Efficient Skyline Refinement using Trade-Offs

Christoph Lofi, Wolf-Tilo Balke
Institute for Information Systems
University of Braunschweig
38106 Braunschweig - Germany
{lofi, balke}@ifis.cs.tu-bs.de

Ulrich Güntzer
Institute of Computer Science
University of Tübingen
72076 Tübingen, Germany
ulrich.guentzer@informatik.uni-tuebingen.de

Abstract— **Skyline Queries have received a lot of attention due to their intuitive query formulation. Following the concept of Pareto optimality all 'best' database items satisfying different aspects of the query are returned to the user. However, this often results in huge result set sizes. In everyday's life users face the same problem. But here, when confronted with a too large variety of choices users tend to focus only on some aspects of the attribute space at a time and try to figure out acceptable compromises between these attributes. Such trade-offs are not reflected by the Pareto paradigm. Incorporating them into user preferences and adjusting skyline results accordingly thus needs special algorithms beyond traditional skylining. In this paper we propose a novel algorithm for efficiently incorporating such typical trade-off information into preference orders. Our experiments on both real world and synthetic data sets show the impact of our techniques: impractical skyline sizes efficiently become manageable with a minimum amount of user interaction. Additionally, we also design a method to elicit especially interesting trade-offs promising a high reduction of skyline sizes. At any point, the user can choose whether to provide individual trade-offs, or accept those suggested by the system. The benefit of incorporating trade-offs into the strict Pareto semantics is clear: result sets become manageable, while additionally getting more focused on the users' information needs.**

Keywords-Database Personalization, Skyline Queries, Trade-Off Incorporation,

I. Introduction

Searching through large information spaces (like product search in the Web or document search in digital libraries) is a demanding problem. Besides response time and scalability issues, users often get either a flood of results, or empty results produced by either under- or over-specified queries. In any case, perfect items with respect to the user's query will hardly ever exist and users will always have to relax on at least some of their constraints by accepting compromises.

In such applications users often have to deal with decision problems involving several attributes. Regarding each single attribute, preferences are clear. But providing preferences on objects characterized by several attributes is difficult. Consider for example a decision problem in real estate: most people would agree that a bigger apartment is nice to have; similarly less expensive rental prices are considered being better. Thus, the ordering of real estate *per attribute* is clear. In contrast, specifying a weighted function *compensating* between size and price is not so easy: How much money is an additional square meter exactly worth?

To avoid this problem the *skyline query* paradigm has been introduced. Users just need to specify the set of interesting attributes and a basic order per attribute. They do not need to supply complex weightings or utility functions. Nevertheless, the skyline result set contains all possibly optimal choices. The semantics of skylines follow the notion of *Pareto-optimality* as introduced in [6]. The skyline contains all objects which are not dominated by others with respect to a given set of preferences. An object dominates another if it is better or equal with respect to all attribute preferences and strictly better in at least one attribute. If only one attribute is considered for object comparisons and all remaining attributes remain fixed, this is often referred to as *ceteris paribus* ('all else being equal') semantics [15].

User preferences [20] individually order domain values with respect to each attribute. Skyline queries are usually restricted to attributes with numerical domains using their natural orders (see e.g. [6], [25], or [4]). But also arbitrary partial orders on categorical domains can be supported (see e.g. [8] or [20]). However, computing skylines over such general partial order preferences has shown to be computationally expensive. A very important practical class of preferences, that is more efficient to handle, are weak orders. They form a total order of equivalence classes of domain values [24]. The user is indifferent between all values within each equivalence class. This model assumes the indifference relation to be transitive. Weak orders can be applied on numerical, as well as on categorical domains. For the remainder of this paper, we will assume all user preferences to be in the form of weak orders.

Example: Let us assume a user wants to rent an apartment in San Francisco. His/her search will involve an individual set of important attributes, such as location, size or rental price. For instance, regarding the rental price and size users usually prefer bigger apartments to smaller ones and cheaper apartments to more expensive ones (given that the remaining attributes show similar values). Each user will also show certain preferences with respect to some categorical attributes like location (e.g. Russian Hill or North Beach is considered better than the Mission District) or the apartment's amenities. The set of 'best' objects (the *skyline set*) can be computed by evaluating all preferences on the attributes in the user's query following the Pareto semantics.

Using such preferences within skyline queries is easy for the user. But this ease of use comes at a price: the respective computation times are impractical for larger databases [13]. And even worse, skyline sizes may grow very fast with the number of query attributes and quickly reach the size of the entire database [12]. Hence for practical usage of the paradigm, reducing skyline set sizes is necessary.

In this work, we focus on directly incorporating user feedback into the query process. We enable the efficient incorporation of user-generated trade-offs extending the concept of 'amalgamated preferences' as introduced in [2]. In the course of this paper we will show that our trade-offs form an expressive and useful tool to integrate user feedback into the skyline evaluation and can effectively combat the adverse effect of anti-correlation on the skyline size. The benefits of our approach are threefold:

- We provide a novel algorithm for efficiently incorporating individual user trade-offs into skyline computations. By exploiting the trade-off information, less relevant objects can be incrementally removed from the skyline and thus a valuable means of personalization is provided. Generally users are allowed to state arbitrary trade-offs for skyline reduction.
- Our extensive experiments on practical as well as synthetic data sets show that the trade-off integrations are quick enough to allow for interactive user feedback meeting real time constraints even for large datasets.
- To support users in specifying trade-offs, we propose a scheme for suggesting some promising trade-offs to the user. This frees users from manually exploring the current result set for finding suitable trade-offs and (having a greedy reduction strategy) also reduces the number of necessary interaction steps. Still the user can always reject suggestions and thus stays in control.

II. Decreasing Skyline Sizes

The problem of too large skyline sizes is not only aggravated by larger numbers of attributes in the query, but also by anti-correlations in the data set. Tests in [6] show that in the case of only 6 dimensions the resulting skylines for a correlated, independently distributed and anti-correlated synthetic dataset are 0.02%, 2% and 26% of the database, respectively. In this case particularly the anti-correlation leads to obviously unmanageable result set sizes. However, in practical queries anti-correlation with respect to the preferences over a dataset occurs quite naturally (e.g., users would naturally prefer highest quality items for the lowest price in product databases, but they will hardly ever exist).

Example (cont.): For apartment searches, there will be at least some interesting attributes that invite anti-correlations, like the rental price and the size. In similar locations a larger apartment is bound to be more expensive than a smaller one. Moreover, for similar sizes the location's quality will adversely affect the rental price. Let us focus on two typical preferences: the number of rooms and the rental price. Most users like spacious apartments for a low price. That means, if the user in our case is offered two 1-bedroom apartments in the same location, only the one with the lower rental price is considered to be part of the skyline.
This paradigm is a very natural way of structuring result sets: only looking at the two attributes price and size a user might be ambiguous between a cheap 1-bedroom apartment and a more expensive 2-bedroom apartment (and indeed both would be part of the skyline), but the user would definitely not like a more expensive 1-bedroom apartment. Since in apartment rental decisions usually more than two attributes are involved, the user might end up with a large skyline. This is at least partly due to the anti-correlation between price and size. Thus, the result set has to be reduced.

The traditional way to deal with the problem of large skyline results are utility functions (e.g. [17], [1]), where for instance the trade-off between size and price is captured in a utility function with personalized weightings. This function quantifies how much a user is willing to pay for more space. However, in practice such functions are extremely hard to state for each user. Still they may have a strong effect on the subsequently derived result set. A different way for dealing with the problem are *trade-offs* built in a qualitative way. In [2] we give a basic formalization of qualitative compensations: users may 'amalgamate' two or more preferences by defining preferences or equivalences *across* the respective attributes. We will build on this formalization (which is revisited in definitions 1-3 of section 3), because it almost exactly defines the notion of trade-offs that we need for our user-centered skyline reductions here.

However, the algorithms presented in [2] can only work directly on the extended Pareto product order *P**. This is a relation containing object comparisons between *all* database objects, i.e. *P** grows quickly to impractical sizes. Since in earlier approaches, the processing of trade-offs needed to materialize this relation, the resulting space and time consumption are prohibitive for online algorithms. In fact, our experiments in [2] report average runtimes of several minutes even in the optimal case (in contrast to between 0.08 and 1.12 seconds for the new approach presented here, see section 5). In a nutshell, our new approach frees the algorithm for incorporating trade-offs from materializing the product order. It is able to work on easy set operations which are efficiently supported *in real time* by current database technology, while still guaranteeing consistency of all user trade-offs. Instead of laboriously materializing *P**, our new algorithm needs only to consider and compare two small subsets of the current skyline. The effort necessary for this is smaller by several magnitudes.

