_m(p)$ with their corresponding volumes. First, one important property about the $R(p)$ is presented in the following lemma.

LEMMA 2.1. *For any skyline object p, $R(p)$ is either empty or a convex polytope.*

PROOF. Based on linear programming theory, if p is the i^{th} object p_i in the skyline, the computation of $R(p_i)$ can be directly transformed to the satisfaction of the following inequations:

$$\begin{cases} P(\overrightarrow{w}, \overrightarrow{f(p_i)}) \leq P(\overrightarrow{w}, \overrightarrow{f(p_1)}) \\ ... \\ P(\overrightarrow{w}, \overrightarrow{f(p_i)}) \leq P(\overrightarrow{w}, \overrightarrow{f(p_{i-1})}) \\ P(\overrightarrow{w}, \overrightarrow{f(p_i)}) \leq P(\overrightarrow{w}, \overrightarrow{f(p_{i+1})}) \\ ... \\ P(\overrightarrow{w}, \overrightarrow{f(p_i)}) \leq P(\overrightarrow{w}, \overrightarrow{f(p_n)}) \\ w_i \in [0,1] \end{cases}$$

The above inequations are a set of linear constraints on the weight space, because each $P(\cdot)$ is the linear function with respect to weight vector $\overrightarrow{w}$ given the $\overrightarrow{f(\cdot)}$ and the attributes for each $p \in S$. Therefore, computing the $R((p))$ is equivalent to solving the feasible range of linear constraints. The boundary theory of linear programming [26] proves that each inequality specifies a half space in an n-dimensional Euclidean space, and their intersection is the set of all feasible values the variables can take. The region is either empty, unbounded, or a convex polytope. In our case, the region is either empty, or a convex polytope, because it is bounded by weight space with $w_i \in [0, 1]$. □

According to Lemma 2.1, these regions can be determined by computing their boundaries. Ideally, we first discover vertices of $R(p)$ or $R_m(p)$ for each skyline object p, and then derive $\mathcal{SIM}_m(p, q)$. However, the cost of this method is too expensive. For a convex polytope, there are at most $(u!)/(v!(u-v)!)$ vertices, where u is the number of inequations and v is number of variables [26]. Accordingly, the $R_m(p)$ can also be viewed as a union of all possible combinations of linear constraints that p is smaller than at least $|S| - m$ skyline objects. As illustrated in Figure 1(c), the shaded region, which is $R_4(p_2)$, is the union of two separate parts. This simple example shows that the computation of $R_m(p)$ is much more complicated than the computation of $R(p)$. Therefore, we conclude that finding the exact boundary of top region and top-m region is unrealistic.

3. RELATED WORK

The skyline operator was introduced into the database community by Borzsonyi *et al.*[3]. The efficiency was improved by Chomicki *et al.*[7] and Godfrey *et al.*[25] significantly by means of sorting. By exploiting index structures, the efficiency of skyline query processing can be further improved. Kossmann *et al.*[17] presented a nearest neighbor search algorithm and Papadias *et al.*[24] proposed a branch-and-bound algorithm (BBS). Both methods are based on R-tree structure [11]. This operator has been studied in the context of distributed systems [2], P2P networks [29, 30], parallel environment [31], data streams [27], microeconomic data analysis [19, 20, 34] and processing queries with minimum communication [32].

The problem of having too many skylines in high dimensional space were first highlighted by us in [33, 5, 4] and solutions were proposed in the form of *strong*, *frequent* and *k-dominant* skyline respectively. Subsequently, [21] proposed *representative skylines* where k representative skyline objects must be found such that they together dominate the most objects. From a ranking point of view, this ensures that the representatives will somehow not rank too low since the dominated objects will never rank higher than them with any weight settings. Next, *distance-based representative skylines* [28] grouped the skyline objects into k clusters based on Euclidean distance and the medoid of each cluster is selected as a representative skyline. Spatial proximity, however, does not necessary means similarity in ordering. Going back to our example in Figure 1, although p_2 and p_4 are spatially closer to p_3 when compared to p_1 and p_5 respectively, they are never ranked higher than 4^{th} whenever p_3 is the top object. Instead, p_2 can rank higher (up to 2^{nd}) when p_1 is the top object and likewise for p_4 whenever p_5 is the top object. Besides this, it is well known that the distance-based method can never avoid the curse of dimensionality, in the sense that the Euclidean distance of a given skyline object from its nearest and farthest neighbor tends to converge [1]. In contrast, an order-based method is robust to the increase in dimensionality, which is more suitable for high dimensional context. In view of the large number of proposals on alternate skyline definition, a general framework for doing so was developed in [35].

Preference discovery and mining is also related to our work. Kießling [16] modelled various preference constructors and integrates them into database systems. Based on the preference construction approach aforementioned, Jiang *et al.*[15] introduced the scenario of mining preferences using superior and inferior examples. Recently, Denis *et al.*[22] proposed a framework called *p-skylines* which enriches skylines with the notion of attribute importance. These works differ from ours in two ways. First, their main aim is to elicit the preference of categorical values within some categorical attribute domains. Second, they focus on finding unknown **atomic** preferences, i.e. an attribute is either more important, less important or incomparable to other attributes. Our work involves the concept of weighted attributes which can model tradeoffs between the attributes. For example, we can model the fact that a user is willing to take a notebook with a CPU that is 20% slower if 50% more memory is given.

Order information is well studied by the database and data mining communities. Rank aggregation [9] is an useful technique in web applications and other scenarios related to aggregating results from heterogeneous sources. Cohen *et al.* [8] developed an algorithm to learn a linear preference function. In this algorithm, feedbacks are iteratively given by users in the form of "p is preferred to q" and the weights are iteratively adjusted based on the feedbacks. Our approach differs in that we focus on skylines as a representative of orders and provide a hierarchical visualization framework to elicit the preference of users systematically[3].

4. METHODOLOGY

According to earlier analysis, finding the exact regions for $R(p)$ and $R_m(p)$ for all $p \in S$ can be very computationally intensive. Since we are only interested in $V(R(p))$ and $V(R_m(q) \cap R(p))$, we can adopt a sampling approach to estimate these values. This is done by performing a uniform sampling in the weight space and generating a set of weight settings W. For each $w \in W$, we find $\pi_w[i]$ for $1 \leq i \leq m$ and keep a count on the occurrences of the skyline objects. Once the sampling is complete, we can simply estimate $V(R(p))$ by $count(\{w|w \in W, \pi_w[1] = p\})/|W|$, i.e. the number of instances w in which p is ranked top and divide it by $|W|$. Likewise, $V(R_m(q) \cap R(p))$ is $count(\{w|w \in W, \pi_w[1] = p, \pi_w[i] = q, i \leq m\})/|W|$, the number of instances in which q is ranked among top-m whenever p is the top object and normalize it by $|W|$. Finally, we obtain the following formula according to the definition 2.1:

$$\mathcal{SIM}_m(p, q) = \frac{count(\{w|w \in W, \pi_w[1] = p, \pi_w[i] = q, i \leq m\})}{count(\{w|w \in W, \pi_w[1] = p\})}$$

[3]In many ways, our approach is similar to how we judge the results of a search engine; We conclude that the search ranking is useful if the first few results are good.

