Chapter 2
On Skyline Queries and How to Choose from Pareto Sets

Christoph Lofi and Wolf-Tilo Balke

Abstract. Skyline queries are well known for their intuitive query formalization and easy to understand semantics when selecting the most interesting database objects in a personalized fashion. They naturally fill the gap between set-based SQL queries and rank-aware database retrieval and thus have emerged in the last few years as a popular tool for personalized retrieval in the database research community. Unfortunately, the Skyline paradigm also exhibits some significant drawbacks. Most prevalent among those problems is the so called "curse of dimensionality" which often leads to unmanageable result set sizes. This flood of query results, usually containing a significant portion of the original database, in turn severely hampers the paradigm's applicability in real-life systems. In this chapter, we will provide a survey of techniques to remedy this problem by choosing the most interesting objects from the multitude of skyline objects in order to obtain truly manageable and personalized query results.

2.1 Introduction

The ever growing amount of available information is one of the major problems of today's information systems. Besides solving the resulting performance issues, it is imperative to provide personalized and tailored access to the vast amount of information available in information and data systems in order to avoid flooding the user with unmanageably large query results.

Christoph Lofi
Technische Universität Braunschweig, Germany
e-mail: lofi@ifis.cs.tu-bs.de

Wolf-Tilo Balke
Technische Universität Braunschweig, Germany
e-mail: balke@ifis.cs.tu-bs.de

B. Catania and L.C. Jain (Eds.): Advanced Query Processing, ISRL 36, pp. 15–36.
springerlink.com

As a possible remedy to this problem, Skyline queries [1] have been proposed, filling the gap between set-based SQL queries and rank-aware database retrieval [2]. Due to the paradigm elegance and simplicity, it has stirred a lot of interest within the database community in recent years. Skyline queries rely on the notion of *Pareto dominance*, i.e., given the choice between two objects, with one object being better with respect to at least one attribute but at least equal with respect to all other attributes, users will always prefer the first object over the second one (the first object is said to *dominate* the second one). This simple concept can be used to implement an intuitive personalized data filter as dominated objects can be safely excluded from the data collection, resulting in the *Skyline set* of the query. The semantic justification of this filter is easy to see using an example: if two car dealers in the neighborhood offer the same model (with same warranties, etc.) at different prices, why should one want to consider the more expensive car?

In order to compute the Skyline set in a personalized fashion, the user needs only to provide so-called *ceteris paribus* ("all other being equal") preferences on each individual attribute (e.g., "lower prices are better than higher prices given that all other attributes are equal"). Although, many works on skyline queries only consider numerical domains and preferences [1,3,4], skylining can generally also be extended to qualitative categorical preferences (e.g., on colors, "given two cars with free color choice, a black car would be better than a red car") which are usually modeled as partial or weak orders [5,6]. Furthermore, many of these preferences don't require any user input during elicitation as they can be deducted from common content in the collection of user profiles (e.g., preferences on price; no reasonable user would prefer the same object for a higher price).

This focus on individual attribute domains and the complete fairness of the Pareto paradigm are the major advantages of skyline queries: they are easy to specify and the algorithm will only remove definitely suboptimal objects. However, these characteristics also directly lead to the paradigms major shortcomings: Skyline queries completely lack the ability to relate attribute domains to each other and thus prevent compensation, weighting or ranking *between* attribute domains. This often results in most objects being incomparable to each other and thus generally causes skyline sets to be rather large, especially in the quite common case of anti-correlated attribute dimensions. This effect is usually referred to as "curse of dimensionality". It has been shown (under certain assumptions on e.g. the data distribution) that the skyline size grows roughly exponential with the number of query attributes [7,8]. However, there is still no reliable and accurate algorithm for predicting skyline sizes given arbitrary database instances and user preferences. Experimentally, it has been validated that already for only 5 to 10 attributes, skylines can easily contain 30% or more of the entire database instance [1,9,10] which is a size clearly unmanageable for most users, rendering the skyline paradigm inapplicable for many real-world problems.

Thus, reducing the size of result sets by choosing the most interesting or most relevant objects from the skyline is a major and prominent problem. However, "interestingness" is usually individual perception and is specific for each user and is thus hard to formalize. Nevertheless, for rendering the skyline paradigm useful for common real world scenarios, such techniques are mandatorily required. Accordingly, an impressive number of approaches have been developed in the recent years

introducing various heuristics for capturing the semantics of "interesting" in order to choose meaningful and manageable subsets from skylines in an efficient manner.

Generally speaking there are four major approaches to address the problem:

- *Relaxation of Pareto Semantics* use weaker variants of the Pareto semantics which less likely lead to incomparability between database objects. These approaches include for example Weak Pareto Dominance or k-Dominant Skylines and are discussed in Section 2.3.
- *Summarization approaches* are presented in Section 2.4 and aim at returning a representative subset which still maintains the diversity and flavor of the original skyline set. Often such approaches are intended to enable the user to grasp a quick overview of the whole skyline set. Examples are Statistical Sampling Skylines and Approximately Dominating Representatives.
- *Weighting approaches* try to induce a ranking on the Pareto incomparable skyline items based on some structural or statistical properties of the data set. Usually, they numerically quantify the "interestingness" of a skyline object explicitly and return the k-most interesting objects. These approaches are showcased in Section 2.5 and are often based on extensive subspace skyline computation. They include for example Top-K Frequent Skyline, Skyrank, or Personalized Top-k Retrieval.
- *Cooperative approaches*, presented in Section 2.6, interactively try to elicit more information from users to refine the preferences in order to focus the skyline sets in a personalized fashion. While abstaining from using heuristics for selecting the skyline objects, they impose an additional interaction overhead on the user. These approaches include for example Trade-Off Skylines.

After introducing the basic formalities necessary for modeling preferences and computing skylines, we will present selected techniques for each of these major approaches.

2.2 Formalization of Skyline Sets Following the Pareto Semantics

Before we survey the field of skyline selection algorithms let us formally introduce the skyline paradigm and the underlying Pareto semantics. Skyline sets can be defined for every database relation $R \subseteq D_1 \times ... \times D_n$ over n attributes. Assessing a preference for the values in domains D_i of each attribute, users can provide a set of up to n complex attribute preferences to personalize the skyline retrieval process:

- A preference P_i on an attribute A_i with domain D_i is a *strict partial order* over D_i. If some attribute value $a \in D_i$ is preferred to some other value $b \in D_i$, then $(a, b) \in P_i$. This is often written as $a >_i b$ (read "a dominates b wrt. to P_i").
- Analogously, an equivalence Q_i on some attribute is an *equivalence relation* on D_i compatible with P_i (i.e., no tuple in Q_i may contradict a tuple in the transitive closure of P_i). If two attribute values $a, b \in D_i$ are equivalent, i.e., $(a, b) \in Q_i$, we write $a \approx_i b$.

- Finally, if an attribute value $a \in D_i$ is either preferred over, or equivalent to another value $b \in D_i$, we write $a \succsim_i b$.

As an example, assume some preferences for buying a car considering the attributes price, color, horsepower, and air conditioning as illustrated in Fig. 2.1.