Example (cont.): Let us consider a short and simplifying example: assume a database containing 1–4-bedroom apartments in a price range of 300–1050 USD. A user may pose that more rooms and a lower price are preferable. The preference will look like depicted in Figure 2. The problem is clear: Since there will usually be no perfect 4-bedroom apartment for 300 USD, the skyline will contain objects around the diagonal perpendicular to the bisector of the score axes, containing e.g. 2-bedrooms for 600 USD, 3-bedrooms for 750 USD and 4-bedrooms for 1050 USD. Those choices do not dominate each other, because offering more space comes at the price of higher rent. The top choices with respect to

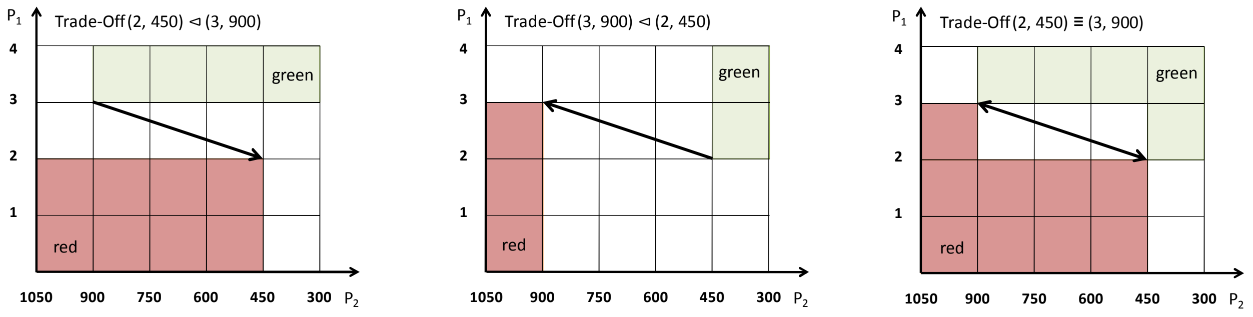


Figure 1. Areas of Domination Resulting from Different Trade-Offs (left and middle: preferences, right: equivalence)

the base preferences (the upper- right triangle-shaped area) will usually be empty.

The basic knowledge (or belief) about such an anti-correlation therefore leads in practical applications to different querying strategies. Probably everybody would prefer a 450 USD apartment over any 1050 USD apartment that is similar in all other aspects (the intuitive ceteris paribus preference on rental prices). Nevertheless specifying a simple 'always prefer the cheapest apartment' preference in a real world system is not helpful, because one cannot expect a sensible result set. In fact, most users are able to refine their needs by investigating usual rental prices for a certain size, i.e. the consideration of suitable trade-offs. Given the data set in Figure 2, a user might price-wise prefer a 3-bedroom apartment over a 4-bedroom apartment. Similarly he/she might actually even prefer a price range of 450–750 USD over cheaper offers, because the respective cheaper apartments (usually) would be much smaller.

Example (cont.): Let us assume the user has a certain opinion about his/her monetary benefit of more space. A *trade-off* can be specified, e.g. a user might state an equivalence between 450 USD 2-bedroom and 900 USD 3-bedroom apartments. The effect on the skyline will be that the relationship between 'pricey' 2-bedroom apartments (more than 450 USD) and relatively 'inexpensive' 3-bedroom apartments (less than 900 USD) is re-established. Within certain bounds, the trade-off presents us with a qualitative notion of what a user considers 'inexpensive' or 'pricey' relative to the apartment's size. In fact, for example a 750 USD 3-bedroom apartment would now dominate a 600 USD 2-bedroom apartment. The user has thus stated a price-related preference for the size of the apartment (with respect to two or three rooms).

Figure 1 shows the effect on the skyline: if there are objects

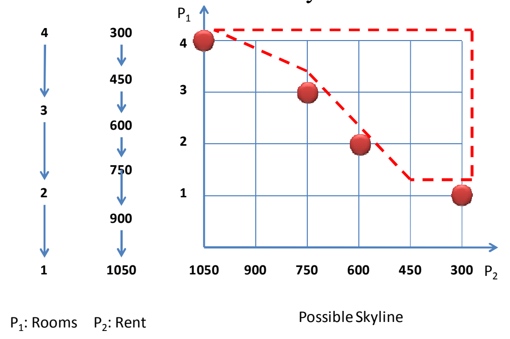


Figure 2. Example Base Preferences and Skyline

in the green area, corresponding objects (respecting ceteris paribus semantics) in the red area are dominated.

III. Related Work

The processing of skyline queries has recently received much attention and many efficient evaluation algorithms have been designed like [6], [25], [4], [10], [19], [8] and more. All these approaches have in common that the emphasis lies on the efficient computation of the entire skyline set, though some of them already stress the problem of large result sizes (e.g., [6], [4]). Having realized this serious challenge for the practical applicability of the skyline paradigm, further research identified basically two ways to deal with the problem: selections based on special characteristics of the data or to rely on user interaction in the form of feedback. We will briefly describe the relevant approaches and then focus on the most closely related to our work.

The idea of exploiting structural characteristics for skyline reductions (often referred to as data-centric or user-oblivious ranking) spawned two basic approaches: in the first class of approaches one tries to give a representative, yet manageable overview of possible choices. The idea is to offer the user a wide range of different options. The second class of approaches relies on focusing on some skyline items with special characteristics. The characteristics are chosen to single out some items more relevant to the user.

Among the first approaches, [18] tried to select a set of approximately dominating representatives such that all skyline objects lie in a sphere with fixed diameter around the selected items. Similarly, [22] uses the number of objects actually dominated by a skyline object as a measure for its respective quality and therefore selects a fixed number of items from the skyline that together dominate the highest number of database objects. However, both selection problems were proven to be NP-hard for three or more attributes. A different approach was introduced in [5] where low-dimensional skyline sets (subspace skylines) were taken as basis for deriving a sample of the skyline.

In contrast, [3] introduced the purely structural notion of weak Pareto optimality. Here all items are removed from the skyline, which are dominated by other items with respect to all comparable attributes, whereas incomparable attribute values are ignored. Hence, a larger number of objects are dominated resulting to smaller skyline sizes. [27] proposes a measure where the number of subspace skylines each item occurs in is taken as a ranking criterion on the complete skyline set. Both of the last two approaches exploit some basic characteristics that seem intuitive, but on the other hand might not be adopted by each user. Hence, they are of heuristic character only.

Other approaches address the problem of large skyline sizes by relying on user feedback (often referred to as personalized ranking or user-centered approaches). Among the first was the online algorithm in [19] that computed the entire skyline, but could be steered by the user to focus on more interesting areas first. The algorithm in [5] involved user feedback on a skyline sample to derive some adequate utility function(s) as an input for subsequent top-k retrieval. The framework given in [10] provides methods for preference revision. If they modify the query in an order preserving way, result sets can be evaluated incrementally.

Another way to express more or less interest in specific attributes is imposing an order on attributes in the query. This has been heavily exploited in [21]: in case of a too large skyline the user provides an order on the attributes used. Finally, the skyline of the subspace omitting the least important attribute(s) is returned instead. Another way of user interaction is given in [26] and [29] where the user is enabled to explore the skyline with respect to different subspaces in an OLAP-like fashion. The so-called *Skycube* precomputes all subspace skylines. It allows for the typical interactions like slicing, drilling down, etc. Especially the concept of *decisive subspaces* in [26] is heavily related to our work. Subspace skylines are projections of the complete skyline onto some important attributes. The decisive subspace(s) for each object contain those attributes explaining why an object is in the skyline. Up to a certain degree this resembles the subspaces chosen in our framework for eliciting trade-off information from the user.

Apart from the application in skyline retrieval the notion of preference elicitation has already attracted attention for personalizing intelligent information systems. [9] gives a good overview of the current state of the art in elicitation methods. Especially, the work in [28] is relevant (but orthogonal) to our approach. Here the user is enabled to critique example items, thus allowing for the implicit generation of preferences best reflecting the criticism. Although no explicit trading of characteristics is supported, we would like to explore an integration of this approach into our framework in future work. Still, we share the idea of a navigational approach to preference-based search.

IV. Theoretical Foundations

As stated before, we consider preferences as weak orders on the domain values with respect to a certain attribute. These are defined as follows:

Definition 1: (Base Preference, Base Equivalence)
Let $D_1, \dots, D_m$ be non-empty sets of m domains (i.e. sets of attribute values) on m attributes $A_1, \dots, A_m$ such that D_i is the domain of A_i. Furthermore, let $O \subseteq D_1 \times D_2 \times \cdots \times D_m$ be a set of database objects.
For any $i \in \{1, \dots, m\}$ *base preference* $P_i \subseteq D_i \times D_i$ is a weak order on D_i, and a *base equivalence* $Q_i \subseteq D_i \times D_i$ is an equivalence relation of those values which are in the same equivalence class with respect to P_i.
To simplify notation, let $D_\mu := \bigtimes_{i \in \mu} D_i$, for any $\mu \subseteq \{1, \dots, m\}$.