There are two remaining issues. First, we need to ensure that generating these samples is efficient. Second, we need to ensure that our estimation based on these samplings have certain accuracy. We will address these two issues in the next two subsections. Once these issues are resolved, we will then move on to present our clustering algorithm based on our measure of similarity.

4.1 Generating Samples

Instead of computing the ordering for individual samples, we conduct the sampling in batches and apply the TA algorithm [10] concurrently for all samples within the same batch. Assuming that the main memory can handle b samples and we want to have a total of s samples, then the TA algorithm will be applied $\lceil s/b \rceil$ times.

The sampling method is shown in Algorithm 1. For all the sample weight settings, it only needs to discover top-m skyline objects from the disk once using the TA algorithm. Here, m is set to be (number of skyline objects)/k based on the assumption that the skyline objects have uniform probability of appearing in the top-m list of any of the (eventual) k representative objects and thus setting m to this value ensures that each of the objects has a non-zero probability of appearing in the top-m list of one of the k representatives. This m can then be fixed for processing future batches of samples.

In order to perform TA algorithm, we further need to store d sorted lists in the disk. In τ_i, the skyline objects are sorted from the smallest to the largest based on the values on dimension i. Because the score function is monotonic, we perform sorted access and random access on d ranking lists to find the top-m skyline objects efficiently. According to these top-m lists, we can calculate the $V(R(p))$ and $V(R(p) \cap R_m(q))$ and derive $\mathcal{SIM}_m(p,q)$ for every $p, q \in S$.

Intuitively, the approximation of region computation has high precision based on random uniform sampling if the sampling size is sufficiently large. Thus, we approximately achieve region computation as well as derive $\mathcal{SIM}_m(p,q)$ according to definition 2.1. As in Line 12-15 in Algorithm 1, we only keep in memory the counting information of skyline objects which can be the top object. Since the skyline set could be too large to fit in the memory, this strategy greatly reduce the memory consumption. In addition, after performing the TA based algorithm, the batch of samples can be safely discarded to free up the memory for the next batch. Let the top object set T_0 satisfy $\{T_0|p$, if $p \in S$ and $V(R(p)) > 0\}$. Accordingly, T_α is defined as $\{T_\alpha|p$, if $p \in S$ and $V(R(p)) > \alpha\}$. Assume the skyline set size is n, Algorithm 1 utilizes $O(|T_0|n)$ instead of $O(n^2)$. $|T_0|$ is determined by the monotonic function, which is much smaller than n. Therefore, this improves the scalability of our algorithm. Besides the probability information, the regions of $p \in T_0$ defined below are incrementally updated in Line 14 based on samples. This information is critical for hierarchical processing in Section 5.

Definition 4.1. *Object Coverage*
Given $\{W|$weight setting $w \in W$ if $\pi_w[1] = p\}$, the coverage of p is the minimal bounding rectangle(MBR) of W on weight space.

The MBR for the object p is the minimal bounding rectangle that encloses all the $w \in W$ whenever $\pi_w[1] = p$. However, to determine a sufficient number of samples is challenging. The sample space is infinite and the definition of sufficiency is unclear. Before finding the k representative skylines, we first show what is the quantitative relationship between sampling size and sampling accuracy and how to calculate the sampling size s in Line 2 of Algorithm 1.

Algorithm 1: **SamplingTopM**

```
   Input: # representatives k and # samples b
   Output: 2d array SIM_m to store SIM_m(p,q)
1  m ⟵ # skyline objects/k
2  Calculate the required sampling size s
3  while s > 0 do
4      Generate next b random uniform samples W
5      s ⟵ s − b
       // TA based method for b samples
6      while score_i(m^th item on heap_i) > δ_i, i ∈ [1,w]
       do
7          Round-robin sorted access on τ_1, ..., τ_d
8          Update thresholds θ_1, ..., θ_b for each sample
9          Random access to get next skyline object p
10         if score_i(p)<score_i(m^th item on heap_i) then
11             Swap p with m^th item on heap_i
12     foreach skyline object q in top-m list when p is
       the top object do
13         if p ∉ SIM_m then new array SIM_m[p]
14         Update the region for R(p)
15         SIM_m[p][q]++
16 foreach skyline object p ∈ SIM_m do
17     Count[p] ⟵ count({w|w ∈ W, π_w[1] = p})
18     foreach skyline object q do
           // calculate the probablity
19         SIM_m[p][q] ⟵ SIM_m[p][q] / Count[p]
20 return SIM_m;
```

4.2 The analysis of sampling accuracy

The sampling size determines the tradeoff between accuracy and efficiency. Intuitively, we expect the approximations of $V(R(p))$ and $\mathcal{SIM}_m(p,q)$ to be close to the accurate values if the values are larger than certain thresholds. The constraint of $V(R(p))$ refers to definition 2.1. Furthermore, $\mathcal{SIM}_m(p,q)$ should be accurate if it is no smaller than the user-defined threshold β. Taking these two thresholds into consideration, we can derive the following bound for $V(R(p) \cap R_m(p))$:

$$V(R(p) \cap R_m(p)) = V(R(p)) \cdot \mathcal{SIM}_m(p,q) > \alpha\beta$$

Therefore, we will focus on the accuracy of $V(R(p) \cap R_m(p))$ to satisfy the required sampling quality. Next, we provide guidelines for the choice of sampling size using statistical analysis.

Let $V(R(p) \cap R_m(p))$ be the unknown value that we are trying to estimate. For simplicity, we utilize probability P representing $V(R(p) \cap R_m(p))$ to do the analysis. Then, the $\overline{P}$ stands for the complementary set of $V(R(p) \cap R_m(p))$. Assume that we have N samples and find that $X = \tilde{P}N$ of these samples satisfy q does not appear in the top-m lists when p is the top object. Given a sufficiently large number

of samples, we expect $\overline{P}$ to be close to $\tilde{P}$ as much as possible. Furthermore, to ensure sufficiently large coverage, P should be larger than or equal to $\alpha\beta$, which is equivalent to $\overline{P} \leq 1 - \alpha\beta$. We formally express the problem as follows.

PROBLEM 1. *Given thresholds α, β, confidence interval $1-\gamma$ and margin of error δ, how to determine the sampling size N to ensure*

$$Pr(\overline{P} \in [\tilde{P} - \delta, \tilde{P} + \delta]) > 1 - \gamma$$

when $\overline{P} \leq 1 - \alpha\beta$.