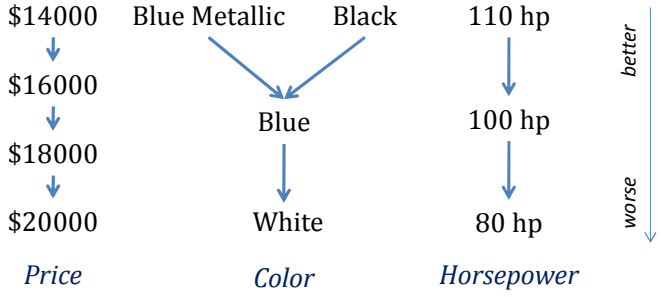


Fig. 2.1 Example partial order preferences for the domain of cars.

Skyline sets as introduced in [1] are defined using the Pareto semantics from the field of economy [11]: some object o_1 dominates an object o_2, if and only if o_1 is preferred over o_2 with respect to any attribute and o_1 is preferred over or equivalent to o_2 with respect to all other attributes. Formally the dominance relationship is denoted as $o_1 > o_2$ and can be expressed as given in Definition 2.1.

Definition 2.1 (Dominance Relationships following Pareto Semantics).

$o_1 > o_2 \Leftrightarrow \exists\, i \in \{1, \ldots n\}\colon \mathrm{o}_{1,\mathrm{i}} >_i \mathrm{o}_{2,\mathrm{i}} \;\wedge\; \forall\, i \in \{1, \ldots, n\}\colon \mathrm{o}_{1,\mathrm{i}} \succsim_i \mathrm{o}_{2,\mathrm{i}}$ where $\mathrm{o}_{\mathrm{j,i}}$ denotes the *i*-th component of the database tuple o_j.

The skyline set (or Pareto skyline) can then be defined as the set of all *non-dominated* objects of the database instance R with respect to all preferences following the Pareto semantics:

Definition 2.2 (Skyline Set).

$$Sky := \{o_1 \in R \mid \neg\exists\, o_2 : o_2 > o_1\}$$

For actually computing a skyline set, there are multiple algorithms available which can be classified into Block Nested Loop algorithms, Divide-and-Conquer algorithms, and Multi-Scan Algorithms. Basically, each object has to be compared to each other one and tested for dominance. However, advanced algorithms try to avoid testing every object pair by employing optimizations and advanced techniques to eliminate as many objects as possible early in the computation process. Efficient skyline algorithms thus require only a fraction of the number of object dominance tests than less sophisticated algorithms.

Block-Nested-Loop (BNL) algorithms are probably the most popular algorithm class and were developed quite early [1]. However, also many state-of-the art algorithm use the BNL approach [12-16] .These algorithms scan linearly over the

input database and maintain a list containing the current intermediate skyline (called windows or block). Each newly scanned object is compared to the objects in the windows, eliminating dominated objects or being eliminated and discarded. If the object is not dominated by any object in the window, it is also added to the window. More sophisticated version of this algorithm maintain the basic design principles of using a single scan and maintaining a window, but employ additional techniques like presorting or indexing to increase the overall performance. Due to the single-scan nature, these algorithms are especially suited to be used in database systems which are often optimized for linear access.

The second popular class of skyline algorithms are Divide-and-Conquer approaches which recursively split the input data and then joins the partial skylines. Although these algorithms have excellent theoretical properties [1,7], there is no efficient implementation of this recursive process [12]. Thus, this algorithm class is very popular from a theoretical point of view but rarely used in actuals software systems.

The third class of skyline algorithms is based on multiple scans of the database instance and includes algorithms like Best or sskyline [17-19]. They can especially provide highly efficient cache-conscious implementations. These algorithms may eliminate objects especially early, but require scanning and modifying the input database numerous times. Thus these algorithms are mainly used when the whole input relation fits into main memory.

From an *order-theoretical point of view*, in skyline computations all attribute preferences have to be aggregated, thus forming the *full product order* P. This product order materializes all dominance relationships between all possible database objects. The Skyline then consists of all those objects *existing in* R which are not dominated by other *existing objects* with respect to P. The formal notion of the full product order is given by Definition 2.3. However, for complexity reasons skyline algorithms obviously cannot materialize the full product order. Nevertheless in later sections, we will encounter approaches relying on the materialization of at least parts of the full product order.

Definition 2.3 (Full Product Order P).

The full product order is given by $P \subseteq (D_1 \times ... \times D_n) \times (D_1 \times ... \times D_n)$, where for any $(o_1, o_2) \in P$ holds $o_1 > o_2$. The semantics of $>$ are given by the Pareto dominance in Definition 2.1.

As an example for Pareto skylines, consider Fig. 2.2: On the left-hand side of the figure, two partial-order attribute preferences P_1 and P_2 are given. The resulting object order P of seven example database objects is shown in the top of the figure. Only two objects are dominated using Pareto semantics, thus five objects form the skyline. In particular, note that object $(i, 1)$ is in the skyline as the attribute value i is isolated in the preference P_1, i.e., no database object may ever dominate $(i, 1)$.

Please note that partial order preferences reflect an intuitive understanding of preferences given by simple statements for each attribute like “I like A better than B”. But when relying on partial orders the additional possibilities for objects

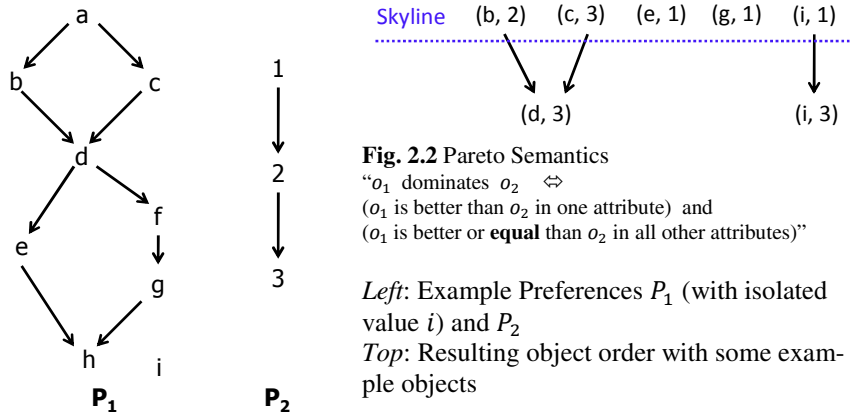


Fig. 2.2 Pareto Semantics
"o_1 dominates o_2 ⇔
(o_1 is better than o_2 in one attribute) and
(o_1 is better or **equal** than o_2 in all other attributes)"

Left: Example Preferences P_1 (with isolated value i) and P_2
Top: Resulting object order with some example objects

being incomparable may introduce efficiency issues for the algorithm design (especially by preventing effective, yet simple pruning conditions). Hence the first introduction of Skylines in [1] only dealt with attribute scorings (i.e., weak order preferences). While this allowed for very efficient query evaluation, the preferences' expressiveness was rather limited [20]. But this drawback was quickly remedied by [21] and [22], which both helped to popularize the use of partial order preferences.