Definition 2: (Pareto Dominance, Strict Pareto Dominance)
Let $\mu \subseteq \{1, \dots, m\}$. The *Pareto dominance* relation induced by preferences $\{P_i \mid i \in \mu\}$ and equivalences $\{Q_i \mid i \in \mu\}$, denoted by Par_μ, is the set of pairs $(u, v) \in D_\mu \times D_\mu$ satisfying $(u_i, v_i) \in (P_i \cup Q_i)$, for all $i \in \mu$.
The *strict Pareto dominance* relation induced by $\{P_i \mid i \in \mu\}$ and $\{Q_i \mid i \in \mu\}$, denoted by SPar_μ, is the set $(u, v) \in D_\mu \times D_\mu$ such that $(u_i, v_i) \in (P_i \cup Q_i)$, for all $i \in \mu$, and $(u_i, v_i) \in P_i$, for some $i \in \mu$.

Note that $\mathrm{SPar}_\mu \subseteq \mathrm{Par}_\mu$ and that SPar_μ is a strict partial order on D_μ. Even for small attribute domains and small m, this order contains many symmetrically non-dominated tuples. This is an obvious weakness of the Pareto semantics for preference modeling. We extend this model by allowing for trade-offs. The following definition introduces a preference relation that extends $\mathrm{SPar}_{\{1,\dots,m\}}$.

Definition 3: (Extended Preference and Equivalence)
Let $\mu \subseteq \{1, \dots, m\}$. An *extended preference* with respect to μ is a strict partial order P_μ on D_μ satisfying $\mathrm{SPar}_\mu \subseteq P_\mu$. An *extended equivalence* with respect to μ is an equivalence relation Q_μ on D_μ such that $\mathrm{Par}_\mu - \mathrm{SPar}_\mu \subseteq Q_\mu$.
P_μ and Q_μ are *compatible* if and only if $P_\mu \cap Q_\mu = \emptyset$ and $P_\mu \circ Q_\mu \subseteq P_\mu$ and $Q_\mu \circ P_\mu \subseteq P_\mu$(spanning transitivity).

In the following, instead of $(u, v) \in P_\mu$ (i.e. v dominates u with respect to P_μ) we write $u \lhd_\mu v$. Similary, $(u, v) \in Q_\mu$ is abbreviated by $u \equiv_\mu v$. We write $u \unlhd_\mu v$ if and only if $u \lhd_\mu v$ or $u \equiv_\mu v$. Also, we abbreviate $P_{\{1,\dots,m\}}$ and $Q_{\{1,\dots,m\}}$ by P and Q, respectively. Analogously, $\lhd_{\{1,\dots,m\}}$, $\unlhd_{\{1,\dots,m\}}$, and $\equiv_{\{1,\dots,m\}}$ will be denoted by , , and .

With these definitions, we can derive Pareto skylines by selecting all objects $o \in O$ which are maximal with respect to P. Pareto optimality treats all attributes in the same way and builds the product order out of the base preferences and equivalences. With the strict Pareto semantics also the base preferences and equivalences can be fully reconstructed from the product order. This useful characteristic is referred to as separability [7]. However, when selecting best choices from the skyline sacrificing the separability is a sensible thing to do as the following example shows.

Example (cont.): To get an intuition about trade-offs let us assume the user's monetary value of more space has not yet been specified. Consider the basic preferences: 'the more space the better' and 'the smaller the price the better'. With respect to these preferences a 450 USD 2-bedroom and a 460 USD 3-bedroom apartment (with equivalent values regarding the remaining attributes) are symmetrically non-dominated and would both be part of the skyline. However, for most people an additional room would definitely more than compensate an only slightly higher rental price. Unfortunately this kind of compensation cannot be expressed within the Pareto semantics.

The concept of preference and equivalence trade-offs remedies this shortcoming by allowing to take into account two or more attributes together. Trade-offs *add new relationships* to an extended preference between previously non-dominated tuples following the ceteris paribus semantics. As soon as a trade-off has been specified, however, the separability characteristic is lost for the extended preference order and it can no longer be constructed from the base preferences/equivalences regarding each attribute.

Example (cont.): Assume that for instance the user stated a general equivalence between 450 USD 2-bedroom and a 900 USD 3-bedroom apartments (all else being equal, see Figure 1) and reconsider the example objects from above. If all remaining attribute values are equal, a 460 USD 3-bedroom is clearly better than a 900 USD 3-bedroom apartment, which in turn, by the given trade-off is considered equivalent to a 450 USD 2-bedroom apartment. Now the 460 USD 3-bedroom dominates the 450 USD 2-bedroom apartment, which can thus be removed from the skyline set.

Now we will formally define trade-offs (Definition 4) and specify which relationships have to be added to P under the ceteris paribus semantics with respect to a given trade-off (Definition 5).

Definition 4: (Trade-Offs)
Let $\mu \subseteq \{1, \dots, m\}$ and P_μ, Q_μ a compatible extended preference and equivalence as given by Definition 3.
For any pair of tuples x_μ and y_μ that are incomparable in both P_μ and Q_μ, we call the statement $(x_\mu \lhd y_\mu)$ a *preference trade-off.* Analogously, $(x_\mu \equiv y_\mu)$ is called an *equivalence trade-off.*

With trade-offs, the user introduces new relationships between previously incomparable tuples following the ceteris paribus semantics.

Definition 5: (Incorporating Trade-Offs)
Let P and Q be compatible, extended preferences and equivalences on D and let $\mu \subseteq \{1, \dots, m\}$ and $\overline{\mu} := \{1, \dots, m\} \setminus \mu$.

a) Let $(x_\mu \lhd y_\mu)$ be a preference trade-off for μ. We define an extended preference $P^* \supseteq P$:
   - by extending P to P' by *adding* all relationships $o_1 \lhd o_2$ satisfying the following: $(o_1|_\mu = x_\mu \wedge y_\mu = o_2|_\mu)$ and $(o_1|_{\overline{\mu}} = o_2|_{\overline{\mu}})$
   - and then computing P^*as transitive closure of P'.

b) Let $(x_\mu \equiv y_\mu)$ be an equivalence trade-off for μ. We define $Q^* \supseteq Q$ and $P^* \supseteq P$:
   - by first extending Q to Q' by *adding* all relationships $o_1 \equiv o_2$ satisfying the following: $[o_1|_\mu = x_\mu \wedge y_\mu = o_2|_\mu] \vee [o_1|_\mu = y_\mu \wedge x_\mu = o_2|_\mu]$ and $(o_1|_{\overline{\mu}} = o_2|_{\overline{\mu}})$ and computing Q^* as transitive closure of Q'.
   - And then by computing P^* as transitive closure of $P \cup Q^*$

Please note that the previous definitions base on established work summarized in [2]. However, all those previous approaches relied on actually materializing and extending the preference and equivalence relations P and Q (as well as P^* and Q^*). Those relations quickly grow large and may contain as many relationships as the square of the number of database objects. The benefit of the following approach is to propose an incorporation scheme mimicking the effects of the procedure described in Definition 5, however due to a sensible restriction of the trade-offs, with a drastic reduction of the necessary skyline computation effort.

According to Definition 4, trade-offs can involve an arbitrary number of attributes. However, for user interactions, visualizing more than 2 dimensions is difficult. Moreover, eliciting suitable compromises for multiple attributes is confusing for users in terms of understanding the effect on the skyline. The same holds for chains of compromises spanning over different values in different attributes (e.g., better performance with respect to the first attribute is traded against worse performance with respect to the second, which in turn is traded against some other attribute). Again, though the final effect on the skyline result can be computed, it is rather hard to understand for the user. For these reasons we restrict trade-offs to pairs of attributes only (i.e. trade-offs of the form 'performing worse in *A* can be compensated by performing better in *B*'), and will not allow interleaved chains of trade-offs spanning multiple attributes. These restrictions also have an additional beneficial effect: they allow for decreasing the complexity for incorporating trade-offs and checking consistency as explained in the next section.

Still, finding the new skyline incorporating given trade-offs means to find the maximum objects of the extended preference. Since trade-offs do not add objects to the skyline, the recomputation can be restricted to current skyline objects. In the next section we will show that this only needs comparisons of all objects better than the preferred trade-off values with those worse than the dominated trade-off values. For equivalence trade-offs this has to be done in both directions (cf. Figure 1). Of course, in all these comparisons the ceteris paribus rule has to be obeyed.

A. Selecting Objects and Consistency

Let us consider a user-specified trade-off $(x_\mu \lhd y_\mu)$ or $(x_\mu \equiv y_\mu)$, respectively. Introducing this trade-off means inducing more equivalence and domination relationships on previously incomparable objects into the respective extended preference P and equivalence Q. After integrating a new trade-off, we will call the new extended preference and equivalence P^* and Q^*, respectively. Please note that since we only allow consistent additions, all the previous relationships stay valid, i.e. $P \subseteq P^*$ and $Q \subseteq Q^*$. That means that, if M is the skyline with respect to P and Q, the new skyline M^* with respect to P^* and Q^* can only become monotonically smaller: $M^* \subseteq M$.

