Approaching the Efficient Frontier: Cooperative Database Retrieval Using High-Dimensional Skylines

Wolf-Tilo Balke[1], Jason Xin Zheng[1], Ulrich Güntzer[2]

[1] Electrical Engineering & Computer Science,
University of California, Berkeley, CA 94720, USA
{balke, xzheng}@eecs.berkeley.edu
[2] Institut für Informatik,
Universität Tübingen, 72076 Tübingen, Germany
guentzer@informatik.uni-tuebingen.de

Abstract. Cooperative database retrieval is a challenging problem: top k retrieval delivers manageable results only when a suitable compensation function (e.g. a weighted mean) is explicitly given. On the other hand skyline queries offer intuitive querying to users, but result set sizes grow exponentially and hence can easily exceed manageable levels. We show how to combine the advantages of skyline queries and top k retrieval in an interactive query processing scheme using user feedback on a manageable, representative sample of the skyline set to derive most adequate weightings for subsequent focused top k retrieval. Hence, each user's information needs are conveniently and intuitively obtained, and only a limited set of best matching objects is returned. We will demonstrate our scheme's efficient performance, manageable result sizes, and representativeness of the skyline. We will also show how to effectively estimate users' compensation functions using their feedback. Our approach thus paves the way to intuitive and efficient cooperative retrieval with vague query predicates.

1 Introduction

In today's information systems instead of just retrieving all objects from databases that *exactly match* a user's query, often a set of objects *best matching* a set of query predicates has to be retrieved. To cater for this retrieval model, objects in the underlying database(s) are assigned a degree of match with respect to each 'soft' query predicate. Based on this paradigm cooperative information systems retrieve best compromises between (often mutually unsatisfiable) user needs by gradually relaxing soft constraints, where traditional SQL-based systems would just return empty result sets. Thus cooperative systems can even efficiently process overspecified queries and a tedious manual query refinement process is avoided. Consider a short example:

Example: A researcher from the Bay Area is planning a conference trip to New York. The basic 'hard' constraints for the trip are given by the dates and location of the conference. All necessary characteristics for e.g. booking a suitable flight can be evaluated relative to these constraints. For instance the flight has to arrive before the conference starts, but preferably close to the start date. Similarly the flight has to depart from a Bay Area airport and arrive in or at least close to New York. Usually the price should be minimized. Thus we have a mix of hard and soft constraints.

Retrieving best matches for such requests containing vague predicates is difficult: top k retrieval techniques efficiently answer vague queries, if adequate weightings of a compensation function (the 'utility function') are specified. However, expressing preferences by numerical weightings is neither intuitive, nor sensible without knowing the relationships between query predicates as reflected by the underlying database instance. In contrast skyline queries do offer intuitive querying by just specifying a user's basic query predicates, not exact weightings. Still, skylines guarantee to deliver *all* best objects with respect to *all* query predicates. But this advantage comes at a price: skyline sizes are known to grow exponentially with the number of query predicates and thus can exceed manageable levels already for fairly simple queries.

In this paper we propose to combine the advantages of intuitive skyline queries and manageable top k answer sets for cooperative retrieval systems by introducing an interactive feedback step presenting a representative sample of the (high-dimensional) skyline to users and evaluating their feedback to derive adequate weightings for subsequent focused top k retrieval. Hence, each user's information needs are conveniently and intuitively obtained, and only a limited set of best matching objects is retrieved. However, the huge size of skyline sets and the necessary time for their calculation remains an obstacle to efficient sampling for getting user feedback. Therefore we propose a novel sampling scheme to give users a first impression of the optimal objects in the database that is *representative* of the skyline set, *manageable* in size and *efficient* to compute, without computing the actual skyline. We will prove these characteristics and show how to subsequently estimate a user's compensation functions by evaluating feedback on objects in the sample. Our innovative approach promises to overcome the drawbacks of today's cooperative retrieval systems by utilizing the positive aspects of skyline queries in databases efficiently for the first time.