Obviously, we want both the interval size 2δ and the error probability γ to be as small as possible. Since the sampling process can be viewed as the Bernoulli Trials on the weight space, $X = \tilde{P}N$ satisfies a binomial distribution with parameters N and $\overline{P}$. Therefore, we can apply Chernoff bounds [12] to compute

$$\begin{aligned} & Pr(\overline{P} \notin [\tilde{P} - \delta, \tilde{P} + \delta]) \\ = \; & Pr(X < N\overline{P}(1 - \delta/\overline{P})) + Pr(X > N\overline{P}(1 + \delta/\overline{P})) \\ < \; & e^{-N\overline{P}(\delta/\overline{P})^2/2} + e^{-N\overline{P}(\delta/\overline{P})^2/3} \end{aligned}$$

The bound in above equation is meaningless if the value of $\overline{P}$ is unknown. A simple relaxation is based on the fact that $\overline{P} \leq 1 - \alpha\beta$, yielding

$$Pr(\overline{P} \notin [\tilde{P} - \delta, \tilde{P} + \delta]) < e^{-N\delta^2/2(1-\alpha\beta)} + e^{-N\delta^2/3(1-\alpha\beta)}$$

Setting $\gamma = e^{-N\delta^2/2(1-\alpha\beta)} + e^{-N\delta^2/3(1-\alpha\beta)}$, we obtain the tradeoff between these parameters. Given the requirements on α, β, γ and δ, the above equation calculates the minimum number of sampling size N to guarantee the sampling accuracy. As shown in Section 6, the reasonable sampling size is around below ten thousand, resulting in satisfactory response time.

4.3 Finding representative skylines

Since the $\mathcal{SIM}_m(p, q)$ has already been calculated approximately, the next step is to discover the order-based representative skylines. Our objective is to maximize the quality of the representative set K as well as cover a sufficiently large size of the weight space. In order to efficiently discover k representative skylines, we adopt the k-medoids clustering algorithm, as presented in Algorithm 2. By partitioning the skyline objects into groups, we choose the medoid as representative for each group and other skyline objects are assigned to the closest representative.

One noticeable difference of Algorithm 2 is the filtering method. As in line 3, it sifts out candidate set C as T_α. The default setting of α is $1/|S|$, the average volume of $R(p)$ for $p \in S$ on weight space. This is reasonable since the volume of representative skylines should be at least no worse than the average situation, otherwise the objects can be safely pruned. User also has the flexibility to adjust the threshold in order to achieve the tradeoff between the quality and the coverage of K. This is not only beneficial to finding better k representative skylines, but also reducing the candidate size, especially for skyline objects with skew top region sizes. After filtering, the rest is straightforward. The k-medoids clustering is derived from CLARANS [23], which randomly chooses k medoids and refines its quality by neighborhood searching. In detail, the *swapcost* in line 10 is the quality difference because of swapping. Line 13 guarantees that the cluster K is updated only if the new cluster K' has better quality. Finally, the clustering algorithm determines *numlocal* k-medoids sets with local best quality, and chooses the best of them as the final order-based k representative skylines.

Algorithm 2: **FilterClustering**

```
   Input: # representatives k,threshold α and SIM_m
   Output: k order-based representative skylines
 1 Candidate set C ⟵ ∅
 2 k-medoids set K ⟵ ∅
 3 foreach skyline objects p ∈ T_0 do
 4     if V(p) > α then C ⟵ C ∪ p
 5 bestquality ⟵ 0, bestrepresentative ⟵ ∅
 6 for i = 1 to numlocal do
 7     K ⟵ randomly choose k objects from |C|
 8     for j = 1 to maxneighbor do
 9         Randomly swap p from K and q from C − K
10         swapcost ⟵ ΔQuality(K, S)
11         if swapcost < 0 then
12             j ⟵ 1, update K
13     if Quality(K, S) > bestquality then
14         bestrepresentative ⟵ K
15         bestquality ⟵ Quality(K, S)
16 return bestrepresentative
```

5. ELICITING USERS' PREFERENCES

In this section, we further extend our work to support skyline browsing and visualization in order to elicit users' preference effectively and efficiently. In general, it is a hierarchical navigation approach to locate user's preferred region on the weight space. A visual interface is developed to support this exploration.

5.1 Hierarchical Browsing

Hierarchical browsing is an effective way to interact with the user. As shown in Algorithm 3, this process can be viewed as iterative refinements based on a combination of sampling and clustering. First, shown with the initial k representatives, the users will then select a subset of them as object/s of interested. Second, re-sampling is performed on the region covered by the subset and related clusters. This focused sampling will allow us to have more accurate sampling result on the area of interest. We first define the **cluster coverage** as follows.

DEFINITION 5.1. *Cluster Coverage*
Given a cluster c, the cluster coverage r_c of c on the weight space is the minimal bounding rectangle(MBR) that bounds the object coverage of all skyline objects in c.

Therefore, the area of interest is the MBR covering all the clusters generated by the selected skyline subset. The clustering algorithm will then be applied on the new samples so that the next level of k representatives can be found. This procedure will iterate until the user reaches the skyline object that is of interest to him/her or when $|T_0|$ is smaller than k. As shown in Line 5 of the algorithm, the hierarchical

process terminates and displays the final results to the user if one of the above two conditions is satisfied.

Algorithm 3: **HierarchicalBrowsing**

Input: $w, k, \alpha, \mathcal{SIM}_m$
$p \longleftarrow SamplingTopM(w)$
$K \longleftarrow FilterClustering(k, \alpha, \mathcal{SIM}_m)$
Output K to user
User chooses interesting subset H and sets k
while $H \neq \emptyset$ ***and*** $|T_0| \geq k$ **do**
  $sampleregion \longleftarrow \emptyset$
  Candidate set $C \longleftarrow \emptyset$
  foreach *object* $p \in H$ **do**
    Calculate cluster region r_c from the cluster c with medoid p
    Update *sampleregion* covering the r_c
  Update C as all the objects in the cluster c with medoid $p \in H$
  `// sampling on` *sampleregion*
  $p \longleftarrow SamplingTopM(m, w)$
  $K \longleftarrow FilterClustering(k, \alpha, p)$
  Output K to user
  User chooses interesting subset H and sets k
Output final set of skyline objects user preferred

5.2 Visualization

To support our hierarchical browsing process, we provide a visualization tool to ensure that users can easily see the difference between the representatives and select the representatives that are of interest to them.

Parallel coordinates [13] is a common way of visualizing high-dimensional geometry and analyzing multivariate data. To show a set of objects in an d-dimensional space, this technique represents data dimensions as d parallel lines spaced equally. A data object in d-dimensional space is represented as a polyline linking n vertices on each axes. The i^{th} vertex is mapped to position on i^{th} axis proportional to its value for that dimension. For our purpose, each of these axis represents a dimension in the weight space and the users can thus indirectly indicate their preferred weight setting by selecting the clusters based on the visualization. For each cluster, we take the average values of the samples along each dimension and plot a line that goes through these averages for each dimension. Users can estimate the average weight setting for each dimension by looking at this line. Generally, the range of different clusters overlapping and crossing each other, which renders the graphic representation unclear. Instead, we approximate display the cluster coverage. For the cluster c, the weight setting w belongs to it if $\pi_w[1] = p \wedge p \in c$. Let the mean value of all the weight settings belongs to cluster c on the i^{th} dimension be $\mu_i(c)$, and the standard deviation of them be $\sigma_i(c)$. To ameliorate the visualization, we restrict the range as $[\mu_i(c) - \sigma_i(c), \mu_i(c) + \sigma_i(c)]$ on the i^{th} dimension to control the size of the cluster c.