The next question is how a skyline object can lose its optimality in P^* so that it has to be removed from the skyline? The following theorem shows how to prune skylines incrementally and proves our pruning method's correctness:

Theorem 1: (Incremental Skyline Calculation)
With skyline set M, the preference and equivalence relations P, Q, $P^*$and Q^* as above and using only trade-offs like in Definition 4, but limited to disjoint pairs of attributes (i.e. $P = P_\mu \times P_{\overline{\mu}}$).

a) Given a preference trade-off $(x_\mu \lhd y_\mu)$, the new skyline M^* incorporating the trade-off can be computed by removing all objects $o \in M$ with $o \trianglelefteq_\mu x_\mu$ if and only if there exists an $o' \in M$ with $y_\mu \trianglelefteq_\mu o'$ and $o \trianglelefteq_{\bar{\mu}} o'$.

b) Given an equivalence trade-off $(x_\mu \equiv y_\mu)$, M^* can be computed by removing all objects $o \in M$ with:
If $o \vartriangleleft_{\bar{\mu}} o'$:
either $o \trianglelefteq_\mu x_\mu$ and there exists $o' \in M$ with $y_\mu \trianglelefteq_\mu o'$, or $o \trianglelefteq_\mu y_\mu$ and there exists $o' \in M$ with $x_\mu \trianglelefteq_\mu o'$.
If $o \equiv_{\bar{\mu}} o'$:
either $o \trianglelefteq_\mu x_\mu$ and there exists $o' \in M$ with $y_\mu \vartriangleleft_\mu o'$, or $o \vartriangleleft_\mu x_\mu$ and there exists $o' \in M$ with $y_\mu \trianglelefteq_\mu o'$, or $o \trianglelefteq_\mu y_\mu$ and there exists $o' \in M$ with $x_\mu \vartriangleleft_\mu o'$, or $o \vartriangleleft_\mu y_\mu$ and there exists $o' \in M$ with $x_\mu \trianglelefteq_\mu o'$.

Proof: Assume a trade-off has been specified between some attributes with indexes i and j and define $\mu := \{i, j\}$. Consider some skyline object $o \in M$, but $o \notin M^*$. Since o is dominated by some object in M^*, but was not dominated by any object in M, a set of new domination relationships must have been introduced with respect to the new trade-off, i.e. especially wrt. the attributes with index in μ. Since trade-offs are defined only between disjoint attribute pairs and the attributes with value in μ have been amalgamated, domination can be checked for μ and $\bar{\mu} := \{1, \dots, m\} \backslash \mu$ separately and then combined using the Pareto semantics, i.e. we have P_μ, $P_{\bar{\mu}}$, Q_μ, $Q_{\bar{\mu}}$, such that P is the Pareto product order of P_μ, $P_{\bar{\mu}}$, Q_μ, and $Q_{\bar{\mu}}$, and Q is the product of Q_μ and $Q_{\bar{\mu}}$. Furthermore, P^* is the Pareto product order of $P_\mu{}^*$, $P_{\bar{\mu}}$, $Q_\mu{}^*$ and $Q_{\bar{\mu}}$. Analogously Q^* is the product of $Q_\mu{}^*$ and $Q_{\bar{\mu}}$. Any object dominating o thus has to dominate it with respect to the projection on attributes with indices in μ and to the projection with indices in $\bar{\mu}$.

Ad a) '=>' Let us take a closer look at a preference trade-off $(x_\mu \vartriangleleft y_\mu)$ as defined above. We know $o \in M$, but $o \notin M^*$. Any domination relationship wrt. o in P^* can only have been introduced by the new domination relationship between amalgamated attributes x_μ and y_μ. Therefore, when projected to the set μ of attributes, o's values have to be worse or equivalent to x_μ, i.e. with Q_μ, P_μ as the extended equivalence, resp. preference regarding only attributes with index in μ: $(o \equiv_\mu x_\mu \vee o \vartriangleleft_\mu x_\mu)$. But in that case we can always compare o to some (artificial) object x constructed as $\forall\, i \in \mu : (x_i := \pi_i(x_\mu) \wedge \forall\, i \in \bar{\mu} : (x_i := \pi_i(o))$ with π_i as the projection on the i-th component in the sense of relational algebra. Now by construction either x dominates o, or x is equivalent to o. Using the trade-off information we also know x to be dominated by some object y constructed analogously as $\forall\, i \in \mu : (y_i := \pi_i(y_\mu) \wedge \forall\, i \in \bar{\mu} : (y_i := \pi_i(o))$. Since the new trade-off information has to be used, in order to dominate o under the ceteris paribus semantics there has to be some (actually existing) object o' either dominating y or equivalent to y. Hence, for o' has to hold $(y \equiv_\mu o' \vee y \vartriangleleft_\mu o') \wedge (y \trianglelefteq_{\bar{\mu}} o')$, i.e. in particular $o \trianglelefteq_{\bar{\mu}} o'$.
'<=' The other direction is clear. If there exits some o' with the above characteristics, then $o \trianglelefteq_\mu x \vartriangleleft_\mu y \trianglelefteq_\mu o'$ holds, but $o \vartriangleleft o'$ can only hold, if also $o \trianglelefteq_{\bar{\mu}} o'$.

Ad b) The proof for the case of new equivalence trade-offs $(x_\mu \equiv y_\mu)$ is similar. Again comparable items x and y have to be constructed like above, but since they have been defined as being equivalent by the user, a domination relationship of some database object $o \in M \backslash M^*$ needs to include a domination relationship either by one of the (artifical) trade-off objects, or the dominating object needs to dominate one of the trade-off objects. Thus, one of the domination/equivalence relationships over/by x or y really has to be a *strict* domination relationship. In summary we are facing a total of five cases where always $o \trianglelefteq_{\bar{\mu}} o'$ has to hold:

$$\begin{aligned}&(o \vartriangleleft_\mu x \equiv_\mu y \trianglelefteq_\mu o') \vee (o \trianglelefteq_\mu x \equiv_\mu y \vartriangleleft_\mu o')\\ &\vee (o \vartriangleleft_\mu y \equiv_\mu x \trianglelefteq_\mu o') \vee (o \trianglelefteq_\mu y \equiv_\mu x \vartriangleleft_\mu o')\\ &\vee (o \equiv_\mu y \equiv_\mu x \equiv_\mu o' \wedge o \vartriangleleft_{\bar{\mu}} o')\end{aligned}$$

And from either case follows $o \vartriangleleft o'$. ■

Some important observations regarding Theorem 1 are as follows:

- Since we restrict trade-offs to pairwise disjoint attribute pairs, Theorem 1 can be iteratively applied for incorporating multiple trade-offs successively. This is because the separability characteristic is sustained between the amalgamated pairs.
- For incorporating a trade-off $(x_\mu \vartriangleleft y_\mu)$, all objects o have to be found with $o \trianglelefteq_\mu x_\mu$ and compared to objects o' with $y_\mu \trianglelefteq_\mu o'$ (or using $\vartriangleleft_\mu$ in either case, respectively). Taking advantage of the restriction of pairwise disjoint trade-off sets and weakly ordered domains, selecting comparison candidates can be performed with a simple SQL query on the attribute score in μ. In case of previous trade-offs on the same μ, their respective candidates have also to be considered to accommodate also transitive relationships.
- After candidates for o and o' have been selected, it has to be checked whether $o \trianglelefteq_{\bar{\mu}} o'$. To perform that test, it is sufficient to compare the attributes in $\bar{\mu}$. If any trade-offs with indices in $\bar{\mu}$ have been specified before, they also have to be checked in this test. However, respecting these trade-offs is simple as their transitive effects are limited to just two attributes and no complex multi-attribute effects are possible due to our previous restrictions.

V. Algorithm

Since trade-offs are generally data-driven and therefore *should not be elicited a-priori*, our algorithm has to incorporate trade-off information interactively and thus personalizes the skyline in a cooperative fashion. The initial skyline is always derived as the maximum objects of the Pareto order with respect to the base preferences $P_1, \dots, P_m$ and base equivalences $Q_1, \dots, Q_m$.

A. Basic Trade-Off System Procedure

The basic procedure for refining skylines with trade-off information is described in Algorithm 1. During the first steps, two appropriate attributes for performing trade-offs are chosen by the user. If he/she cannot decide on appropriate candidates, the system can propose trade-off pairs based on the initial skyline and preferences as covered in detail in section C. Based on the user's feedback the algorithm incorporates the trade-offs into the current skyline and removes all newly dominated objects. The incorporation has to pay special attention to intertwining trade-offs which may form trade-off chains. Section B will describe a basic algorithm for performing this task.