2 Intuitive Querying vs. Manageable Results

Cooperative retrieval systems generally feature a query model that allows the combination of 'hard' constraints and 'soft' constraints like shown in e.g. [14]. Since hard constraints can be quite efficiently evaluated using SQL, our sampling scheme will focus on the processing of soft constraints only. We will use a numerical query model, where each soft predicate of the query is evaluated by assigning a score in [0, 1] to each database object expressing the objects degree of match with respect to the predicate. Database objects can thus be understood as points in $[0, 1]^n$ with n as the number of soft constraints in a query. We assume that attribute values for soft predicates can always be mapped to numerical scores. For instance for our traveling researcher a flight's arrival time/date can be scored using the relative difference to the conference start, centered around the best possible arrival time. The choice of an airport in the Bay Area can similarly either be achieved by e.g. using the relative distance to our traveler's home for direct scoring, or assigning relative scores for an explicit statement of our traveler's preference as shown in [14].

2.1 Related Work in Cooperative Retrieval

The idea of extending exact-match querying frameworks by *cooperative retrieval* is generally based on the principle of query relaxation (see e.g. [13]). The idea of cooperation arises naturally due to necessary tedious query refinements in the case of empty result sets (overspecified query) or a flood of result objects (under-specified queries). Research focuses on two paradigms: Top k retrieval and skyline queries.

By aggregating all predicate scores with a single compensation function, top k retrieval (see e.g. [9], [7], and [5]) provides a simple, direct way to cooperative retrieval. Generally any monotonic function (like an average or weighted sum) to aggregate the basic scores into a single value (often called utility) can be used. The basic idea is to compensate between different aspects of objects, i.e. a good object in one predicate can afford to be worse in another. Thus objects become comparable by their respective utility to the individual user and the k best matching objects can be returned. However, expressing information needs a-priori as compensation functions is usually not very intuitive leading to a limited effectiveness of top k retrieval.

Skyline queries present a more intuitive paradigm: users only specify all important predicates for a query. The skyline then is the set of all objects that are not dominated with respect to *all* dimensions. This notion is well known in economical applications as Pareto-optimality and several basic algorithms for skylining in databases have been proposed, see e.g. [4], [12], or [1]. The exponential growth of skyline sets with the number of query predicates, however, limits the paradigm's applicability. [8] presents rough bounds for skyline sizes assuming independently distributed scores and shows that positive correlation of scores usually decreases and anti-correlation increases skyline sizes. However, no analytical model is known yet to accurately estimate skyline sizes for a given query and database instance. Moreover, post-processing skylines to get a minimum set cover with a difference of at most some fixed value ε between any skyline object and some object in the set cover has been proved NP for more than 3 dimensions [11].

Computing convex hulls over datasets (or data envelopment analysis) is a related problem, since all vertices of the convex hull over points in $[0, 1]^n$ form a subset of the respective skyline (for an efficient multi-dimensional algorithm see [3]). Thus, a convex hull could provide a coarse impression of the skyline. However, especially for anti-correlated predicates, many actual skyline objects may reside far from the convex hull, whereas desirable objects on the convex hull may not exist in the database.

2.2 Basic Query Models of Top k and Skyline Queries

Though top k retrieval is a cooperative and efficient way to answer queries, users first have to define individual utility functions. However, it is hard to guess 'correct' weightings for this function. Consider the example of our traveling researcher: usually a lot of different flights will be available, but most trade-offs are difficult to assess. What does it mean that a closer airport is 0.46 times more important than a less expensive ticket? Moreover, a really good bargain might compensate for most other inconveniences, but assigning high weightings to the price from the start might not be a good strategy, if the database only contains rather similar priced flights.

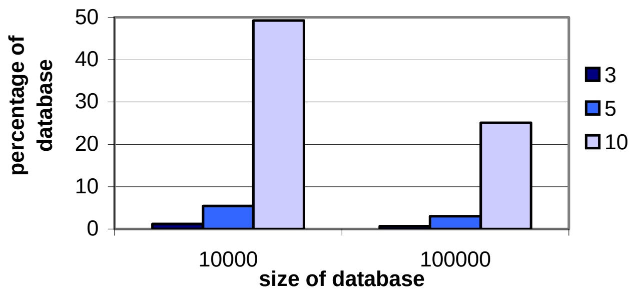

Fig 1. Skyline sizes (in % of database retrieved) for 3, 5 and 10 soft query predicates