Figure 2 gives an example of our visualization technique. According to the definition of weight space, all the dimensions are within the range of 0.0 to 1.0. In the diagram, three clusters are represented with three representatives: rep-1, rep-2 and rep-3. Each cluster is visualized as a polygon. In each dimension, the cluster region is restricted by the upper bound and the lower bound respectively. Furthermore, the polyline in the middle of each region is shown for users to estimate the average weight settings of the cluster in each dimension. In addition, the values of each of the representative skyline in the data space are also presented for users to link their preferences back to the actual domain that is familiar to them. To distinguish different clusters, the polyline and representative label belonging to the same cluster are colored similarly, while the colors are different between clusters. This framework supports highlighting cluster as well. For instance, the cluster with representative rep-1 is highlighted with red color in Figure 2.

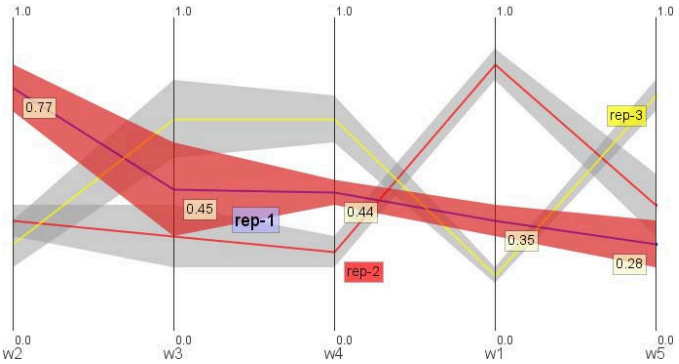


Figure 2: Visualization example

Ordering of Axes: Another simple but powerful feature of our visualization tool is that it supports dynamic ordering of the axes based on the selected cluster c. The dimensions are arranged from left to right following the order $w_1, w_2, \ldots, w_d$ if $u_1(c) \geq u_2(c) \geq \ldots \geq u_d(c)$. By looking at the order of these dimensions, users can quickly assess the strength of the cluster by looking at the relative ranking of the dimensions and compare these ranking against what they preferred.

Furthermore, the gradient of the polyline after the ordering presents a good indication of the tradeoff between the attributes for a particular cluster. A steep gradient indicates that the tradeoff between the attributes is high while a gentle gradient indicates that the importance between the attributes are almost the same. For instance, the dimensions of Figure 2 are reordered to be $\{w_2, w_3, w_4, w_1, w_5\}$. The underlying meaning is that this cluster of skyline objects ranks high mainly because of the dominance on the attribute w_2. Moreover, the steep gradient of this cluster demonstrates that the quantities on attributes A_1 and A_5 must drop substantially for the sake of sustaining the strength on A_2.

Alternatively, users will be also allowed to rank the dimensions themselves. Once they ordered the dimensions, they can identify clusters of interest to them by looking for polylines that are approximately decreasing from left to right. Among all those that are decreasing, they can also assess the tradeoff by looking at the gradients.

6. EXPERIMENTS

We now present the experimental study to evaluate order-based representative skylines. For simplicity, we refer to our algorithm as *SampleClus*, while refer to the best algorithm in distance-based representative skylines [28] as *I-greedy*. In Section 6.1, the proposed algorithm is measured with respect to the efficiency as well as effectiveness on synthetic datasets. Section 6.2 further illustrates its performance on

the real NBA dataset [4]. At last, the effect of different monotonic functions and the process of hierarchical elicitation are evaluated in Section 6.3. All experiments are executed on the Windows operating system with Intel Core-2 Duo processor and 4 GB RAM.

6.1 Synthetic Data

The synthetic datasets are created using the anti-correlated distribution according to the classical method [3]. Every attribute on each dimension is normalized to [0,1]. Table 1 shows the range and default values (in bold) of the parameters. In each experiment, we adjust a single parameter while keeping the rest at their default values. Note that the confidence interval $1-\gamma$ and margin of error δ are two variables for controlling suitable sampling size. Due to the space constraint, other two parameters α and β are fixed in our experimental settings. The default value for α is 0.01 as we expect the representative skyline object covers at least one percent of the weight space. The default value of β is set to be $1/k$, i.e. $\mathcal{SIM}_m(p,q) \geq \beta = 0.1$, since the closest representative should be better than the average case to represent the non-representative objects. The evaluation is based on the dot linear product preference function.

Table 1: Parameter settings

Parameter	Range
γ	0.1,**0.2**,0.3,0.4
δ	0.02,0.03,**0.04**,0.05
Dimensionality	2,**3**,4,5
k	4,6,8,**10**,12
Data Size(100K)	2,4,6,8,**10**

Table 2: Varying γ

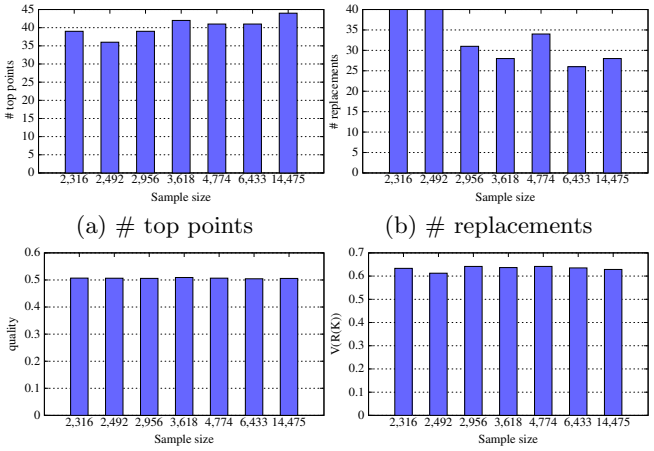
γ	0.1	0.2	0.3	0.4
Sampling Size	4,774	3,618	2,956	2,492
Sampling time(ms)	1,842	1,409	1,133	953

Table 3: Varying δ

δ	0.02	0.03	0.04	0.05
Sampling Size	14,475	6,433	3, 618	2,316
Sampling time(ms)	5,558	2,513	1,409	887

We measure the performance of the algorithm in seven aspects. The first five refer to # top objects, # replacements, $Quality(K,S)$, $V(R(K))$, and $Er(K,S)$, which evaluate the effectiveness of the algorithm. The first one reflects how many distinct skyline objects appear as top objects in the samples. The # replacements is utilized to evaluate the robustness of the clustering algorithm, which records the number of objects varying from one cluster to another due to the alteration of sampling size. Recall that the $Quality(K,S)$ is described in definition 2.1. The $V(R(K))$ is the summation of all $V(R(p))$ as long as $p \in K$ since they are mutually exclusive. For $Er(K,S)$, it indicates the representative error to measure the distance between the representative skylines and the other skyline objects [28]. The remaining two aspects, # IO and CPU time, assess the efficiency of the algorithm. # IO consists of two parts, the random access times (RA) and the sorted access times (SA). Moreover, CPU time is also divided into sampling time and clustering time for better understanding the performance of our *SampleClus* algorithm. The breakdown of execution time provides a deeper and clearer view of the experimental result. Furthermore, due to the random nature of the sampling output, we repeat each experiment ten times and report the average measurements.