In the following, the condition of an object being "equal to or better/worse than another object with respect to some attribute" is frequently used. To check such a condition efficiently we deploy *scoring functions*: for each attribute a scoring function is derived of the respective base preference/equivalence. Since we rely on weak orders, numerifications of domain values are possible such that the conditions $o_1 \lhd_i o_2 \Leftrightarrow score_i(o_1) < score_i(o_2)$, and $o_1 \equiv_i o_2 \Leftrightarrow score_i(o_1) = score_i(o_2)$ hold [11]. The use of scores allows for an efficient computation of dominance on attribute values as they can directly be stated in SQL and executed by a database engine.

Algorithm 1. Basic Skyline Refinement

```
Method: interactiveTradeOffs(M, P_1, ..., P_m, Q_1, ..., Q_m)
Parameters:
•  A set of skyline objects $M$ on database objects $O$
•  The base preferences $P_1, ..., P_m$ used for computing $M$
•  The base equivalences $Q_1, ..., Q_m$ used for computing $M$
Output:
•  The reduced skyline set $M^* \subseteq M$
•  The set $M_{removed} := M \setminus M^*$
Global Variables:
•  For each pair of attributes involved in trade-offs a list $trade\text{-}off_{i,j}$, $i < j \in \{1, ..., m\}$ containing all trade-offs interactively specified by the user for this respective pair.
•  Scoring functions $score_i$ for each $A_i$, $i \in \{1, ..., m\}$
Procedure:
Initialization:
0.  Create scoring functions for all $A_i$, $i \in \{1, ..., m\}$
Elicit and incorporate suitable trade-offs:
1.  $\{t_1, ... t_v\}$ := elicitTradeOffs($M$) in Algorithm 4
2.  $M^*, M^*_{removed}$ := incorporateTradeOffs($M$, $\{t_1, ... t_v\}$) in Algorithm 2
Visualize result: Provide an overview of the effects of the trade-off incorporation to the user by visualizing $M^*$ and $M^*_{removed}$
3.  If the user does not like the result of the trade-off and wants to undo it, start over with step 1.
4.  $M := M^*$, $M_{removed} := M^*_{removed}$
5.  If the user is already satisfied
    5.1.  Return $M^*$, $M^*_{removed}$ and terminate algorithm
6.  Else Start over with step 1.
```

B. Incorporating Trade-Offs

This section covers incorporating given trade-offs into the skyline. During this process, newly dominated objects have to be removed respecting Theorem 1. At first, preference trade-offs of the form "y_μ is preferred over x_μ" ($x_\mu \lhd y_\mu$) are considered. Given these preference trade-offs, we can identify two sets of skyline objects (see Figure 1): *green* containing all objects which are equal or better than y_μ with respect to attributes in μ; and *red* containing all objects which are equal or worse analogously.

A preference trade-off has to remove all objects from the skyline which are in the *red* set and are equivalent to or dominated by at least one object in the *green* set wrt. the attributes in $\bar{\mu}$. Initially, the set *red* consists of all objects being dominated by x_μ wrt. attributes with index in μ while the set *green* contains those values being equal or better with respect to indices in μ than y_μ. Please note that during the course of user interaction a newly stated trade-off may intertwine with some previously given ones regarding the same μ and may form trade-off chains. For instance, consider two trade-offs $(a_\mu \lhd b_\mu)$ and $(b_\mu \lhd c_\mu)$: As the domination relationship is transitive, also $(a_\mu \lhd c_\mu)$ holds and has to be incorporated. This task has to be independent of the order in which the trade-offs were given. To cater for this requirement, the *red* and *green* candidate sets have to be transitively expanded considering previous trade-off regarding the current μ. In particular this means, if there are preference or equivalence trade-offs pointing to some object of the current *green* set, the set has to be expanded by the previous trade-offs' *green* sets. In the same way, the *red* set has to be expanded by the respective *red* sets of trade-offs originating from an object in the current *red* set.

After establishing the candidate sets, all objects in *green* are compared to those in *red* using function *compare* in Algorithm 3 to determine whether the current green object dominates or is equivalent to the current red one with respect to $\bar{\mu}$. If this is the case, the current red object has to be removed from the skyline.

Algorithm 2. Incorporate Trade-Offs

```
Method: incorporateTradeOffs(M, {t_1, ... t_v})
Parameters:
•  The set of current result objects $M$ originally based on the skyline on database objects $O$
•  A list of $v$ trade-offs $\{t_1, ... t_v\}$
Output:
•  The set $M^*$ with objects affected by the trade-off removed
•  The set $M_{removed}$ containing the removed objects
Procedure:
Incorporate the trade-off: This algorithm uses the set of skyline objects (green set) which may dominate other skyline objects (red set) due to the new trade-off information. For pruning the red set, each green candidate is compared to each red candidate and checked for domination.
0.  $M_{removed} := \emptyset$
Consider all trade-offs:
1.  For each trade-off $t$ in $\{t_1, ... t_v\}$
    1.1.  Select the appropriate $\mu$, $x_\mu$ and $y_\mu$. Also, let $type$ be either $'pref'$ or $'equiv'$ according to $t$.
    Identify green and red sets:
    1.2.  Define the set
          $green := \{o \in M \mid \forall\, i \in \mu: score_i(o) \geq score_i(y_\mu)\}$.
    1.3.  Define set
          $red := \{\, o \in M \mid \forall\, i \in \mu: score_i(o) \leq score_i(x_\mu)\,\}$.
    Respect trade-off chains by expanding the sets:
    1.4.  If there was already a trade-off defined stating a domination/equivalence relationship between some $x'_\mu$ and some $g \in green$ with respect to $\mu$ : $(g \unlhd_\mu x'_\mu)$
          1.4.1.  Expand current set $green$ with set $green'$ computed for preference trade-off between $x'_\mu$ and $g$
```

```
        1.5. If there was already a trade-off defined stating a domination/equivalence relationship (between some $r \in red$ and any $y'_\mu$ with respect to $\mu$ : ( $y'_\mu \trianglelefteq_\mu r$)
             1.5.1. Expand current set $red$ with set $red'$ computed for the preference trade-off between $r$ and $y'_\mu$
    Compare candidates:
    1.6. $\overline{\mu} := \{1, \ldots, m\} \setminus \mu$
    1.7. For $r \in red$ , $g \in green$ with $\exists\ i$ : $g_i \not\equiv r_i$
         1.7.1. If ($type$ = $'pref'$)
                1.7.1.1. If $compare(r, g, \overline{\mu}) \neq 'incomp'$
                         1.7.1.1.1. Add $r$ to $M_{removed}$
         1.7.2. If ($type$ = $'equiv'$)
                1.7.2.1. If $compare(r, g, \overline{\mu}) = 'dominated'$ or ( ($y_\mu \lhd_\mu g$ or $r \lhd_\mu x_\mu$) and $compare(r, g, \overline{\mu}) = 'equivalent'$)
                         1.7.2.1.1. Add $r$ to $M_{removed}$
Remove dominated objects:
2. $M^* := M \setminus M_{removed}$
3. Return $M^*$ and $M_{removed}$
```

Algorithm 3. Comparing candidates

```
Method: compare (candidate r, candidate g, index set $\overline{\mu}$)
Parameters:
•   A candidate object $r \in red$
•   A candidate object $g \in green$
•   The index set $\overline{\mu}$
Output:
•   $'incomp'$, $'dominated'$ or $'equivalent'$
Procedure:
1. $e := true$
2. For $i \in \overline{\mu}$
   2.1. If $score_i(r) > score_i(g)$
        2.1.1. If there is no trade-off involving the i-th attribute and any attribute with index $j \in \overline{\mu}$
               2.1.1.1. Return $'incomp'$
        2.1.2. If g does not dominate r with respect to the order induced on attributes i and j by $P_i$, $P_j$ and all the trade-off information between i and j
               2.1.2.1. Return $'incomp'$
        2.1.3. If $score_i(r) \neq score_i(g)$ then $e := false$
3. If $e = true$
   3.1. Return $'equivalent'$
   3.2. Else return $'dominated'$
```

C. Suggesting Suitable Trade-Offs

This section introduces an algorithm to support the selection of trade-offs. The user may either specify them manually under certain restrictions or may ask the system to propose candidates.

But what characterizes a good trade-off? Introducing trade-offs between two positively correlated attributes shows a low effectiveness. This is because skyline objects with correlated attributes for indexes in μ are incomparable with respect to at least some attributes in $\overline{\mu}$ and generally only few objects can be removed by a trade-off. On the other hand, anti-correlated attributes can be expected to cause a large number of incomparable objects in the skyline. Thus, resolving incomparability between anti-correlated attributes will significantly reduce the skyline size.

Our greedy heuristics is thus based on the current database instance and the degree of anti-correlation between attributes: a trade-off is considered more effective the more objects it will remove from the skyline set, thus enforcing a stronger focus. Thus, a good trade-off should aim for increasing the number of objects in the *red* and *green* sets. To suggest trade-off sets we employ hierarchical clustering [16] (denoted as $cluster_k$ in Algorithm 4) on the data set. The number of clusters is also derived within that cluster algorithm by minimizing the average error between cluster values denoted as $intraError_k$ for a given $cluster_k$.