To overcome these problems retrieving a set of optimal objects with respect to all characteristics ('skyline' queries) has been established in query processing. However, the size of result sets generally strongly depends on the number of query predicates. Experiments in [1] show that the skyline grows exponentially and even for relatively small numbers of predicates already contains a vast portion of the entire database (cf. fig. 1 for the case of independently distributed scores). So, for practical applications the entire skylining paradigm often proves useless due to prohibitive result sizes:

Example (cont.): The query our researcher will pose to get an adequate flight will usually have to optimize several soft predicates: the flight booking will generally include predicates on the airport in the Bay Area, the departure/arrival date/time for the flight, the airport in New York, the departure date/time for the return flight, the specific airline and the respective ticket price. On all these predicates usually preferences exist (closest time for flight dates and distance for airports, specific airlines for frequent traveler programs and a minimum price). Even stating only the bare necessities we thus already are left with 6 soft predicates for the basic query rapidly increasing, if we also take class/upgrades, shortest traveling times, number of stops, etc. into account. The same applies for booking a suitable hotel, the predicates here may contain the hotel's preferred category, distance to the conference venue, amenities and price. And even if some of these characteristics (like a hotel's category and its price) are correlated limiting down the skyline size, we will nevertheless experience many independently distributed dimensions (like arrival date and choice of airports).

3 Sampling Skylines and Refining Queries with User Feedback

We have seen that even simple skyline queries may already contain many predicates. Though intuitive to ask, skylines are inefficient to compute and even if they could be computed in acceptable time, it would still not be sensible to return something like 25-50% of the entire database to the user for manual processing. Our approach aims at combining the advantages of both top k retrieval and skylining. Fig. 2 shows their respective strengths. The basic idea is to form an interactive workflow for cooperative retrieval using samples of the skyline to give users an overview of the best possible objects, evaluate their feedback and perform focused, yet meaningful top k retrieval.

	intuitive querying	manageable result set
skyline queries	✓	✗
top k retrieval	✗	✓

Fig 2. Characteristics of top k retrieval vs. skyline queries

Bridging the gap between efficient retrieval and vague query predicates has already occurred in query processing. For instance in multimedia databases describing images for retrieval is quite a hard problem that is often dependent on inherent features like color histograms or textures used for content-based retrieval (see e.g. [6] or [16]). Whenever users are not able to directly express query needs by stating exact values for query predicates, it has proven helpful to let users pick some representative examples from the database collection and then retrieve objects similar to the chosen examples. The algorithms used are generally quite similar and rely on the idea to evaluate common characteristics of the examples to derive some abstract values for improving the retrieval. We will rely on the basic techniques for database retrieval by multiple examples as presented in [10] for the application in our cooperative querying framework. For simplicity we focus only on positive feedback, but also explicit negative feedback on disliked objects could be used and exploited in step 3:

Generic algorithm for query by example:

1. Derive a suitably small set of examples (documents, images, etc.) from the database collection (randomly or given by the skyline of a first tentative query)
2. Get the user's feedback on preferred objects within the example set.
3. Derive the characteristics of a query that would have scored all positive examples most successfully (feature selection, re-weighting)
4. Re-evaluate this new query over the database collection and return the best objects to the user
5. If the user is not yet satisfied with the result set returned, take the set of step 4 as new examples and repeat the algorithm from step 2.

Please note that in contrast to e.g. query by visual example (usually starting with random images) our application in cooperative retrieval already starts with a skyline query. The feedback on subsequently retrieved examples is usually already of sufficient quality to deduce a final top k query, because users will never be offered dominated objects as examples. Hence, feedback can already be expected to be sufficiently focussed allowing to skip step 5. To demonstrate this advantage we will compare our skyline samples to randomly chosen samples from the entire set in section 4.1.