There are multiple reasons for objects being incomparable (e.g., no object can dominate the other one): a) two objects are incomparable if they have antagonistic attribute values, e.g., when considering two cars, one with (75 HP, 5 Liter / 100km) and one with (120 HP, 9 Liter / 100km), these two cars would be incomparable when using the default preferences "more HP is better" and "lower fuel consumption is better" as none of the two objects is clearly better than the other. b) two objects are incomparable if there is no relationship defined between the respective attribute values (often referred to as "missing information"). For example, considering the preferences in Figure 2.2 above, the objects *(e, 1)* and *(f, 3)* are incomparable because the there is no information on whether *e* is preferred over *f* or vice versa. This effect is especially severe for any object sporting an isolated attribute value like e.g. *(i, 2)*. Incomparability due to missing information is a problem exclusive to partial order preferences and is non-existent for total or weak orders. c) two objects are incomparable due to the user being indifferent with respect to certain attribute values. This happens commonly for weak orders where there are equivalence classes of attribute values which are considered equally preferred. But also partial order preferences commonly allow for explicitly modeling equivalences. As an example, consider a partial user preference with an equivalence statements "Red is as desirable as yellow". Then two completely similar objects with one being red and the other being yellow would be incomparable, and both could be potential skyline objects.

2.3 Relaxing the Pareto Semantics

As we already argued, the major problem of skyline queries are the often unmanageable result sets possibly dumping thousands of items on the user for manual inspection. Considering the definition of Pareto semantics, it is obvious that the manageability problems of skylines are heavily aggravated by incomparable attribute values. As soon as two database items are incomparable with respect to even a single attribute, the entire objects are incomparable and may both end up in the skyline. One could say that the Pareto semantics generally is 'too fair'. The obvious solution to restrict skyline sizes is to move from full-fledged partial order preferences to total attribute orders (or at least weak orders), but this in turn might cause problems in the preference elicitation process (more user interactions, sometimes even the introduction of cyclic preferences). The question is whether it is possible to keep partial order preference semantics while adapting the Pareto semantics' fairness to reduce skyline sizes. In this section, we will present skyline approaches based on partial orders which rely on a weaker definition of dominance relationships than given in Definition 2.1 in order to obtain more manageable result sets: weaker dominance conditions mean more object dominance relationships and thus smaller skyline sets. From an algorithmic point of view, these algorithms stay similar to normal Skyline algorithms and just use a different notion of dominance. However, in some cases (like e.g. for weak dominance), the weaker dominance conditions may allow for additional pruning heuristics which may increase the algorithms efficiency.

A first straightforward approach is to explicitly derive weak orders from given partial orders, leading to so-called *level order skylines*. Here counting from the most preferred values down to the least preferred ones, all values are assigned a level. Now any attribute value can be considered dominated by all values on higher levels

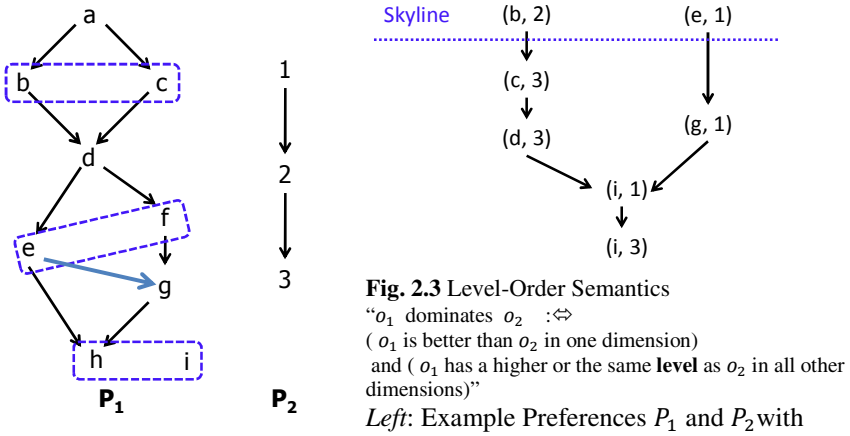


Fig. 2.3 Level-Order Semantics
"o_1 dominates o_2 :⇔
(o_1 is better than o_2 in one dimension)
and (o_1 has a higher or the same **level** as o_2 in all other dimensions)"
Left: Example Preferences P_1 and P_2 with levels order equivalences
Top: Resulting object order with example objects

(see Fig. 2.3). Especially, isolated values for which the user did not provide any preference are considered to be part of the lowest level (and are thus easily dominated).

This concept can be extended into *weak* attribute dominance by further loosening up the dominance semantics. The Pareto semantics and its previously presented variants require an object o_1 in order to dominate an object o_2 to be better in one dimension and at least equal in all others. In case of an object being incomparable with respect to one of the dimensions, no dominance relationship can be established.

However, the basic intuition of skylining is the result should contain all "best" objects – "best" could be interpreted as "there are no better objects". This interpretation can be formalized into the weak attribute dominance as illustrated in Fig. 2.4 and can be phrased as "o_1 dominates o_2 if o_1 is better than o_2 in one dimension and there is no dimension such that o_2 is better than o_1". In general, this dominance criterion leads to significantly reduced result sets (called weak Pareto skylines, and defined by imposing the Pareto semantics on the weak attribute dominance relationships). Weak Pareto skylines can be quite efficiently computed, see [23]. However, the size reductions are usually quite significant, and often even desired skyline objects are removed. When considering the example in Fig. 2.4, it can be observed that the isolated object $(i, 1)$ now dominates the object $(b, 2)$ which shows an overall solid performance, and thus might have been a good choice for many users which is not available anymore.

A less aggressive dominance criterion is provided by the *substitute value* (SV) semantics in [24] which consider Pareto incomparable values within one attribute preference as being equal if they share the same parents and children within the partial preference order. Naturally, the resulting SV skylines will be larger than a corresponding weak skyline.

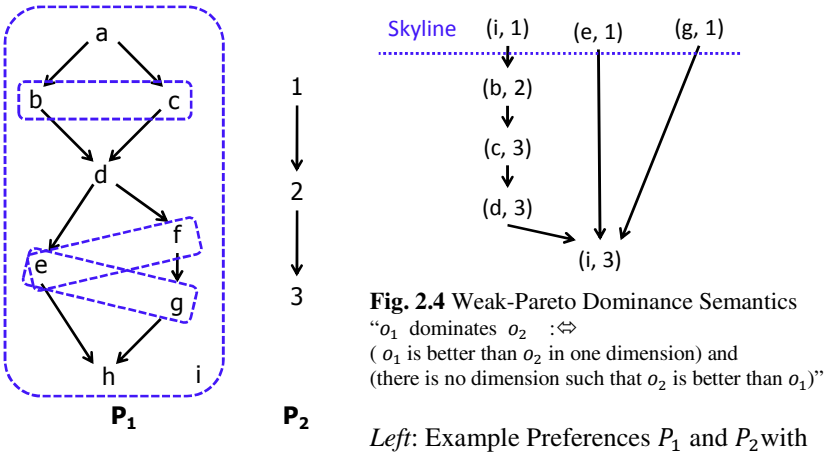


Fig. 2.4 Weak-Pareto Dominance Semantics
"o_1 dominates o_2 :⇔
(o_1 is better than o_2 in one dimension) and
(there is no dimension such that o_2 is better than o_1)"

Left: Example Preferences P_1 and P_2 with weak-Pareto equivalences
Top: Resulting object order and restricted skyline

In [25], the concept of k-dominant skylines is introduced, based on the measure of *k-dominance*. An object o_1 is said to k-dominate another object o_2 if there are at least $k \leq d$ dimensions in which o_1 is better than or equal to o_2 and o_1 is strictly better in at least one of these k dimensions. Using this definition, a Pareto skyline is a k-dominant skyline with $k = d$. K-dominant skylines with $k < d$ are motivated by the observation that with the growing number of dimensions, it becomes highly unlikely that an object is equal or worse with respect to all dimensions than another one as most objects have at least some strong attribute values in an high dimensional space. Thus, rarely any dominance relationships can be established. K-dominant skylines allow objects exhibiting only few good attribute values to be dominated by objects showing many good attribute values. By decreasing the user provided k, the resulting k-skylines are decreased in size. K-Skylines with smaller k are subsets of those with higher a value for k, e.g., given a fixed dataset, the 2-skyline is a subset of the 3-skyline, etc. All k-skyline are a subset of the original skyline. Computing k-dominant skylines requires specialized algorithms which are also provided in the work.