Furthermore, research in the area of operations research shows that when considering goals in some attribute, users tend to favor a certain point in the feature space and show diminishing preference for more distant values (see e.g. ideal point model [14]). Given that for suggesting a trade-off our algorithm always focuses on two rather anti-correlated attributes, the trade-off choices can more or less be discriminated along a single principal axis. Hence, the ideal point model is indeed applicable (like also used in conjoint analysis) and some typical types of trade-off chains will often occur and should be offered as default options. Namely these are monotonically increasing or decreasing trade-off chains (practically discriminating one of the attributes) and so-called ‘single-peaked preferences’ (that means preferring a certain value and increasingly disliking choices to both sides of that value).

Once the user has selected a pair of attributes and given an opinion on the respective trade-offs (either individually or by accepting one of the default options), the trade-off information is ready for incorporation. But since the median objects are usually in the middle of a cluster, we do not use those median trade-offs directly, but instead incorporate trade-offs between the objects given by the first and third quartile. Thus, the mass of objects is included, whereas the outliers of the respective clusters are ignored.

In a nutshell, the idea for the following elicitation algorithm can be summarized as follows: 1) Find attribute pairs that have not yet been considered and show the highest degrees of anti-correlation. Offer them for selection. 2) Cluster the values with respect to the attribute showing less variation and offer the respective cluster medians as choices to the user. By default suggest monotonically in-/decreasing trade-off chains and single-peaked preferences. 3) Once the user has decided for adequate preference/equivalence relationships extend the trade-offs to the first/third quartile to include the mass of the cluster and return them for incorporation

Algorithm 4. Elicit suitable Trade-Offs

```
Method: elicitTradeOffs (object set M)
Parameters:
•   Set of current result objects M
Output:
•   Set of trade-offs $T_\mu = \{t_1, \ldots t_q\}$, each consisting of the index pair $\mu$, tuples $x_\mu$ and $y_\mu$ and a $type \in \{'pref', 'equiv'\}$
```

```
Procedure:
Elicit Trade-Offs: In this algorithm, a set of trade-offs is elicited from the user. Of course, the user may state the trade-offs directly. Alternatively, based on statistical characteristics of the skyline the system can suggest promising trade-offs.
1.  If user wishes to state trade-offs directly
    1.1. Let the user provide trade-offs $t_1, \dots t_q$
    1.2. Return the set of trade-offs and exit method
Find trade-off attributes by computing the correlation coefficients of all attributes which have not been used in a trade-off before.
2.  Define a candidate index set $I := \{i | 1 \leq i \leq n$ and $\nexists\, 1 \leq j \leq n$ such that $trade\text{-}off_{i,j}$ or $trade\text{-}off_{j,i}$ exists$\}$
3.  Compute all correlation coefficients $cc_{k,l}$, $k < l \in I$ between $score_k(x)$ and $score_l(x)$ with $x \in M$
4.  Create a list $\mu_1, \dots, \mu_r$ by selecting pairwise disjunctive pairs $\{k, l\}$ ordered ascending by their respective $cc_{k,l}$. Only anti-correlated pairs with $cc_{k,l} < 0$ should be considered.
5.  Present $\mu_1, \dots, \mu_r$ to the user (e.g. as attribute pairs like weight vs. displaySize, price vs. cpuSpeed, ...) and let the user select any $\mu$ for trade-off elicitation.
Compute clusters and determine primary attribute of $\mu$
6.  For $k \in \mu$
    6.1. Compute clusters wrt. $score_k(M)$ as $Cluster_k = \{cluster_{k,1}, \dots, cluster_{k,nk}\}$
    6.2. Compute the average intra-cluster error $intraError_k$ for the set of clusters $cluster_k$
7.  Select $primary \in \mu$ as the index with minimal $intraError_k$,
    Select $secondary \in \mu$ as the remaining index.
Compute cluster statistics:
In this step, for each cluster of the primary attribute, the median values with respect to both attributes, as well as the respective first and third quartile are computed.
8.  For each $cluster$ in $Cluster_{primary}$
    8.1. Select the set $M_c$ of all objects $m \in M$ with $score_{primary}(m) \in cluster$
    8.2. Compute $median_{i,primary}$, $median_{i,secondary}$, $quartile1_{i,primary}$, $quartile1_{i,secondary}$, $quartile3_{i,primary}$ and $quartile3_{i,secondary}$ of the respective scores of $M_c$
9.  Generate a list of trade-off object representatives
    $R := \{r_1, r_2, \dots, r_{n\_primary}\}$ such that
    $r_i := [median_{i,primary}, median_{i,secondary}]$ (e.g. weight-displaySize: (7",1kg), (10", 1.5kg), (12", 1.8 kg), ...)
10. Let the user provide preferences and/or equivalences on the representative objects in $R$ (specification may be incomplete).
11. Defaults are monotonic or single-peaked orders on $r_{primary}$.
    Also arbitrary orders on $r_{primary}$ can be provided.
12. $T_\mu := \emptyset$
13. For each relationship $r_i \lhd_\mu r_j$ provided in step 10
    13.1. Create trade-off $t := \{\mu, \{quartile3_{(i,primary)}, quartile3_{(i,secondary)}\}, \{quartile1_{(j,primary)}, quartile1_{(J,secondary)}\}, 'pref'\}$; $T_\mu = T_\mu \cup t$
14. For each relationship $r_i \equiv_\mu r_j$ provided in step 10
    14.1. Create trade-off $t_1 := \{\mu, r_i, r_j, 'equiv'\}$ ; $t_2 := \{\mu, r_j, r_i, 'equiv'\}$; $T_\mu = T_\mu \cup t_1 \cup t_2$
15. Return $T_\mu$
```

VI. EVALUATIONS

A. Evaluation Setup

We will showcase a sample user interaction on a real world dataset and we will also evaluate effects and performance on real as well as on synthetic datasets. In recent research, practical datasets used for evaluation usually consist of rather correlated attributes like e.g. statistics of basketball players (games played, points, assists, etc.) in the NBA dataset in [26] or [21]. This ensures lean skyline sizes, but seems hardly a practical application scenario. In contrast, our two practical datasets comprises real world product data that was extracted from 166 different online shops. They contain 20,537 sale offers for notebook computers and 18,114 digital cameras, respectively. For the notebook dataset each offer is described by a total of 30 attributes. To get a first impression of practical searches we created typical domain preference for 7 selected attributes (1. *price*: the cheaper the better, 2. *CPU type*: an expert opinion weak order on 17 processor types; 3. *weight*: the lighter the better, 4. *display size*: the bigger the better, 5. *RAM*: the more the better, 6. *hard drive size*: the bigger the better; 7. *graphics card memory*: the bigger the better). Posing a skyline query with these simple preferences over 7 attributes resulted already in a Pareto skyline of 182 offers. This skyline contains a wide variety of different systems ranging from light sub-notebooks, cheap mainstream products up to powerful multimedia desktop-replacements – but still, manually browsing through more than 150 offers is a tedious task.

We ran our experiments on an Intel Pentium M 2.0 GHz CPU with 1.5 GB RAM using MySQL 5 CE running on a separate server. We used Sun Java 6 in standard settings.

B. A Showcase Scenario

In our scenario, a user starts interacting with the system to find a new notebook. Assume the user intends to watch DVD movies on the system. Therefore, the display size should be larger while still being light enough for traveling.

First the initial skyline using all known user preferences is computed eliminating all suboptimal offers. If the skyline is too large, the method *elicitTradeOffs* is called and the user can decide to rely on the suggestion system. After computing the pairwise correlation of all attributes, for our notebook dataset $\mu := \{3, 4\}$ (3:*weight* vs. 4:*display size*) is suggested as pair for the first proposed trade-off, because it shows the highest degree of anti-correlation (Pearson correlation -0.89). Assume the user decides to focus on this pair of attributes. The domain values of both attributes are then clustered resulting in 10 clusters with normalized intra-cluster error of 0.09 for *weight* and 9 clusters with intra-cluster error of 0.02 for *display size*. Thus, display size is selected as primary attribute. The visualization of the resulting clusters on μ and their respective sizes (computed in Algorithm 4 step 6) are shown in Figure 3. Now, trade-off candidates are proposed to the user based on the cluster medians. The user can

- decide for a simple monotonic trade-off sequence like 'the bigger screen the better' or 'the lighter weight the better' (in fact discriminating one attribute)

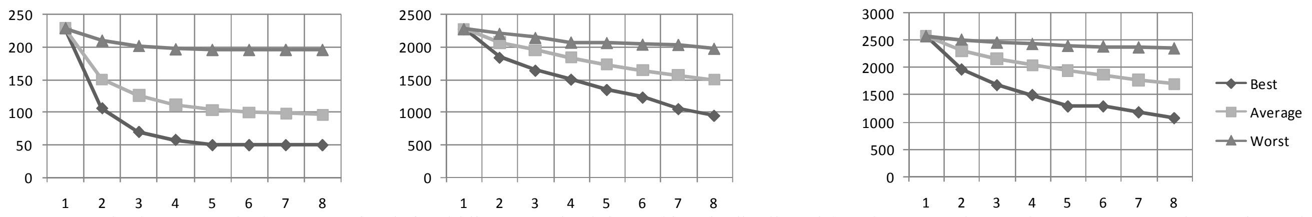


Figure 5. Result Size on Synthetic Data, Left: Zipf, Middle: Normal, Right: Uniformly distributed (x-axis: current interaction user step, y-axis: result set size)

- or decide for some single-peaked preference like 'the ratio between size and price is best for (17'', 4.4 kg)'
- or might assess the trade-offs between pairs of clusters individually.