Still, one problem when using skylines to get user feedback is their prohibitive size and computation costs. Deriving a good sample of the skyline for getting feedback is therefore essential. Our work in [1] has laid the foundation to derive skyline samples without actually having to compute the entire skyline set. Since skylines tend to be still manageable in smaller dimensional problems, we propose to use the skylines of subsets of query predicates. We will now present a new algorithm for query processing and in the next section show that the sample derived is of sufficient quality. In the following we will assume that the parameter q is chosen sufficiently high to allow for each score list to be in at least one choice of subsets and that m is chosen adequately to produce low-dimensional subsets of the query predicates ($m < n$). Please note that the algorithm will output only roughly k objects depending on the numbers of regions of interest derived from the feedback. We will assume that the result set should contain at least a single object from each region of interest specified in the feedback.

```
Algorithm: Cooperative Querying with Reduced Dimensions Skylines Samples
0.   Given a top k query with $n$ query predicates, where each predicate can be evaluated by assigning numerical scores to all objects in the collection, and given a score list $S_i$ ($1 \leq i \leq n$) for each such predicate ordered by descending scores. Initialize two sets P, F := $\emptyset$ to contain sample and feedback objects together with their scores, a set W of $n$-dimensional vectors of weightings and a counter for regions of interest as given by feedback. Initialize a counter j := 0.
1.   Randomly select $q$ subsets of score lists, each containing $m$ different lists, such that each of the $n$ lists occurs in at least one of the $q$ subsets as shown in [2].
2.   For each $m$-dimensional subset do
     2.1. Calculate the complete skyline $P_i$ with respect to the $m$ score lists in the subset and for all objects retrieve their missing scores from the other ($n$ - $m$) dimensions
     2.2. Compare any two objects in $P_i$ that have equal values with respect to all $m$ dimensions pairwise. If an object is dominated by another, discard the dominated object from $P_i$.
     2.3. Union the reduced dimension skyline with the sample set $P := P \cup P_i$ (duplicates are eliminated).
3.   Deliver the objects in set P as a sample of the $n$-dimensional skyline to the user ,and allow the user to pick a set $F \subseteq P$ of objects from the sample.
4.   Calculate the difference set D between P and F, i.e. D := P\F
5.   While F is not empty do
5.1. Set j := j+1
5.2. Remove an object $o$ from F, add it to a new set $F_j$ and calculate the minimum distance $d$ from $o$ to any object in D.
5.3. Remove all objects, whose Euclidian distance from $o$ is smaller than $d$, from F and add them to $F_j$.
5.4. Calculate the $n$-dimensional vector $w_j$ of weightings for the best query point having the minimum distance from the objects in $F_j$ with respect to the generalized ellipsoid distance as shown in [10]. Add $w_j$ to set W.
6.   If k > j choose k' as closest integer number larger than k divided by j, else k':=1.
7.   For all j elements in W initiate a top k' search using the arithmetical average weighted with $w_j$ as utility function and return all results as query result.
```

In our experiments sampling with three-dimensional subsets ($m = 3$), values of $q = 10$ for ten score lists ($n = 10$) and $q = 15$ for 15 score lists ($n = 15$) has already provided sufficient sampling quality. Section 4.1 will deal in detail with an experimental evaluation of our sampling scheme's quality using different score distributions. Deriving optimal query weightings from feedback objects has been in deep investigated by [10] for the case of a single region of interest.

4 Theoretical Foundations and Practical Analysis

We bridge the gap between top k retrieval and skylining by providing a first impression of the database that allows for an automatic deriving of weightings from user feedback. To facilitate this, our skyline samples have to fulfil several conditions:

- **No dominated objects:** Positive feedback on already dominated objects may lead to weightings inconsistent with optimal solutions in the final result.
- **Practical runtime:** Retrieving final result sets needs user feedback, thus deriving samples in a timely manner is essential for good overall runtimes.
- **Manageable size:** Sample sizes have to be quite limited to allow for manual feedback, while staying representative enough of the actual skyline.
- **Representativeness:** To provide a good impression of the database content samples must be representative of the entire skyline set and not biased.

Let us now consider the conditions that we posed for our sampling scheme. The following theorem states that -without having to calculate the high-dimensional skyline-our sampling nevertheless always contains only actual skyline objects.