2.4 Summarizing the Skyline

In this section, approaches are presented which aim at finding a subset of objects which serves optimally as a summarization of the full skyline. The main idea behind these approaches is to return just a summarizing set of objects which still maintains the diversity and characteristics of the original skyline, but exhibits a much more manageable size, i.e., sacrificing the completeness of skylines for the sake of manageability. The focus within these approaches is to enable the user to grasp a quick overview of the nature and contents of the skyline result set such that she is easily able to further refine her preferences and / or is directly able to perform subsequent queries for narrowing down the results even further (e.g., appending a top-k query which ranks the skyline result, or provide some SQL constraints to remove unwanted data points).

We will focus on two approaches: a) approximately dominating representatives [26] which returns a subset minimally covering all skyline objects within ϵ-balls and b) statistical sampling approaches [10] with subsequent top-k ranking. Both approaches try to maintain the diversity of the original skyline and do not suggest a ranking of skyline objects.

2.4.1 Approximately Dominating Representatives

Computing *approximately dominating representatives* (ADR) [26] is a subsequent refinement technique of skyline queries which aims at covering the full skyline by an optimal sample in terms of size and accuracy. The ADR query returns a set of k (whereas k is a user specified value) objects $adr_\epsilon \coloneqq \{a_1, \dots, a_k\}$ with the following property: based on a user specified value ϵ, for any other object $a \notin adr_\epsilon$ holds that there is an $i \leq k$ such that the vector $a_i \cdot (1 + \epsilon)$ (i.e., a_i boosted by ϵ

in all dimensions) dominates a. In other words, ADR tries to find a minimal number of skyline objects such that, if they are assumed to be the center of a sphere with radius ϵ in the data space, all other skyline objects (which are not the center of such a sphere) are within one of the spheres around the selected skyline objects (see Fig. 2.5).

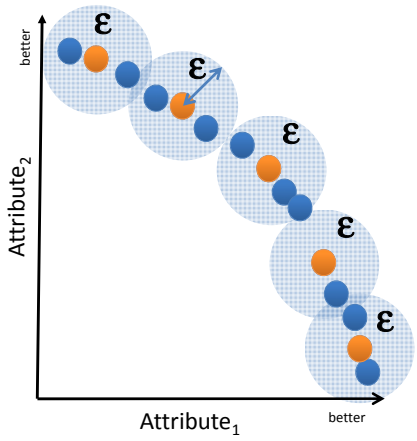


Fig. 2.5 Approximately Dominating Representatives with ϵ-spheres. Only the center objects of each ϵ-sphere is returned.

ADR is designed as a subsequent step after the actual skyline computation, thus no performance advantages can be gained as still the full skyline needs to be computed. Furthermore it has been shown that although finding the smallest possible ADR (assuming the skyline is given) is in linear time complexity in the number of skyline elements for two attribute dimensions, the problem is unfortunately NP hard for more than two dimensions. Accordingly, approximation algorithms have been developed which run in polynomial time, but sacrifice some of the accuracy of the cover.

The resulting cover will contain some objects from the full spectrum of the skyline, disregarding any additional semantic or structural properties. For example, when considering a simple two dimensional skyline on cars with the attributes top speed and price, it will contain all flavors of different cars with respect to those two attributes: very fast cars which are very expensive, many in-between variations of less expensive but slower cars, down to very slow and very cheap cars.

2.4.2 Statistical Sampling Skylines

Sampling skylines [10] follow a similar base idea as they are designed to return a representative subset of the original skyline. However, the *sampling skyline* approach is specifically designed for cooperative retrieval scenarios and is intended to precede a later query-by-example user interaction. Compared to approximately dominating representative skylines, the focus is more on computational performance and less on the accuracy and representativeness of the returned sample.

The intended user workflow is a follows: a) quickly generate an sample set from the database using the sampling skyline algorithm (this set is intended be representative for the most interesting objects of the database according to some provided attribute preferences) b) elicit feedback from the user on which objects from the sample she prefers c) discover common characteristics of the chosen objects algorithm, e.g., in order to derive a top-k utility function d) return a ranked list of best database objects to the user.

The main goals of the sampling algorithm is that the resulting sample can be computed fast (this especially means that it can be computed significantly faster than the full skyline), and that the result is representative for the full skyline in terms of the object diversity. Also, the size of the sample should be small and manageable.

The algorithm is based on the idea that for sampling a skyline with n query attributes, q subspace skylines on randomly selected subspaces with a dimensionality of $m < n$ are computed and then summarized (thus, this approach is a randomized variant of the subspace analysis-based approaches presented in the next section). Each of these q subspace skylines represents a "topic of interest" and contains objects which are "optimal" considering the respective focus of the chosen subspace. Furthermore, these subspace skylines can be computed extremely fast compared to computing the full skyline. By generating a duplicate-free union of the cleaned subspace skylines, a sample is obtained (see Fig. 2.6).

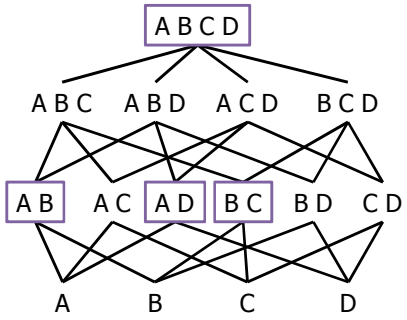


Fig. 2.6 Statistical Sampling Skylines. A skyline query on four attributes (A, B, C, D) is sampled by a union of the randomly selected subspace skylines with $m = 2$ dimensions (A, B), (A, D), and (B, C).

This sample can be shown to be statistically representative for certain values of m and q, leading to a sufficient number of different topics of interest to be covered, e.g., considering a skyline of a car database, the sample could contain some fuel efficient cars, some cheap cars, some fast cars, and some luxurious cars. Furthermore, the sample contains only real skyline objects and can be computed significantly faster than the full skyline.