As an example, focus on the decision between just two clusters C7 and C8 given by (17", 4.4 kg) and (15", 3.0 kg). As the user intends to watch DVDs he/she states a preference for the 17" screen machines weighing 4.4 kg. For the actual incorporation of the trade-off the method *incooperateTradeOff*(x_μ, y_μ) is called. It relaxes the selected trade-off to use the first quartile for y_μ, and the third quartile for x_μ instead of the medians ((17", 4.4 kg) ⊳ (15.2", 2.7 kg)) to include the intended mass of the database objects. This determines to the sets *red* (size 25) and *green* (size 42), both illustrated by red/green rectangles in Figure *4*. During the pairwise comparison of objects in the *red* and *green* sets, 24 *red* objects are marked for removal by the method *compare*, resulting in the new skyline M^* containing only 158 objects (86.8% of the size of M). This effect can be explained by the fact that all removed notebooks were part of the skyline just because they weighed less than their counterparts with bigger screens while being inferior/equal wrt. all other attributes (e.g., more expensive, less memory, etc.).

In fact, if instead for a single trade-off the user had decided for a single-peaked preference favoring 17'' screen notebooks (and thus implicitly decided for relationships between all 9 clusters), the skyline is already reduced to 84 results in a single step (45% of the original skyline). Then, considering for instance a trade-off between the *price* and *RAM* (with a smaller Pearson coefficient of –0.25) and choosing more RAM as the better option further decreases the skyline to 59 (32%) objects. The smaller degree of anti-correlation only allows for smaller reductions. Hence, we see that preferably eliciting information about highly anti-correlated attributes results in a greedy reduction scheme. Of course, the user stays in control during the entire personalization process and at any point can provide arbitrary trade-offs. After every trade-off sample objects of the new skyline and the removed objects are displayed. Thus, the user can roughly gauge the effects. If the user wishes, the algorithm elicits the next trade-offs.

Incorporating trade-offs into skylines does not only cope with the problem of large result set sizes, but also offers a considerable amount of personalization by focusing the results. Figure 7 illustrates the cluster sizes of the notebook dataset before and after incorporating the previously mentioned single-peaked preference favoring 17" screen notebooks. The new skyline is strongly focused on the 17" cluster while many notebooks which are in clusters containing smaller or larger machines lost their optimality and thus have been removed from the skyline. However, another valuable characteristic of the Pareto paradigm can be observed nevertheless as the other clusters still contain some (but fewer) notebooks. The remaining notebooks exhibit outstanding attribute combinations (like remarkable performance or eminently good price) and hence might still be interesting to the user. This behavior is clearly superior to just employing filters on skylines using hard constraints.

C. Real World Data Experiments

We performed experiments on the scenario described in the last section. Based on the notebook and digital camera dataset, all possible sequences of trade-offs between each two adjacent clusters are incrementally incorporated into the skyline, starting with the largest clusters first. For each cluster pair with

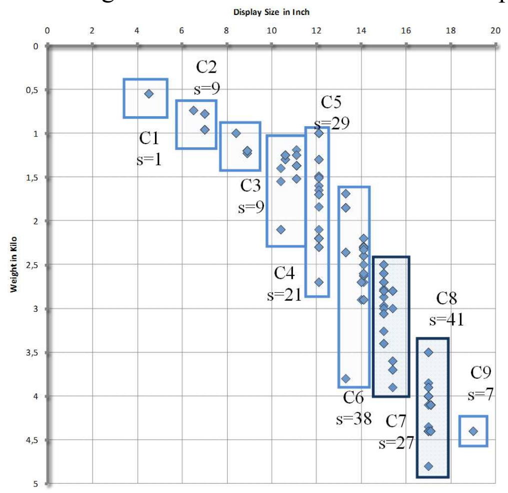


Figure 3. Clusters and sizes on attributes *weight vs. display size*

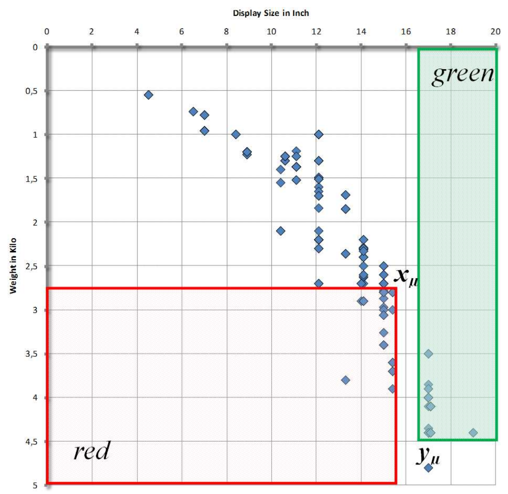


Figure *4*. Red and green sets for (15, 2.7) ⊲ (17, 4.4) representatives x_μ and y_μ, three choices are possible: $(x_\mu \lhd y_\mu)$, $(y_\mu \lhd x_\mu)$ and $(x_\mu \equiv y_\mu)$.

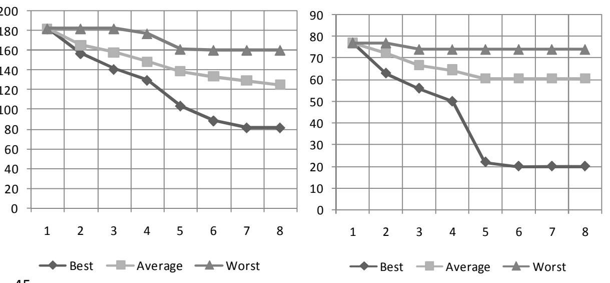
In Figure 6, the average sizes of all these sequences and the respective best/worst sequence is plotted. After incorporating a sequence of 7 trade-offs, the result set was reduced to 66% of its original size on average. The most effective sequence in the sample reduced the result to 44%. Also for the digital camera dataset, significant size reductions could be observed: The average size was reduced to 78% and best reduction resulted in 25%. Please note that eliciting those 7 trade-offs, in the case of single-peaked or monotonic preferences, basically comes down to a single user interaction.

D. Synthetic Data Experiments

We also evaluated our algorithm on larger, synthetic datasets showcasing the effects of different data distributions. Each of the datasets contains 100,000 randomly generated records with 8 attributes each. The attribute's domains contain 15 values which show either Zipfian (Skew -0.5), normal (mean=7, variance=7), or uniform distribution. Note that the respective skyline sizes indeed strongly differ for different distributions. Whereas Zipfian distribution shows an almost manageable skyline size of about 300 objects, normal and uniform distributions result in unmanageable sizes of around 2,500 objects. Again, we simulated all possible trade-off sequences analogously to the previous section.The resulting result set size reductions are plotted in Figure 5. The measured average size for Zipf-distributed data resulted in 42% o
f the original size, for uniformly and normal distributed data in 62% and 61% respectively.

E. Computation Time Evaluation

Skyline computation and incremental skyline refinement algorithms often suffer from poor runtime performance not



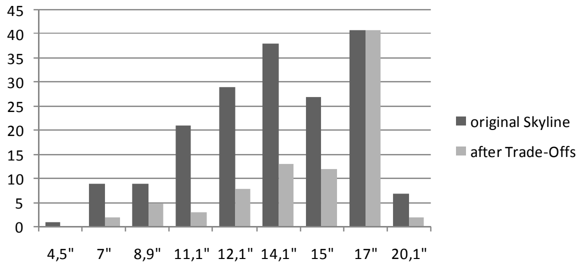


Figure 7. Cluster sizes before and after incorporating a single-peaked trade-off preferring 17" notebooks

allowing for interactive applications. However, in this section we show that our algorithm does not only reduce the resulting skylines to manageable sizes, but is also capable to interact in a cooperative real-time fashion with the user.

During the experiments of the last sections, the average runtime per trade-off was measured (see Table 1). For the real world data sets, a single trade-off could be suggested and incorporated in less than 1/10 of second on average and thus is clearly fast enough for real-time user interactions. Also for our larger synthetic datasets, trade-off incorporation usually could be performed in around one second. One reason for achieving these significant performance improvements is the reduction of pairwise object comparisons (down to 1%) as compared to the baseline algorithm comparing all previous skyline objects pairwise. The necessary comparisons relative to the baseline is also in Table 1.