Theorem 1 (Samples contain no dominated objects):
Every object *o* in the sample P chosen by our retrieval algorithm in step 2 is an actual skyline object with respect to all query predicates.
Proof:
We have to show that only skyline objects are added to P in step 2.3. Assume that we have chosen an arbitrary subset of score lists and then calculate the skyline P_i of this subset like shown in step 2.1. Let us further assume that we have derived set P_i' by pairwise comparing objects with equal scores in the chosen subset of dimensions for domination and discarding all dominated objects like shown in step 2.2. Since the set P_i' is added to P after step 2.2, we have to show that it contains only objects of the higher dimensional skyline.
Now for the sake of contradiction let *o* be any object of P_i' and assume that *o* is not a skyline object with respect to all dimensions, i.e. some object *p* exists dominating *o*. Thus, for all lists S_i $(1 \leq i \leq n)$ and the respective scores s_i we get $s_i(p) \geq s_i(o)$ due to the definition of domination. If, however, some index h would exist with $s_h(p) > s_h(o)$ and h would be the index of any score list within the chosen subset, *p* would already dominate *o* with respect to the chosen subset in contradiction to $o \in P_i$'. Thus *o* and *p* have to have equal scores with respect to the chosen subset and since we have compared all the objects with equal scores in the subset pairwise for domination and discarded the dominated ones from P_i', we have to conclude that *p* cannot exist.
Since we have shown P_i' to contain only skyline objects with respect to all query predicates independently of the specific choice of the subset, also the union of different choices of subsets contains only skyline objects. ■

4.1 Performance Experiments and Quality Analysis

We have already proved our samples to contain no dominated objects, but guaranteeing the remaining requirements does not seem as obvious without empirical support:

- The runtime and the size of the samples can be measured directly.
- Since there is no established method of directly measuring the representativeness of samples especially in high dimensions. We will assess the quality of representation with two measures: the set coverage and a cluster analysis.
- Deducing users' preferences involves two steps: grouping user feedback into regions of interest, and then deriving a weighting vector *w* for each region. Quality issues of deriving weightings for a single region has often been investigated (see e.g. [10]). Therefore we will focus on how well these techniques adapt when users have more than one region of interest.

Throughout this section, our testing environment uses a database containing 100,000 objects in different synthesized score distributions for 10 or 15 query predicates. We take the union of 10 or 15 samples, each querying 3 randomly chosen dimensions. We then take averages over 50 runs with newly generated synthetic data. Real databases often contain correlated data. In our traveling researcher example, the distance of a trip will be correlated to the traveling time, and a hotel's price may be correlated to its category. Therefore we investigate different types of data distribution: independent distribution, positive correlation, and anti-correlation. In correlated databases we correlate 6 dimensions pairwise (3 pairs of correlative dimensions) with correlation factor 0.7. Similarly, the anti-correlative datasets correlate 6 dimensions pair-wise with factor -0.7. All tests were run on a single-CPU, 2.2 GHz Pentium-4 PC with 512MB of RAM under MS Windows 2000 and Sun Java VM 1.4.2.

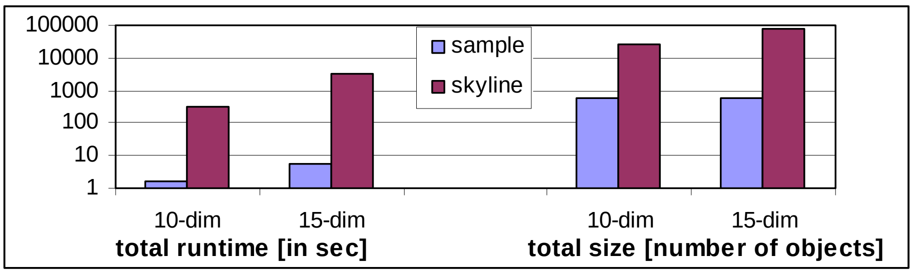

Fig 3. Comparison for runtime for skyline computation (left) and respective skyline size (right)

Total Runtime and Sample Size

Though the actual runtime will vary depending on the hardware/software used, comparing the runtime of skyline sampling versus computing the entire skyline will give us a good idea of our sampling scheme's performance. In Figure 3 (left), we show the comparison of runtime measured in seconds on a logarithmic scale. Our sampling scheme is generally two to three orders of magnitude faster than computing the entire skyline. Note that the sampling scheme can even be parallelized for further runtime improvement, since all subsets are independent in computation and I/O.