2.5 Weighting Characteristics of Skyline Points

Approaches presented in this section also aim at generating a sample of the skyline. However, in contrast to the approaches presented in the previous section which try to cover the full diversity of different skyline objects, the following

section presents approaches which select and also rank a subset of the skyline due to some explicitly modeled measure of "interestingness". Especially, in contrast to summarizing techniques, objects showing some rare and extreme values are often not considered as being interesting by these approaches. The Pareto skyline operator treats all skyline objects as being equal, i.e., it does not impose any ranking on the result set. However, the following approaches claim that there are more important and less important skyline objects, and that "importance" can be captured by properties like e.g. the data distribution, the structure of the subspace skylines, or other statistical means.

The first approaches presented in this section will focus on the analysis of subspace skylines, i.e., objects are more interesting depending on in which and in how many subspace skylines they appear. The latter approaches will try to capture semantic importance of skyline objects by relying on the number of objects a given skyline object dominates.

2.5.1 Skycubes and Subspace Analysis

Focusing on computing and analyzing subspace skylines is a very popular approach to derive rankings of Skyline objects. However, computing a larger number of subspace Skylines is prohibitively expensive for online algorithms. Thus, in order to enable the extended use of subspace analysis, *skycubes* [27] have been developed. Skycubes are similar to datacubes used in data warehousing in that respect that they contain all precomputed non-empty skylines of all (possibly up to $2^d - 1$) subspaces of a given dataset. By precomputing all these skylines, drill-down analysis or subspace skyline membership queries can be answered quickly and efficiently. Furthermore, when computing all subspace skylines for a skycube, specialized algorithms can be used which rely on different computation sharing techniques. Thus, computing a whole Skycube is more significantly more efficient than computing all subspace skylines individually.

Mainly, two methods have been proposed to compute all skylines for all subspaces, both relying on traversing the lattice of subspaces (see, e.g., Fig. 2.7) either in a top-down or bottom-up manner. In the bottom-up approach, the skylines in a subspace are partly derived by merging the skylines from its child subspaces at the lower level. In the top-down approach, recursively enumerate all subspaces and compute their skylines from the top to bottom level. During this computation, lower subspace skylines may reuse results from higher subspace skylines, thus significantly conserving computation time. This turns out to be much more efficient than the bottom-up approach. Ultimately, these algorithms allow for computing a skycube up to two magnitudes faster than computing all subspace skylines individually.

With having all subspace skylines readily available due to the materialization of a Skycube, the path is cleared for extensive subspace analysis. For example in [28], the semantics of subspace skylines are explored and researched. Especially, the concept of decisive subspaces and skyline groups as semantically interesting applications of subspace analyses is introduced. As this type of Subspace analysis

synergizes greatly with the skycube algorithm, even a combined research work unifying both approaches was later provided in [29].

A crucial concept in subspace analysis of skylines are skyline groups. Skylines in subspaces consist of projections of objects. For a projection of an object that is in the skyline of a subspace, the set of objects that share the same projection form a so called coincidence group. The maximal coincidence group for a subspace is a called skyline group.Whether an object is in the skyline of the full space or of some subspaces is determined by the values of the object in some decisive subspaces, i.e., those smallest subspaces for which an object is part of a skyline group. The decisive subspaces and the values in those subspaces vary from object to object in the skyline. For a particular object, the values in its decisive subspaces justify why and in which subspaces the object is in the skyline and thus represent the deciding semantics for the membership of the object in the skyline. It has been suggested that Skyline groups can be used to summarize the full skyline by selecting some common representatives from the skyline groups of the skycube.

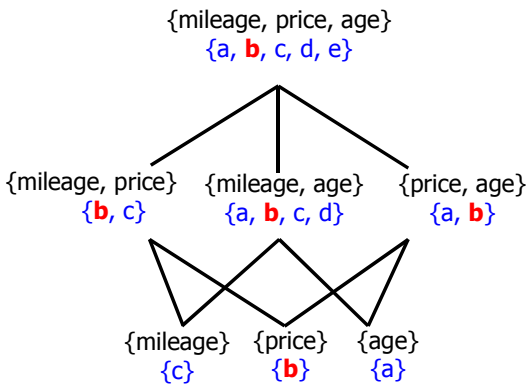


Fig. 2.7 Skycube Lattice of a car database on attributes $mileage$, $price$, age and skyline objects $\{a, b, c, d, e\}$

Top-1 Frequent Skyline: $\{b\}$
Top-3 Frequent Skyline: $\{b, a, c\}$

In [30], it is suggested that a metric called *skyline frequency* can be used to rank and select skyline objects by their interestingness. Skyline frequency counts for each skyline object the number of its occurrence in all possible non-empty subspace skylines, claiming that skyline objects with a higher subspace skyline frequency are more interesting to users than those with lower frequencies. This approach is extended by using skyline frequency to present the user with a ranked list of k most "interesting" skyline objects (so called *top-k frequent skyline*). As an example for this approach consider a skyline query on a used car database with the dimensions mileage, price, and age (see Fig. 2.7). The most frequent skyline object in all subspaces is b, while a and c are similar frequent.

This approach can be seen as a fusion of top-k queries and skyline queries which retains the easy and intuitive query specification of skyline queries, but also allows for limiting the number of objects in a result set to the k most interesting objects which are additionally ranked. The ranking function relies purely on structural properties of the underlying subspace skylines, no additional user input is necessary.

Specialized algorithms are provided for computing the top-k frequent skyline, in an either exact, or due to the high computational complexity, approximate manner. Also, pre-computed skycubes can be used to further accelerate the computation of the top-k frequent skyline.

2.5.2 SKYRANK

The SKYRANK algorithm [31] also relies on ranking skyline objects based on their subspace relationships. However, the measure of "interestingness" is not just based on the number of appearances of an object in any of the subspace skylines, but on the dominance relationships of the skyline objects in the respective subspaces. Following heuristic provides the foundation of SKYRANK:

- A skyline object is more interesting if it dominates many other important skyline points in different subspaces.
- The interestingness of a skyline object is propagated to all skyline object that dominate it in any subspace.

This means, the interestingness of an object in the full space skyline increases the more other skyline objects it dominates in any of the subspaces. Furthermore, this interestingness is amplified if those objects it dominates have been especially interesting themselves (i.e., were also dominating many original skyline objects in some subspaces).

To compute this recursive concept of interestingness, SKYRANK relies on the notion of a skyline graph. The skyline graph contains all skyline objects of the full data space as nodes. The edges are constructed using the skycube of a dataset. For every skyline object o of the full dimensional space (the nodes of the graph), each subspace of the skycube is tested if o is part of the corresponding subspace skyline. If it is, no edge is added. However, if o is not part of the subspace skyline, a directed edge is added from o to all objects which dominate o with respect to the current subspace. This concept is illustrated in Fig. 2.8 for a part of the skyline graph of the full skyline graph just focusing on a two dimensional subspace $\{x, y\}$.

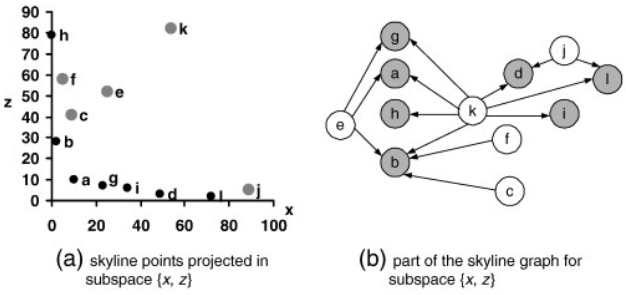


Fig. 2.8 Part of a Skyline graph the subspace $\{x, z\}$ with the respective subspace skyline.
Darker nodes / objects are subspace skyline objects.