We first investigate how the confidence interval $1-\gamma$ and margin of error δ affect the performance of *SampleClus*. Since the α and the β are fixed, the sampling size N is determined by these two parameters. Table 2 shows how sampling size varies as γ changes from 0.1 to 0.4 while fixing $\delta = 0.04$. On the other hand, by increasing δ from 0.02 to 0.05 with $\gamma = 0.2$, we derive the sampling size in Table 3. Because the sampling quality is directly related to the sampling size, we continue the analysis based on the sampling size. To begin with, we generate the initial k-partitioning clusters using 4613 samples, which is the mean of all the sampling sizes in Table 2 and 3. Additionally, by varying the sampling size, we record # top objects, # replacements, $Quality(K,S)$ and $V(R(K))$, and display them in Figure 3. Generally, the trend of # top objects suggests that it increases with growing sampling size. However, the increase ratio tends to converge to the real # top objects, which is larger at the beginning while smaller at the end. Concerning the # replacements, it is large when the sampling size is small but tends to be stable when sampling size is large enough. Most importantly, the $Quality(K,S)$ and $V(R(K))$ of the clustering results demonstrate *SampleClus*'s robustness although the change in sampling size is noticeable. The *stdev* of $Quality(K,S)$ is 0.001 and that of $V(R(K))$ is 0.01. Thus, the experiments show that sampling is a practical approach to solve this problem.



(a) # top points (b) # replacements

(c) Quality(K,S) (d) V(R(K))

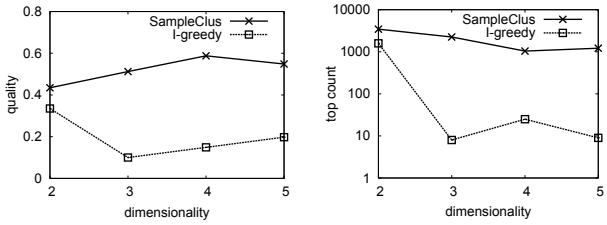
Figure 3: Robustness vs. sampling size

Furthermore, the sampling time is linear related to the sampling size as shown in Tables 2 and 3, since finding the top-m skyline objects for each sample almost costs the same amount of time. Taking both the clustering robustness and sampling time into consideration, we conclude that moderate size of samples is enough for good clustering outputs. Based on this observation, we choose $\gamma = 0.2$ and $\delta = 0.04$, which determine the sampling size to be 3618.

Figure 4 shows the comparison between *SampleClus* and *I-greedy* with respect to dimensionality. Note that we generate ten sample sets other than the one used in *SampleClus* to test the representative skylines of two algorithms. The figure suggests that *SampleClus* is superior to *I-greedy* both

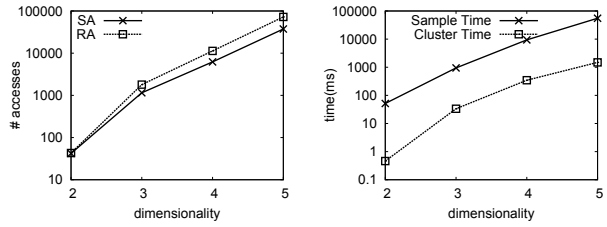
[4]http://www.databasebasketball.com

for $Quality(K,S)$ and $V(R(K))$ in any dimensionality. The $V(R(K))$ is multiplied by the sampling size for clearer display. The closeness of the two algorithms in two dimensional cases is because the number of skyline objects is 57, which is in the same order of magnitude as the number of representatives. Other than this, the distance-based representative skylines can hardly represent the order information as analyzed in the Section 3. Furthermore, the distance based metric is sensitive to the dimensionality. The goal of *I-greedy* algorithm is to minimize the $Er(K,S)$. Accordingly, we define a relative representative error $NormEr(K,S)$ as $Er(K,S)/\sqrt{d}$, where $\sqrt{d}$ is the maximal possible distance between two objects in d dimensional normalized space. By varying dimensionality from 2 to 5, as shown in Table 4, $Er(K,S)$ as well as $NormEr(K,S)$ increases along with the rise in dimensionality. It suggests that this goodness function deteriorates in high dimensional cases.



(a) Quality(K,S) (b) V(R(K))· sample size

Figure 4: Effectiveness vs. dimensionality



(a) #IO (b) CPU time

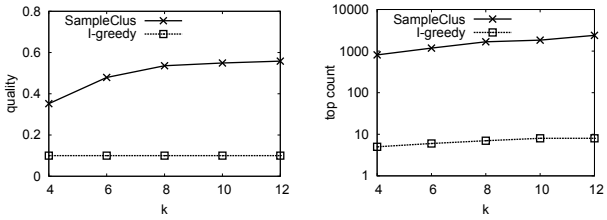
Figure 5: Efficiency vs. dimensionality

Figure 5 shows the efficiency measures as a function of dimensionality. As dimensionality varies from 2 to 5, the $|S|$ increases dramatically because of the property of anti-correlated distribution. The corresponding skyline sizes equal to 57, 990, 7745, 36290 for dimensionalities 2 to 5 respectively. Therefore, both # IO and CPU time rise linearly with respect to dimensionality.

Table 4: The relative representative error

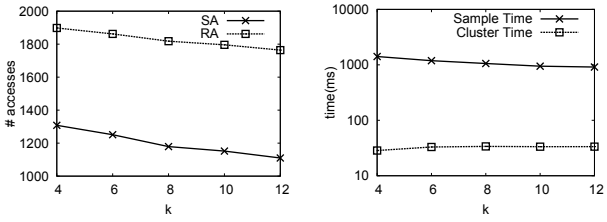
Dimensionality	2	3	4	5
$Er(K,S)$	0.09	0.39	0.64	0.86
$NormEr(K,S)$	0.06	0.23	0.32	0.38

Next, we vary the number of representatives k to explain how this parameter affects the effectiveness of our algorithm. The $V(R(K))$ is multiplied by the sampling size for clearer display. As shown in Figure 6, the $Quality(K,S)$ and $V(R(K))$ of *I-greedy* almost remain constant as the number of representatives increases. For *SampleClus*, the $Quality(K,S)$ and $V(R(K))$ are always greater than those of *I-greedy*, and increase as more representatives are returned.