Similar to the runtime, the manageability can be directly measured as the size of the result set. Figure 3 (right) shows the average sizes of sample sizes and the actual size of 10 and 15 dimensional skylines. Once again, we see a difference of two orders of magnitude between our sample and the actual skyline. Although a few hundred objects are much more manageable than tens of thousands of objects, going through the entire sample set may still seem challenging. Taking fewer numbers of subsets for the sample would further reduce the size of the sample set. However, it is hard to reduce the size without sacrificing the representativeness of the sample.

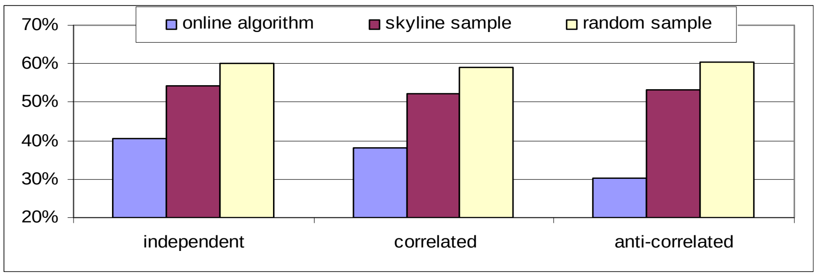

Fig 4. Set coverage [in percent] with respect to an unbiased random sample

Representativeness of Skyline Samples: Set Coverage and Cluster Analysis

The most representative sampling method is taking random samples of the actual skyline. Here no particular region is favored, but obtaining such a random sample requires computing all skyline objects, which is prohibitive in terms of runtime. Previous attempts to give users a first impression of the skyline proposed online algorithms, e.g. [12], where starting with some first examples users were able to steer the computation of the next skyline object towards certain areas of interest, thus resulting in a biased sample. Also our proposed sampling scheme will show some bias in contrast to perfect random samples. The notion of *set coverage* investigates this bias:

Definition of Set Coverage:

Let S be the set of objects in $[0, 1]^n$ and s, s′ $\subseteq$ S two sets with similar cardinality. For each o $\in$ s determine an object o′ $\in$ s′, such that the Euclidian distance between o and o′ is minimized. The set coverage of s over s′ is the percentage of objects from s′ that have been assigned to objects of s.

Measuring the set coverage between independently drawn random samples of the actual skyline set gives us a good impression of the normal coverage between unbiased sets. For our experiments we considered different score distributions (independent, correlated, anti-correlated) and took averages over several runs in each instance choosing a random sample and comparing the respective set coverage of a second independent random sample, our sample and set delivered from the online algorithm in [12]. In all cases the sets were chosen with similar cardinalities, generally different runs show an insignificant variance. Figure 4 presents the results, where the y-axis is the percentage of the objects in the random sample of the actual skyline that are covered by the sample from the online algorithm, our skyline sample and a second independently drawn random sample. With very little variance in all score distributions the best set coverage between two unbiased random samples is around 60%. Clearly the set coverage for our sampling scheme is almost as good showing values around 54% with small variance, while being much better than the online algorithm's ranging between 30-40% and showing a relatively large variance with the score distributions.

Though we have seen our sample to cover randomly drawn samples quite well, we also have to focus on how the objects are distributed over the score space, i.e. if our samples closely represent the *entire* skyline and not just some portions of it. Our *cluster analysis* measures the representativeness of a sample grouped by the relative distribution of objects. Scores for each query predicate are partitioned into two buckets: [0, 0.5) and [0.5, 1] leading to e.g. 2^n different buckets in the case of *n* query predicates. The bucket assigned gives a rough idea where objects are located in *n*-dimensional space. For each sample, we count the number of objects in each bucket and compare the respective numbers using histograms. Thus we can ascertain that clusters of skyline objects are also represented by our sample. Since we cannot show such histograms (already 1024 buckets for 10 dimensions) we use the score space's symmetry and aggregate the buckets having a similar position with respect to the main diagonal. That means we aggregate all buckets with the same numbers of 'upper' score buckets, leading to *n* dimensional histograms for *n* query predicates. Figure 5 shows the histograms assessing the percentage of objects with respect to the sample size in each aggregated bucket (the connecting lines are just inserted to make the histograms more easily comparable). Focusing again on our three different score distributions, figure 5 plots the object distribution over the aggregated buckets of the actual skyline, our skyline sample and a complete random sample of the entire database (to show the normal bell-shaped object distribution under our symmetrical aggregation). We can easily see that unlike the complete sample our sample's distribution aligns smoothly with the original skyline, both showing a distinct shift to the higher buckets. This shift even becomes more pronounced for correlated score distributions (Fig. 5, right). Again our sample resembles the proportions of the actual skyline very well. The diagram for the anti-correlated case is similar, but omitted for brevity.