The idea for capturing the semantics of "propagating interestingness" is to use link-based ranking methods. For SKYRANK, the PageRank [32] algorithm was chosen which was originally designed for ranking web pages for web search engines. The algorithm models the behavior of a random web surfer and iteratively approximates the stationary probability of the surfer visiting a web page (represented by the PageRank score). The basic heuristic is that web pages with many in-links have more authority and are thus more important than those with few in-links, and also that a page may transfer some of its authority to another page by linking to it. The authors of SKYRANK claim that a skyline graph implies similar semantics than a web graph: by linking to an object o_1, an objects o_2 transfers some of its authority to o_1 (i.e., o_2 is dominated by o_1 in some subspace, thus o_1 must be more important). Furthermore, objects with many in-links are more important than those with few in-links (they dominate more other objects).

Accordingly, SKYRANK presents algorithms for efficiently constructing skyline graphs and applies a link-analysis algorithm similar to PageRank in order to compute scores for each skyline object. These scores can be used to select a ranked list of the k most interesting skyline objects. Furthermore, the algorithm can be personalized by providing top-k-style weightings for all of the involved subspaces, thus allowing for an either user agnostic or personalized retrieval model.

2.5.3 k Most Representative Skyline Points

In contrast to the previous approaches which rely on the nature of subspace skylines, the following approach aims at summarizing a skyline by using the k most representative skyline points (RSP) [33]. For capturing the semantics of this skyline selection, RSP defines the concept of "representative" by the population, i.e., a skyline point is more representative the more objects it dominates. This representativeness metric can be used to rank skyline objects. Finally, the k-most representative problem is stated as finding the k skyline points (whereas k is given by the user) with the maximal number of dominated database objects.

This approach also incorporates ideas from top-k retrieval into skyline queries by returning the k most interesting objects in a ranked fashion. Again, the selection and ranking is based on purely structural properties of the skyline objects, not requiring additional user feedback.

The underlying problem of this operation is similar the set cover problem which also underlies the approximate dominating representative approach in Section 2.4.1. Thus, the runtime complexity is also similar: for more than three query dimensions, the problem is NP-hard, but can be approximated by greedy heuristics. In addition, the authors provide a more efficient randomized approach with a better scalability and establish theoretical accuracy guarantees.

Furthermore, this approach has been extended in [34] to retrieve the k representative skyline points defined as the set of k points that minimize the distance between a non-representative skyline point and its nearest representative.

2.5.4 Personalized Top-k Retrieval / Telescope

In [35], the authors propose a more personalized approach to selecting the k best ranked skyline objects. Instead of relying on user oblivious ranking functions like the previous works on k-dominance or k-representatives, the ranking of skyline objects is steered by additional user provided preferences for weighting individual attribute dimensions (but is still refraining from unintuitive quantitative ranking functions as used by traditional top-k querying). Thus, this approach bridges into the area of cooperative approaches presented in Section 2.6 which focus on additional user interactions for selecting skyline subsets, but which also avoid ranking of skyline objects.

In addition to the attribute preferences required to compute the skyline, the user provides preferences expressing a *precedence order* of the involved attribute dimensions. For example, assuming a used car database, a user can provide attribute preferences like e.g. higher fuel efficiency is better, lower prices are better than higher prices, and red is better than blue which is better than yellow. Additionally, she may state that color is more important to her than for example the price.

For actually returning a ranked skyline containing k objects, a data structure similar to a skycube lattice is traversed based on the precedence preferences. For each level, the subspace nodes are ordered with respect of the precedence preference with subspaces containing more important dimensions to the left. The traversal starts at the top node representing the skyline of the full query space and an empty result set. The actual navigation follows one of two access modes:

- Vertical to left-most child: If the current node contains too many skyline objects (i.e., more than k - size of current result set), the evaluation navigates to the child subspace with the highest precedence (i.e., the left most child).
- Horizontal to right sibling: If the current node contains not too many objects (i.e., less than k - size of the current result set), all skyline object of the current subspace are added to the result set and the evaluation continues with the next sibling to the right (the subspace with the next highest precedence).

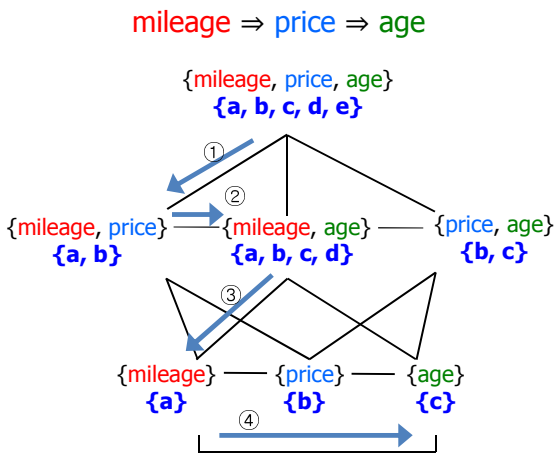
This navigation model is further illustrated in Fig. 2.9 for a used car database. The provided precedence is "low mileage is more important than low price, which is more important than young age", and $k = 3$ skyline objects are to be returned. In the first step, the algorithm starts at the root node of the full subspace which contains more than 3 objects. Thus, it navigates to the next lowest subspace with highest precedence $\{mileage, price\}$ and adds both objects to the intermediate result. It continues to the next sibling subspace $\{mileage, age\}$ which contains four objects (and thus too many to fit the result of required size k). Without selecting any of those objects, the algorithm continues to the next lower subspace $\{mileage\}$ which contains only the object a which is already part of the intermediate result. The same is the case for the next sibling subspace $\{price\}$ and the object $\{b\}$. Finally, object c is added to the result from the next sibling subspace $\{age\}$ and the computation ends.

Fig. 2.9 Telescope on a car database with attributes $mileage$, $price$, age and skyline objects $\{a, b, c, d, e\}$

k:=3 with precedence
mileage ⇒ price ⇒ age

Intermediate Results

Step	Result
1	$\{\}$
2	$\{a, b\}$
3	$\{a, b\}$
4	$\{a, b, c\}$



2.6 Cooperative Approaches

The previously presented approaches strictly rely on structural or statistical properties in order to reduce the size of the skyline set. However, all these approaches need to introduce quite momentous assumptions with respect of the implicated semantics (e.g., a skyline object dominating more database objects than another one is better, Pareto-incomparable objects should be treated as dominated under certain conditions, objects occurring in more subspaces are more important, etc.). Thus, although these approaches successfully decrease the size and, in some cases, also improve the computation speed of the reduced skyline set, it remains unclear if the resulting sets are more helpful to the user.

The approaches presented in this section choose a different strategy for choosing objects from an unmanageable large skyline. Instead of relying on assumed structural semantics capturing interestingness, cooperative approaches interactively elicit additional preference information directly from the user in order to steer the selection of Skyline tuples from large result sets.