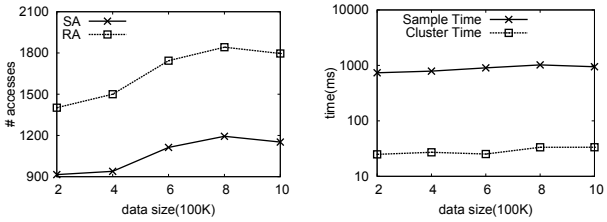
(a) Quality(K,S) (b) V(R(K))· sample size

Figure 6: Effectiveness vs. k



(a) #IO (b) CPU time

Figure 7: Efficiency vs. k



(a) #IO (b) CPU time

Figure 8: Efficiency vs. cardinality

In Figure 7, we present the effect of k on the efficiency measurements. As m equals to $|S|/k$, when the skyline set is fixed, m decreases along with the increase of k. Therefore, both of the random access times and sorted access times decrease accordingly. Similarly, we need to discover smaller top-m skyline list for each sample, so the sampling time reduces since the sampling size keeps invariable. On the other hand, the search space enlarges with respect to k, leading to the growing of the clustering time.

The last set of experiments focuses on the scalability of our algorithm as the function of cardinality. Although the cardinality of the dataset increases, the related skyline sizes are 773, 808, 936, 1101, 990 for cardinality 200K to 1M. Figure 8 presents the result. The performance does not show any significant changes since the major factor is the skyline size, but not the dataset cardinality. The trend of the curve is proportional to the number of skyline objects.

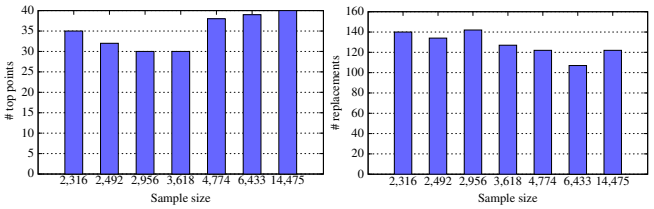
6.2 Real Data

In this section, we report results of experiments performed on the NBA dataset. NBA includes 16399 nine-dimensional objects. We denote each object as $p(A_1, A_2, \ldots, A_9)$, representing the regular season performance of a player from 1973-2008 on nine attributes: points per game (pts), rebounds per game (reb), assists per game (ast), steals per game (stl), blocks per game (blk), assists to turnovers (a/t), field goal percentage (fgp), free throw percentage (ftp) and three points percentage (tpp). The skyline set of NBA consists of 1024 players. Since the dataset's properties are fixed, we adjust γ, δ and k to measure the performance.

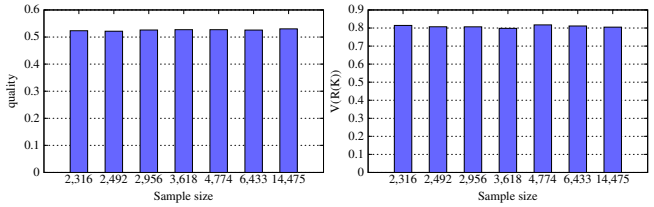
First, we show the quality of the results as a function of γ and δ in Figure 9. Following the same setting of γ and δ, the derived sampling size is the same as that of the synthetic data. The values of # top objects, # replacements, $Quality(K,S)$ and $V(R(K))$ with respect to sampling size are shown in figure 9. Although the robustness properties are similar, there exist several distinctions due to the correlations between NBA attributes. As such, the # top points is fewer and the # replacements becomes larger. Furthermore, the region sizes between different skyline objects are skew, resulting in better $Quality(K,S)$ and larger $V(R(K))$ when compared to these measurements for the synthetic data.

Table 5: Sampling time vs. γ and δ

γ	0.1	0.2	0.3	0.4
Sampling time(ms)	2,782	2,106	1,717	1,420
δ	0.02	0.03	0.04	0.05
Sampling time(ms)	8,343	3,679	2,106	1,340



(a) # top points (b) # replacements



(c) Quality(K,S) (d) V(R(K))

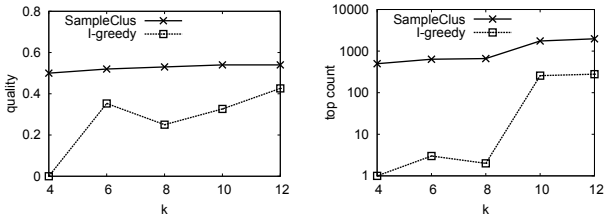
Figure 9: Robustness vs. sampling size

Figure 10 displays the relationship between k and the effectiveness of the representatives. The $V(R(K))$ is multiplied by the sampling size for clearer display. The *I-greedy* algorithm exerts no explicit relationship with the change of k. On the other hand, the order-based representative skylines present better $Quality(K,S)$ as well as $V(R(K))$ in comparison to distance-based representative skylines. Since the NBA dataset has correlated character, the gain of the two measures in *SampleClus* are not so significant by adding more representatives.

Figure 11 shows the relationship between k and the efficiency of the representatives. When k varies from 4 to 12, the values of m are 256, 171, 128, 102, 85 respectively. Consequently, except for clustering time in proportion to k, the random access times, sorted access times and sampling time decrease as k increases.

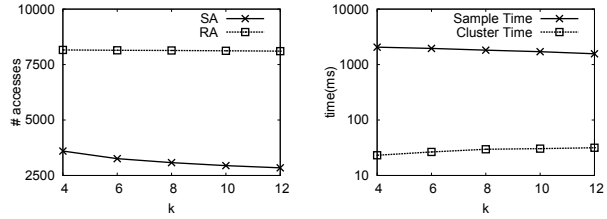
Table 6: The preference functions

	Attributes
$\overrightarrow{f_1(\cdot)}$	$(pts, \sqrt{reb}, ast^2, stl^2, \sqrt{blk}, a/t^2, fgp, ftp, tpp)$
$\overrightarrow{f_2(\cdot)}$	$(pts, reb^2, ast, stl, blk^2, a/t, fgp, \sqrt{ftp}, \sqrt{tpp})$
$\overrightarrow{f_3(\cdot)}$	$(pts, reb, ast, stl, blk, a/t, fgp, ftp, tpp)$



(a) Quality(K,S) (b) V(R(K))· sample size

Figure 10: Effectiveness vs. k



(a) #IO (b) CPU time

Figure 11: Efficiency vs. k

Table 7: The $\overrightarrow{f_1(\cdot)}$ representatives

Player ID	pts	reb	ast	stl	blk	a/t	fgp	ftp	tpp
2006_Nash	18	4	**12**	0.8	0.1	3	.53	.90	.45
1975_Jabbar	28	17	**5**	1.5	4.1	0.0	.53	.70	.00
1987_Bird	30	9	**6**	1.6	0.8	2.2	.53	.92	.41
1987_Jordan	35	5	**6**	3.2	1.6	1.9	.54	.84	.13
1991_Stockton	16	3	**14**	3.0	0.3	3.9	.48	.84	.41