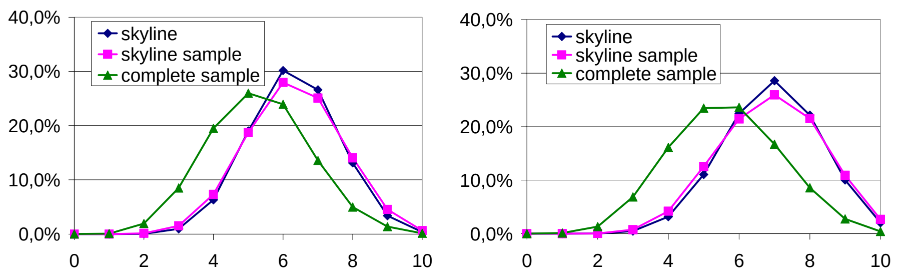

Fig 5. Cluster analysis for independent (left) and correlated (right) score distributions

4.2 Querying Multiple Regions of Interest

Usually in query by example the user selects interesting data objects (e.g. images) and the query is reposed taking the characteristics of selected items into account. However, in skyline sets users might be interested in objects of multiple regions.

Therefore we extended the MindReader re-weighting technique for single regions of interest [10] by setting a threshold for the maximum allowable distance between example objects in a single region and thus separating the selected examples into multiple regions (compare steps 4-7 in our algorithm in section 3). As threshold we chose the minimum Euclidian distance between a chosen example and any example not chosen. We then apply the MindReader algorithm with all chosen examples within this threshold as input and get the optimal weightings for a subsequent top k search. To investigate our query processing algorithm's effectiveness, we set up an experiment on a database containing 100.000 objects. Assuming the existence of three preferred objects in the actual skyline, we randomly selected three 'perfect' objects from the skyline. Ascertaining a certain maximum distance *dist* to at least one of the 'perfect' objects we then picked all eligible objects from the sample drawn and fed these as positive feedback to our re-weighting scheme (please note that the sample may, or may not contain any 'perfect' object). After grouping the examples into regions of interest using the minimum distance to any non-picked object in the sample, we calculated the best weightings (query points) for a top k query and subsequently retrieved the best objects. We then counted how many of the initial perfect objects are in the direct vicinity (again using maximum distance *dist*) of the objects retrieved.

In our experiments for independent score distributions the samples sizes were 336.52 on average, of which an average of 13.52 objects were picked as positive feedback (at an initial value of *dist* = 0.75). These objects had an average distance of 0.579 to the closest 'perfect' object. Our algorithms on average grouped the positive feedback into 9.88 distinct groups leading to the determination of optimal query points with an average distance of 0.580 to the 'perfect' objects. The subsequent top k retrieval for each query point retrieved objects with an average 86.67% of all 'perfect' objects in the direct vicinity (within distance *dist*) of some of them. The experiments for correlated and anti-correlated data distributions show similar results, with increased numbers of positive feedback objects and even higher numbers of correctly retrieved objects in the vicinity of 'perfect' objects, on average over 90%.

5 Summary and Outlook

Cooperative retrieval with vague predicates in databases today uses two techniques: top k retrieval and skyline queries. But whereas top k result sets are well-defined, choosing appropriate compensation functions without detailed knowledge of a database instance is difficult. In contrast posing skyline queries is very simple and intuitive, but result sets grow exponentially with the number of query predicates and thus quickly become unmanageable. In this paper we focused on the efficient utilization of high-dimensional skylines to allow for intuitive querying with vague predicates in information systems, while at the same time guaranteeing manageable result sets.