2.6.1 Interactive Preference Elicitation

Early cooperative approaches like e.g. the online algorithm presented in [3] allowed for a progressive computation of the skyline set, enabling the user to halt the process at any time in order to provide additional or different preferences for further customizing the result set. In contrast, the works in [36] suggest an interactive preference elicitation process suggesting the most informative preference decisions to the user and is geared towards partial ordered preferences on categorical domains. Especially partial order preferences are susceptible to overly large skyline sets, mainly due to incompleteness of the provided user preferences (e.g., isolated values, incomparable values within a single domain, etc.). To address this problem, an iterative elicitation framework is proposed which collects additional

user preferences in each iteration. The framework aims at both minimizing the required amount of user interaction and while also maximizing the resulting skyline reduction. At the same time, it tries to identify a reasonably small and focused skyline set.

This objective is realized by heavily exploiting the knowledge on the cardinality of different domain values with respect to the database instance. During any iteration, the currently available user preferences are examined in context of the current skyline. Based on the distribution of domain values, the algorithm tries to identify those preference statements (i.e., pairs of domain values a and b such that either a preference $a > b$, a$< b$, or $a \sim b$ could be stated by the user) not part of the current user preference which would result in the largest skyline reduction. After identifying the potentially most effective statements in terms of size reduction, they are presented to the user as questions. Now, the user may choose some of the suggested statements which are then added to the user preferences. For example, assume a car database and no previously provided preferences. Assuming that half of the cars are red and the other half is blue, it is sensible to ask the user whether she prefers red or blue cars (or even if she doesn't care about the color). Any decision for either red or blue cars will decrease the skyline significantly. But even if the user is indifferent about the color, the resulting larger skyline set can at least be justified more easily as it can be attributed not to incomplete preference domains but to explicit user feedback.

2.6.2 Trade-Off Skylines

Trade-off skylines [37] approach the problem of overly frequent objects pairs which incomparable by introducing the natural semantics of *trade-offs* or compromises. Trade-offs can be seen as means for compensating between different attribute domains, thus vastly expanding the semantics of simple attribute preferences. However, in contrast to top-k queries [2], the compensation is not of quantitative but of *qualitative* nature.

Consider for example two database objects representing cars: let object A be a 'blue metallic' car for $18,000 and object B be a 'blue' car for $17,000, accompanied by a preference favoring cheaper cars and metallic colors. Looking at the ranking on attribute level, both cars are incomparable with respect to the *Pareto order*: one car is cheaper; the other car has the more preferred color. In this scenario, a natural question of a real-life car dealer would be, whether the customer is willing to compromise on those attributes, i.e., if he/she is willing to pay the additional $1,000 for a metallic paint job for that particular car (such a compromise is called a *trade-offs*). If the answer is yes, then object A is the better choice for the user and should dominate object B with respect to a *trade-off enhanced Pareto order*. However, if some object C like a 'blue' car for $15,000 exists, A and C would still be incomparable as the premium for the metallic color on that car C is larger than the $1,000 the user is willing to pay. The basic idea of trade-off skylines is that if users provide several strong trade-offs, many skyline objects can be removed as they are now dominated with respect to this new user feedback. Thus, the skyline is reduced in size and focused consistently with the refined trade-off

enhanced user preferences. Additionally, this kind of user interaction closely models the natural compromises of peoples every day's decision processes. At the same time, the approach abstains from assuming arbitrary user agnostic heuristics for selecting objects.

Trade-offs are elicited interactively, i.e., after computing a preliminary skyline, the user is guided through a trade-off elicitation process which suggests possible effective trade-offs (similar to a car dealer asking his customer additional questions). After the user decides for a trade-off, the trade-off skyline is recomputed and the user interaction continues until the user is satisfied.

However, computing trade-off skylines is quite hard. This is mainly because trade-offs directly modify the product order resulting from attribute preferences which in turn loses its *separability characteristic* [11]. Seperability describes the possibility of decomposing the object order losslessly into its respective base preferences (this why most skyline algorithms can avoid operating on the object order at all). In contrast, in [38,39], it was shown that trade-offs will induce addional relationships to the object order, breaking the seperability and thus at least materializing some parts of the object order is required. This effect can be explained by the definition of trade-offs: trade-offs can be considered as a user decision between two sample objects focusing on a subspace of the available attributes, while treating all other attributes with ceteris paribus semantics. Furthermore, trade-off relationships are transitive and thus may form complex dominance relationships structures spanning several trade-offs and dimensions which are called *trade-off chains*. Especially, the problem of trade-off inconsistencies poses severe challenges as inconsistencies are difficult to detect as they are basically circles in the materialized object order. This problem has been successfully solved in [40].

The algorithms available for computing trade-off skylines have evolved from early algorithms relying on the full matrialization of the object order [39], trade-off sklyines with severly reduced expressivness but not requiring the object order at all [41], to computing trade-off skylines allowing the specification of any consistent trade-off without restrictions while still providing acceptable performance by relying on a compressed datastructure minimally representing those parts of the object order which are cruicial for computing the trade-off skyline [37].

2.7 Conclusion and Discussion

In this chapter, we have presented different techniques and approaches to address the problem of unmanageable skyline result sets. In general, skyline results have been proven to grow roughly exponential with the number of query dimensions up to result sets containing a significant part of the overall database. Thus, skyline results are usually too large to be useful to users. This effect can be explained by the fairness of the underlying Pareto semantics which cannot establish a sufficient number of dominance relationships between objects in higher dimensions.

One of the central problems for the actual application of skyline queries is thus the question of how to select the most interesting objects from skylines in such a

way that the selection best reflects a user's needs. The resulting selection should be of manageable size, easy to compute, and should contain those objects being most interesting for the user.

The techniques for selecting from skyline sets can be classified in four groups: a) strategies using less strict variations of the Pareto dominance criterion b) approaches aiming at a diverse summarization of the skyline set c) approaches which use additional characteristics of skyline objects or subspace skylines to derive a ranking and selection for the original skyline d) cooperative approaches which elicit additional preference information interactively from the user.

However, please note that each of these presented approaches in some way break the absolute fairness of Pareto semantics and replace them by different heuristics for capturing the notion of a skyline object being "more interesting" than others.

While every presented approach has benefits and advantages on their own right, the imposed heuristics all rely on some "ad-hoc" assumptions on what makes a skyline point more interesting than others. However, the "correctness" and usefulness of these assumptions with respect to the real information needs of a given, individual user is very subjective and thus hard to determine. This issue is especially aggravated by the fact that no approach thoroughly deals with the psychological and perceptive implications of the chosen heuristics. Thus, given a particular application scenario and data source with its user base, it is very hard to decide which approach best tailors to the users' individual personalization requirements, and the decision has to be carefully evaluated by the application designers. Therefore, this chapter is intended to help in choosing a suitable technique for focusing skyline results by summarizing the most prominent techniques and their underlying assumptions.