Table 1. Measured Average Runtime per Trade-Off and Average Percentage of Comparisons

Experiment	*Runtime per Trade-Off*	*% of Baseline Comparisons*
Real "Notebook" Data	0.08 sec	1.1 %
Real "DigiCam" Data	0.03 sec	0.9 %
Zipf Synthetic Data	0.10 sec	6.6 %
Normal Synthetic Data	0.81 sec	7.0 %
Uniform Synthetic Data	1.12 sec	6.3 %

VII. EVALUATIONS

We proposed a necessary extension to the traditional concepts of Pareto skylines by offering users the option to augment and personalize their result sets through trade-off information. The concept of trade-offs is a natural part of people's daily life. Often decision processes are complex and involve several attributes that cannot all be satisfied. A natural heuristic is to focus on only few characteristic attributes at a time, evaluate the possible choices and make up one's mind about potential compromises. This intuitive concept, however, cannot be integrated easily into skyline queries that rely on strict Pareto semantics. In today's systems, those Pareto semantics often lead to unmanageable skyline sizes as practical use-cases show. This is aggravated by the fact that many user preferences invite a high degree of anti-correlation (e.g. high quality vs. low price).

In the course of this paper we have addressed the problem of effective incorporation of users' trade-off decisions into current skyline query processing. Focusing on trade-offs on disjoint pairs of attributes, our algorithm identifies clusters of similar objects with respect to anti-correlated attribute pairs and suggests promising trade-off candidates. The candidates are provided according to a greedy heuristic aiming at large skyline size reductions and thus also minimize the amount of necessary user interaction. Especially offering monotonic and single-peaked preference chains reduces the required effort. Furthermore, our algorithm not only allows reducing skylines to manageable and focused result sets, but also shows an excellent runtime performance. This facilitates real-time interaction cycles cooperatively supporting users in the decision making process.

We performed experiments on two real world datasets: one containing more than 20,000 notebook computer offers and their respective configurations, the other containing around 18,000 digital camera offers. Throughout our experiments the algorithm showed remarkable performance usually more than halving skyline sizes requiring just one single-peaked preference. Moreover, this reduction usually could be computed in less than a second. To further investigate the algorithm's characteristics, we also performed experiments on synthetic datasets containing 100,000 records with different data distributions. Choosing Zipfian, normal and uniform distributions, the claim that skyline sets can easily become unmanageable were confirmed. Nevertheless, in all three cases our algorithm could successfully decrease the sizes down to more manageable levels in acceptable time.

In our future work, we intend to further investigate the process of trade-off suggestion for even more effective skyline reductions. Focusing more closely on the users previous decisions, we plan to adapt clusters and dimensions involved in suggested trade-offs dynamically and thus allow for pruning larger uninteresting parts of the decision space in early interaction steps. For example, considering our notebooks dataset, eliciting if a user would trade more powerful components for smaller device size and weight, we could focus the skyline on the respective device group like sub-notebooks or desktop replacements. Furthermore, we plan to design a sophisticated feedback mechanism giving the user a clear intuition on effects each trade-off decision has on the other attributes. Also, employing dimension reduction strategies may yield further significant improvements in terms of performance and usability.

References

[1] W.-T. Balke, U. Güntzer. Multi-objective Query Processing for Database Systems. *Cnf. on Very Large Databases (VLDB),* Toronto, Canada, 2004.

[2] W.-T. Balke, U. Güntzer, and C. Lofi. Incremental Trade-Off Management for Preference Based Queries. *International Journal of Computer Science & Applications (IJCSA)*, Vol. 4(2), 2007.

[3] W.-T. Balke, U. Güntzer, W. Siberski. Restricting Skyline Sizes using Weak Pareto Dominance. In *Informatik - Forschung und Entwicklung (IFE)*, Vol. 21(3), Springer, 2007.

[4] W.-T. Balke, U. Güntzer, J. Zheng. Efficient Distributed Skylining for Web Information Systems. *Conf. on Extending Database Technology (EDBT)*, LNCS 2992, Heraklion, Crete, Greece, 2004.

[5] W.-T. Balke, J. Zheng, U. Güntzer. Approaching the Efficient Frontier: Cooperative Database Retrieval Using High-Dimensional Skylines. *Conf. on Database Systems for Advanced Applications (DASFAA)*, Beijing, China, 2005.

[6] S. Börzsonyi, K. Stocker, D. Kossmann. The Skyline Operator. *IEEE Cnf. Data Engineering (ICDE)*, Heidelberg, Germany, 2001.

[7] W. J. Bradley, J. K. Hodge, and D. M. Kilgour. Separable discrete preferences. *Mathematical Social Sciences*, 49(3):335-353, 2005.

[8] C. Chan P. Eng, K. Tan. Stratified Computation of Skylines with Partially Ordered Domains. *Conf. on Management of Data (SIGMOD)*, Baltimore, MD, USA, 2005.

[9] L. Chen, P. Pu. Survey of Preference Elicitation Methods. *EPFL Technical Report IC/2004/67*, Lausanne, Switzerland, 2004.

[10] J. Chomicki. Iterative Modification and Incremental Evaluation of Preference Queries. *Symp. on Found. of Inf. and Knowledge Systems (FoIKS)*, Budapest, Hungary, 2006.

[11] P. C. Fishburn. Preference structures and their numerical representations. In *Theor. Computer Science*, 217(2), 1999.

[12] P. Godfrey. Skyline cardinality for relational processing. How many vectors are maximal? *Symp. on Foundations of Information and Knowledge Systems (FoIKS 2004)*. Vienna, Austria, 2004.

[13] P. Godfrey, R. Shipley, J. Gryz. Maximal Vector Computation in Large Data. *Conf. on Very Large Databases (VLDB)*, Trondheim, Norway, 2005.

[14] P. E. Green, A. M. Krieger, Y. Wind. Thirty Years of Conjoint Analysis: Reflections and Prospects. In *Interfaces*, Vol. 31(3), 2001.

[15] S. O. Hansson. Preference logic. *Vol. 4 of Handbook of Philosophical Logic*, 2nd Edition. Kluwer, 2002.

[16] Y. Jung, H. Park, D.-Z. Du and B.L. Drake. A Decision Criterion for the Optimal Number of Clusters in Hierarchical Clustering. In *Journal of Global Optimization*, Vol 25(1), 2003.

[17] R. Keeney and H. Raiffa. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Wiley, 1976.

[18] V. Koltun, C. Papadimitriou. Approximately Dominating Representatives. *Cnf. on Database Theory (ICDT)*, Edinburgh, UK, 2005.

[19] D. Kossmann, F. Ramsak, S. Rost. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. *Conf. on Very Large Data Bases (VLDB)*, Hong Kong, China, 2002.

[20] M. Lacroix, P. Lavency. Preferences: Putting more Knowledge into Queries. *Cnf. Very Large Databases (VLDB)*, Brighton, UK, 1987.

[21] J. Lee, G. You, S. Hwang. Telescope: Zooming to Interesting Skylines. *Conf. on Database Systems for Advanced Applications (DASFAA)*, Bangkok, Thailand, 2007

[22] X. Lin, Y. Yuan, Q. Zhang, Y Zhang. Selecting Stars: The k Most Representative Skyline Operator. *IEEE Conf. on Data Engineering (ICDE),* Istanbul, Turkey, 2007.

[23] M. McGeachie, J. Doyle. Efficient Utility Functions for Ceteris Paribus Preferences. In Proc. of Conf. on Artificial Intelligence and Conf. on Innovative Applications of Artificial Intelligence (AAAI/IAAI), Edmonton, Canada, 2002.

[24] M. Öztürk, A. Tsoukiàs, and P. Vincke. Preference modelling. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, Intl.. Series in OR & Management Science. Springer, 2005.

[25] D. Papadias, Y. Tao, G. Fu, et.al. An Optimal and Progressive Algorithm for Skyline Queries. *ACM Conf. on Management of Data* (*SIGMOD)*, San Diego, USA, 2003.

[26] J. Pei, W. Jin, M. Ester, Y. Tao. Catching the Best Views of Skyline: A Semantic Approach Based on Decisive Subspaces. *Conf. on Very Large Databases (VLDB)*, Trondheim, Norway, 2005.

[27] J, Pei, Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, Q. Zhang. Towards multidimensional subspace skyline analysis. In *ACM Trans. on Database Systems*. Vol. 31(4), 2006.

[28] P. Viappiani, B. Faltings, P. Pu. Preference-based Search using Example-Critiquing with Suggestions. In *Journal on Artificial Intelligence Research (JAIR)*. Vol. 27, 2006.

[29] Y. Yuan, X. Lin, Q. Liu, W. Wang, J.X. Yu, Q. Zhang. Efficient Computation of the Skyline Cube. *Conf. on Very Large Databases (VLDB)*, Trondheim, Norway, 2005.