Starting with a normal skyline query we designed an interactive scheme using feedback on a representative sample of the skyline for subsequent focused top k retrieval guaranteeing manageable result sets. Thus our query processing scheme combines the best of both techniques. We have proven our query processing scheme to *quickly* derive *manageable*, but *highly representative* skyline samples containing *no* *dominated objects*. Evaluating feedback on these samples we have then extended an efficient query by example algorithm to cater for the need in cooperative retrieval to separately investigate several regions of interest. Detailed practical experiments show that our sample can be derived up to two orders of magnitude faster than the actual skyline and is almost as representative as a similar-sized random sample of the actual skyline. Moreover, we showed our focused top k searches to deliver result sets, where over 85% of the actual objects of interest are arbitrarily close to the objects retrieved.

Applications focus on cooperative query processing in information systems for a variety of areas like E-commerce, digital libraries or service provisioning. Our future work will concentrate on further exploiting the scheme and opening up the query by example capabilities for multi-objective retrieval in databases, e.g. [1]. Moreover, we will investigate our sampling scheme's capability for interactively deriving meaningful user preferences that can be used as long-term profiles in specific domains.

References

1. W.-T. Balke, U. Güntzer. Multi-objective Query Processing for Database Systems. *Intern. Conf. on Very Large Data Bases (VLDB'04),* Toronto, Canada, 2004.
2. W.-T. Balke, J. Zheng, U. Güntzer. Efficient Distributed Skylining for Web Information Systems. *Conf. on Extending Database Technology (EDBT'04)*, Heraklion, Greece, 2004.
3. C. Böhm, H. Kriegel. Determining the Convex Hull in Large Multidimensional Databases. *Conf. on Data Wareh. and Knowledge Discovery (DaWaK'01),* Munich, Germany, 2001.
4. S. Börzsönyi, D. Kossmann, K. Stocker. The Skyline Operator. *Intern. Conf. on Data Engineering (ICDE'01),* Heidelberg, Germany, 2001.
5. N. Bruno, L. Gravano, A. Marian. Evaluating Top k Queries over Web-Accessible Databases. *Intern. Conf. on Data Engineering (ICDE'02)*, San Jose, USA, 2002.
6. A. Chianese, A. Picariello, L. Sansone. A System for Query by Example in Image Databases. *Int. Workshop on Multimedia Information Systems (MIS'01),* Capri, Italy, 2001.
7. R. Fagin, A. Lotem, M. Naor: Optimal Aggregation Algorithms for Middleware. *ACM Symp. on Principles of Database Systems (PODS'01)*, Santa Barbara, USA, 2001.
8. P. Godfrey. Skyline Cardinality for Relational Processing. *Int Symp. on Foundations of Information and Knowledge Systems (FoIKS'04)*, Wilhelminenburg Castle, Austria, 2004.
9. U. Güntzer, W.-T. Balke, W. Kießling: Optimizing Multi-Feature Queries for Image Databases. *Int. Conf. on Very Large Data Bases (VLDB'00),* Cairo, Egypt, 2000.
10. Y. Ishikawa, R. Subramanya, C. Faloutsos. MindReader: Querying Databases through Multiple Examples. *Conf. on Very Large Data Bases (VLDB'98)*, New York, USA, 1998.
11. V. Koltun, C. Papadimitriou. Approximately Dominating Representatives. *Int. Conf. on Database Theory (ICDT'05)*, Edinburgh, UK, 2005.
12. D. Kossmann, F. Ramsak, S. Rost. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. *Conf. on Very Large Data Bases (VLDB'02)*, Hong Kong, China, 2002.
13. J. Minker. An Overview of Cooperative Answering in Databases. *Int. Conf. on Flexible Query Answering Systems (FQAS'98),* Springer LNCS 1495, Roskilde, Denmark, 1998.
14. A. Motro. VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM Transactions on Information Systems.* Vol. 6(3), 1988.
15. D. Papadias, Y. Tao, G. Fu, B. Seeger. An Optimal and Progressive Algorithm for Skyline Queries. *Int. Conf. on Management of Data (SIGMOD'03),* San Diego, USA, 2003.
16. S. Santini, R. Jain. Beyond Query by Example. Int. ACM Conf. on Multimedia (MM'98), Bristol, England, 1998.