References

[1] Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: Int. Conf. on Data Engineering (ICDE), Heidelberg, Germany (2001)

[2] Fagin, R., Lotem, A., Naor, M.: Optimal Aggregation Algorithms for Middleware. In: Symposium on Principles of Database Systems (PODS), Santa-Barbara, California, USA (2001)

[3] Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: an online algorithm for skyline queries. In: Int. Conf. on Very Large Data Bases, VLDB, Hongkong, China (2002)

[4] Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: International Conference on Management of Data (SIGMOD), San Diego, USA (2003)

[5] Lacroix, M., Lavency, P.: Preferences: Putting More Knowledge into Queries. In: Int. Conf. on Very Large Data Bases (VLDB), Brighton, UK (1987)

[6] Chan, C.-Y., Eng, P.-K., Tan, K.-L.: Stratified computation of skylines with partially-ordered domains. In: International Conference on Management of Data (SIGMOD), Baltimore, USA (2005)

[7] Bentley, J.L., Kung, H.T., Schkolnick, M., Thompson, C.D.: On the Average Number of Maxima in a Set of Vectors and Applications. Journal of the ACM (JACM) 25 (1978)
[8] Chaudhuri, S., Dalvi, N., Kaushik, R.: Robust Cardinality and Cost Estimation for Skyline Operator. In: 22nd Int. Conf. on Data Engineering (ICDE), Atlanta, Georgia, USA (2006)
[9] Godfrey, P.: Skyline Cardinality for Relational Processing. In: Seipel, D., Turull-Torres, J.M. (eds.) FoIKS 2004. LNCS, vol. 2942, pp. 78–97. Springer, Heidelberg (2004)
[10] Balke, W.-T., Zheng, J.X., Güntzer, U.: Approaching the Efficient Frontier: Cooperative Database Retrieval Using High-Dimensional Skylines. In: Zhou, L.-z., Ooi, B.-C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 410–421. Springer, Heidelberg (2005)
[11] Hansson, S.O.: Preference Logic. In: Handbook of Philosophical Logic, vol. 4, pp. 319–393 (2002)
[12] Godfrey, P., Shipley, R., Gryz, J.: Algorithms and analyses for maximal vector computation. The VLDB Journal 16, 5–28 (2007)
[13] Eng, P.-K., Ooi, B.C., Tan, K.-L.: Indexing for progressive skyline computation. Data 4 Knowledge Engineering 46, 169–201 (2003)
[14] Godfrey, P., Gryz, J., Liang, D., Chomicki, J.: Skyline with presorting. In: 19th International Conference on Data Engineering (ICDE), Bangalore, India (2003)
[15] Papadias, D., Tao, G.F.Y., Seeger, B.: Progressive skyline computation in database systems. ACM Transactions on Database Systems 30, 41–82 (2005)
[16] Ciaccia, P., Patella, M., Bartolini, I.: Efficient sort-based skyline evaluation. ACM Transactions on Database Systems 33 (2008)
[17] Torlone, R., Ciaccia, P.: Finding the best when it's a matter of preference. In: 10th Italian Symposium on Advanced Database Systems (SEBD), Portoferraio, Italy (2002)
[18] Boldi, P., Chierichetti, F., Vigna, S.: Pictures from Mongolia: Extracting the top elements from a partially ordered set. Theory of Computing Systems 44, 269–288 (2009)
[19] Kim, T., Park, J., Kim, J., Im, H., Park, S.: Parallel skyline computation on multicore architectures. In: 25th International Conference on Data Engineering (ICDE), Shanghai, China (2009)
[20] Fishburn, P.: Preference structures and their numerical representations. Theoretical Computer Science 217, 359–383 (1999)
[21] Kießling, W.: Foundations of preferences in database systems. In: 28th Int. Conf. on Very Large Data Bases (VLDB), Hong Kong, China (2002)
[22] Chomicki, J.: Querying with Intrinsic Preferences. In: Jensen, C.S., Jeffery, K., Pokorný, J., Šaltenis, S., Bertino, E., Böhm, K., Jarke, M. (eds.) EDBT 2002. LNCS, vol. 2287, pp. 34–51. Springer, Heidelberg (2002)
[23] Balke, W.T., Güntzer, U., Siberski, W.: Restricting skyline sizes using weak Pareto dominance. Informatik - Forschung und Entwicklung 21, 165–178 (2007)
[24] Kießling, W.: Preference Queries with SV-Semantics. In: 11th Int. Conf. On Management of Data (COMAD 2005), Goa, India (2005)
[25] Chan, C.-Y.: Finding k-dominant skylines in high dimensional space. In: ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2006), Chicago, Illinois, USA (2006)

[26] Koltun, V., Papadimitriou, C.: Approximately Dominating Representatives. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 204–214. Springer, Heidelberg (2005)
[27] Yuan, Y.: Efficient computation of the skyline cube. In: 31st Int. Conf. on Very Large Databases (VLDB 2005), Trondheim, Norway (2005)
[28] Pei, J.: Catching the best views of skyline: a semantic approach based on decisive subspaces. In: 31st Int. Conf. on Very Large Databases (VLDB 2005), Trondheim, Norway (2005)
[29] Pei, J.: Towards multidimensional subspace skyline analysis. ACM Transactions on Database Systems (TODS) 31, 1335–1381 (2006)
[30] Chan, C.-Y., Jagadish, H.V., Tan, K.-L., Tung, A.K.H., Zhang, Z.: On High Dimensional Skylines. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 478–495. Springer, Heidelberg (2006)
[31] Vlachou, A., Vazirgiannis, M.: Ranking the sky: Discovering the importance of skyline points through subspace dominance relationships. Data & Knowledge Engineering 69, 943–964 (2010)
[32] Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30, 107–117 (1998)
[33] Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting Stars: The k Most Representative Skyline Operator. In: 23rd IEEE International Conference on Data Engineering, Istanbul, Turkey (2007)
[34] Tao, Y., Ding, L., Lin, X., Pei, J.: Distance-Based Representative Skyline. In: 25th Int. Conf. on Data Engineering (ICDE), Shanghai, China (2009)
[35] Lee, J., You, G.-W., Hwang, S.-W.: Personalized top-k skyline queries in high-dimensional space. Information Systems 34, 45–61 (2009)
[36] Lee, J., You, G.-W., Hwang, S.-W., Selke, J., Balke, W.-T.: Optimal Preference Elicitation for Skyline Queries over Categorical Domains. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 610–624. Springer, Heidelberg (2008)
[37] Lofi, C., Güntzer, U., Balke, W.-T.: Efficient Computation of Trade-Off Skylines. In: 13th International Conference on Extending Database Technology (EDBT), Lausanne, Switzerland (2010)
[38] Balke, W.-T., Lofi, C., Güntzer, U.: Incremental Trade-Off Management for Preference Based Queries. International Journal of Computer Science & Applications (IJCSA) 4, 75–91 (2007)
[39] Balke, W.-T., Güntzer, U., Lofi, C.: Eliciting Matters – Controlling Skyline Sizes by Incremental Integration of User Preferences. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 551–562. Springer, Heidelberg (2007)
[40] Lofi, C., Balke, W.-T., Güntzer, U.: Consistency Check Algorithms for Multi-Dimensional Preference Trade-Offs. International Journal of Computer Science & Applications (IJCSA) 5, 165–185 (2008)
[41] Lofi, C., Balke, W.-T., Güntzer, U.: Efficient Skyline Refinement Using Trade-Offs Respecting Don't-Care Attributes. International Journal of Computer Science and Applications (IJCSA) 6, 1–29 (2009)