Table 8: The $\overrightarrow{f_2(\cdot)}$ representatives

Player ID	pts	reb	ast	stl	blk	a/t	fgp	ftp	tpp
1980_Gilmore	18	10	**2**	0.6	2.4	0.7	.67	.70	.00
1975_Jabbar	28	17	**5**	1.5	4.1	0.0	.53	.70	.00
1989_Ewing	29	11	**2**	1.0	4.0	0.7	.55	.77	.00
1986_Jordan	37	5	**5**	2.9	1.5	1.4	.48	.86	.18
1974_Mcadoo	35	14	**2**	1.1	2.1	0.0	.51	.81	.00

Table 9: The distance-based representatives

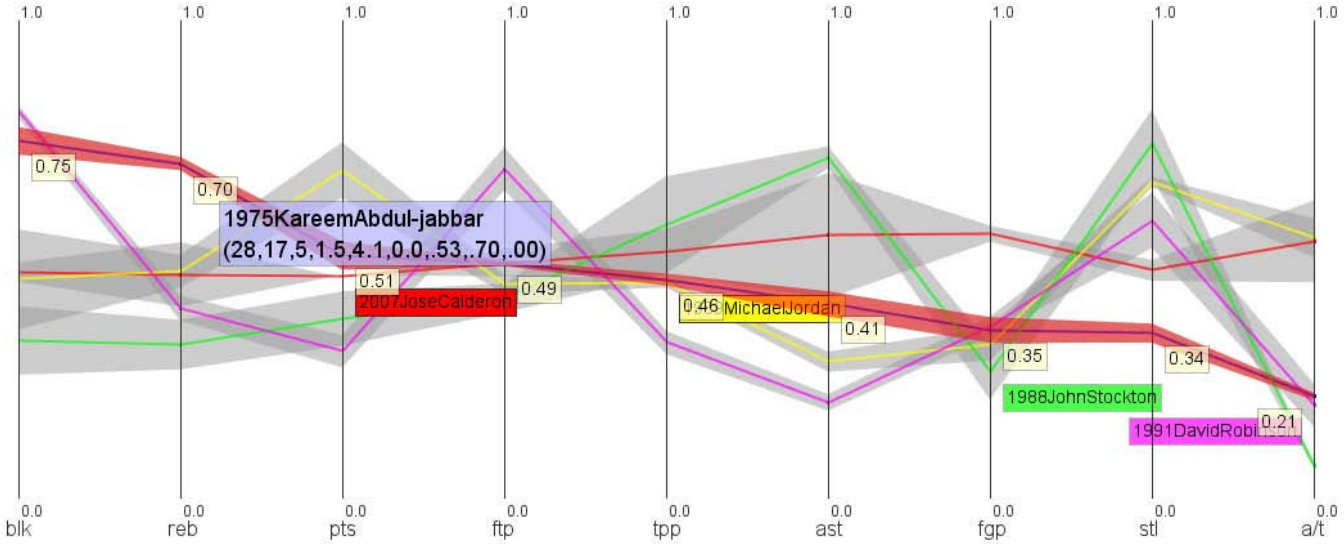
Player ID	pts	reb	ast	stl	blk	a/t	fgp	ftp	tpp
1989_Bogues	11	3	**9**	1.3	0.0	5.1	.48	.89	.19
1997_Rodman	5	15	**3**	0.6	0.2	1.6	.43	.55	.17
2003_Wallace	17	7	**3**	0.8	1.6	1.3	.44	.74	.34
2008_ Diener	4	2	**2**	0.5	0.1	5.8	.41	.80	.39
1986_Jordan	37	5	**5**	2.9	1.5	1.4	.48	.86	.18

6.3 Case Study of Preference Elicitation

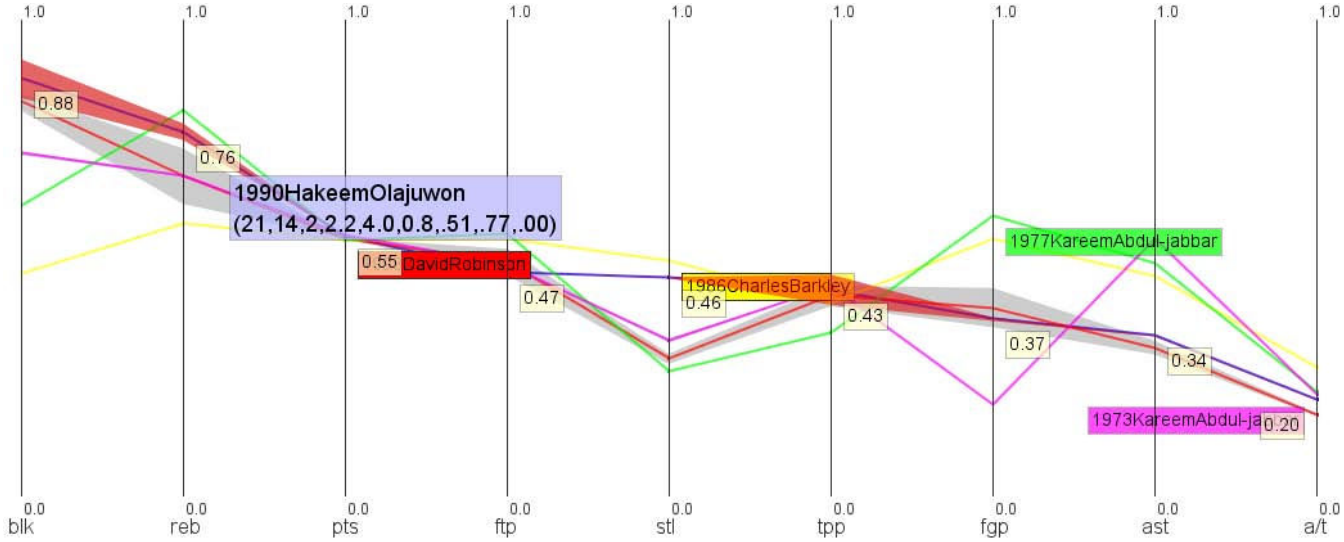
In this section, we further investigate the effect of preference function and the process of hierarchical browsing. Both these two factors exert an important influence on the outcome of preference elicitation. The experiments are conducted on the NBA dataset.

To begin with, we test how different monotonic functions affect the result. Unlike the distance-based representative skylines, the order-based representative skylines could vary to reflect the underlying interest of different users. Therefore, the user-defined function format could help him/her to faster elicit preferences.

We illustrate three different monotonic functions in Table 6 to show the distinct perspectives on the NBA dataset. For the function $\overrightarrow{f_1(\cdot)}$, the user favors players who are compa-



(a) First level visualization



(b) Second level visualization

Figure 12: Example of Hierarchical Browsing

rable in attributes ast, stl and a/t, while $\overrightarrow{f_2(\cdot)}$ could be a good choice if the user prefers players with better reb and blk. Comparing between Table 7 and 8, the five order-based representative skylines of $\overrightarrow{f_1(\cdot)}$ and $\overrightarrow{f_2(\cdot)}$ are of noticeable distinction. The former contains good assisters such as Nash and Stockton, while the latter includes outstanding defenders: Gilmore, Ewing and Macdoo. Moreover, taking ast for instance, the average ast of representatives in Table 7 are much higher than that in Table 8. Note that the output changes according to monotonic function is totally different from ranking based on specific function. The order-based representative skylines achieve a tradeoff between accuracy and heterogeneity, so the all-round players have the high probability to be selected as the representatives, such as Jordan and Jabbar. Besides comprehending the overall situation of the skyline set, users are likely to find desired objects as well. However, the results of distance-based representative skylines, as shown in Table 9, are less satisfactory. Although close to other skyline objects in Euclidean distance, most of the representatives themselves are not quite important. Furthermore, the result is fixed and unable to express the difference between the preference functions.

As displayed above, the tabular view of result is not intuitive especially for high dimensional case. We thus visualize the process of hierarchical browsing of $\overrightarrow{f_3(\cdot)}$ using the approach presented in Section 5. Since we adopt the linear function, the five representative players are averagely excellent. In Figure 12(a), the axes are ordered according to highlighted representative Jabbar, whose strengths are blk and reb. Also, the set of representatives are well separated and covering large area on the weight space. For example, the representative Stockton dominates distinct region comparing with Jabbar, which is reasonable because they are totally different kinds of players. Following the highlight representative, the re-sampling is performed and five new representatives are shown in Figure 12(b). These represen-

tatives are all excellent defenders as the Jabbar in higher level, nicely following the interest of the user. Note that one object represents one regular season record of certain player, so Jabbar appears twice in the new representative skylines with the records in 1973 and 1977 respectively. Furthermore, the ordering of attributes in the second figure is very close to that in the first one, suggesting that the new representatives have the similar strength as Jabbar. In summary, the hierarchical browsing approach enables users to drill down to the preferred region effectively, especially with the help of our visualization tool.

7. CONCLUSION

In this paper, we have introduced the order-based representative skylines, a novel concept that integrates the discovery of representatives with order preference. Unlike previous work, we brought the preference function back into the picture when determining representative skylines in order to elicit the preference. Moreover, a hierarchical sampling-clustering framework was developed based on the new notion. To further consolidate this interesting framework, we provided visualized view to guide the user's refinement of the result. The outcomes from an experimental study demonstrated that our order-based representative skylines can provide more informative views of data.

As for future research, we expect to extend our approach to improve the usability of the database system inspired by Jagadish *et al.*[14]. By interacting with the representative tuples and the visualization tool, users could explore and perceive the semantic of the data in a more sophisticated way. However, integrating the ordering with the tuple semantics remains challenging. Furthermore, how to improve the efficiency to handler huge dataset instead of skyline set is definitely the other interesting area of future work.

8. REFERENCES

[1] C. Aggarwal and P. Yu. Redefining clustering for high-dimensional applications. In *TKDE*, pages 210–225, 2002.

[2] W. Balke, U. Gĺźntzer, and J. Zheng. Efficient distributed skylining for web information systems. In *EDBT*, pages 256–273, 2004.

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

[4] C. Chan, H. Jagadish, K. Tan, A. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *SIGMOD*, pages 503–514, 2006.

[5] C. Chan, H. Jagadish, K. Tan, A. Tung, and Z. Zhang. On high dimensional skylines. *LECTURE NOTES IN COMPUTER SCIENCE*, 3896:478–495, 2006.

[6] J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

[7] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *ICDE*, pages 717–728, 2003.

[8] W. Cohen, R. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, pages 243–270, 1998.

[9] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.

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

[11] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.

[12] T. Hagerup and C. Rüb. A guided tour of chernoff bounds. *Inf. Process. Lett.*, 33(6):305–308, 1990.

[13] A. Inselberg. Lecture notes parallel coordinates. 1999.

[14] H. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *SIGMOD*, pages 13–24, 2007.

[15] B. Jiang, J. Pei, X. Lin, D. Cheung, and J. Han. Mining preferences from superior and inferior examples. In *KDD*, pages 390–398, 2008.

[16] W. Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.

[17] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, pages 275–286, 2002.

[18] K. Lee, B. Zheng, H. Li, and W. Lee. Approaching the skyline in z order. In *VLDB*, pages 279–290, 2007.

[19] C. Li, B. Ooi, A. Tung, and S. Wang. Dada: a data cube for dominant relationship analysis. In *SIGMOD*, pages 659–670, 2006.

[20] C. Li, A. Tung, W. Jin, and M. Ester. On dominating your neighborhood profitably. In *VLDB*, pages 818–829, 2007.

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

[22] D. Mindolin and J. Chomicki. Discovering relative importance of skyline attributes. *PVLDB*, pages 610–621, 2009.

[23] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *VLDB*, pages 144–155, 1994.

[24] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, pages 467–478, 2003.

[25] P. Ryan, R. Shipley, and J. Gryz. Maximal vector computation in large data sets. In *VLDB*, pages 229–240, 2005.

[26] R. Saigal. *Linear Programming: A Modern Integrated Analysis*. Springer, 1995.

[27] N. Sarkas, G. Das, N. Koudas, and A. Tung. Categorical skylines for streaming data. In *SIGMOD*, pages 239–250, 2008.

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

[29] S. Wang, B. Ooi, and A. Tung. Efficient skyline query processing on peer-to-peer networks. pages 1126–1135, 2007.

[30] S. Wang, Q. Vu, B. Ooi, A. Tung, and L. Xu. Skyframe: a framework for skyline query processing in peer-to-peer systems. *The VLDB Journal*, pages 345–362, 2009.

[31] P. Wu, C. Zhang, Y. Feng, B. Zhao, D. Agrawal, and A. Abbadi. Parallelizing skyline queries for scalable distribution. In *EDBT*, pages 112–130, 2006.

[32] Z. Zhang, R. Cheng, D. Papadias, and A. Tung. Minimizing the communication cost for continuous skyline maintenance. In *SIGMOD*, pages 495–508, 2009.

[33] Z. Zhang, X. Guo, H. Lu, A. Tung, and N. Wang. Discovering strong skyline points in high dimensional spaces. In *CIKM*, pages 247–248, 2005.

[34] Z. Zhang, L. Lakshmanan, and A. Tung. On domination game analysis for microeconomic data mining. *TKDD*, pages 1–27, 2009.

[35] Z. Zhang, H. Lu, B. Ooi, and A. Tung. Understanding the meaning of a shifted sky: a general framework on extending skyline query. *The VLDB Journal*, pages 1–21, 2009.