Preference Trade-Offs – Towards manageable Skylines

Christoph Lofi
Institut für Informationssysteme
Technische Universität Braunschweig
Mühlenpfordtstr. 23 – Braunschweig
lofi@ifis.cs.tu-bs.de

Wolf-Tilo Balke
Institut für Informationssysteme
Technische Universität Braunschweig
Mühlenpfordtstr. 23 – Braunschweig
balke@ifis.cs.tu-bs.de

ABSTRACT
Skyline queries are generally considered as a promising technique for mitigating the challenges posed by the ever growing amount of available information. However, despite nearly a decade of research, the application of the Skyline paradigm in real-world information systems still failed to succeed. This fact was mainly attributed to two major problems: the poor performance of the employed algorithms and the hardly convincing usefulness of Skyline sets as personalized and manageable query results. While most performance issues have nowadays been solved, the semantic issues of the result sets still remain: skyline sets are usually far too large to be manageable and show a very low degree of focus with respect to actual user preferences. This problem is mainly a result of the fairness of the underlying Pareto semantics used by Skyline queries: they provide no means to compensate across different attributes which results in inferior result quality. This paper summarizes the recent efforts in overcoming these semantic shortcoming by introducing the natural and intuitive concept of preference trade-offs to Skyline queries which provides a cooperative user interaction for further focusing and improving the semantic quality of skyline result sets.

Categories and Subject Descriptors
H.3.3 [**Information Systems**]: Information Search and Retrieval.

General Terms
Algorithms, Human Factors

Keywords
Skyline Queries, Preferences, Trade-off Management.

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 a vast amount of information in order to avoid flooding the user with unmanageable query results.

To counter this problem, Skyline queries [1] have been proposed and 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 not inferior with respect to all other attribute, 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 personalizable data filter as dominated objects can be safely excluded from the data collection, resulting in the *Skyline set* of the query. The semantic rationale 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, most works on skyline queries only consider numerical domains and preferences [1] [2] [3], skylining can 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 [4] [5] (see Figure 1 for an example). Furthermore, many of these preferences don't require any user input during elicitation as they can be deduced 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).

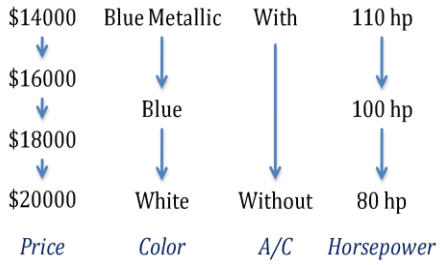

Figure 1. Some preferences within the domain of cars

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 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 large amounts of incomparable objects and generally causes skyline sets to be rather large, especially in the quite common case of anti-correlated

GvD Workshop'10, 25.-28.05.2010, Bad Helmstedt, Germany.

attributes. It has been shown that already for only 5 to 10 attributes, skylines can easily contain 30% or more of the entire database instance [1] [6] [7] [8] which is clearly unmanageable.

In order to ease this problem, various approaches have been proposed which either take advantage of structural properties of yet-to-proven semantic implications [9] [10] [11] [12] or rely on iterative and interactive Skyline computation [13] [14]. However, none of those recent approaches are able to capture the natural semantics of a compensating *trade-off* as used in people's everyday decision processes.

Consider for example three cars: let object A be a 'blue metallic' car for \$18000 and object B be a 'blue' car for \$17000, 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 has the most preferred color. In this scenario, a natural question of a car dealer would be, whether the customer is willing to compromise on those attributes, i.e. if he/she is willing to pay an additional \$1000 for the metallic paint job (we call such a compromise 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*. Still, if some object C like a 'blue' car for \$15000 exists, A and C would still be incomparable as the premium for the metallic color on that car C is larger than the \$1000 the user is willing to pay.

However, actually computing a trade-off enhanced skyline efficiently from the respective enhanced Pareto order is very hard. This is because the modified Pareto order loses some properties (namely the *separability characteristic* discussed in the next section) which, for computing normal skylines, allows for the design of efficient algorithms avoiding the actually underlying object orders altogether. In this paper, we summarize our recent efforts in integrating trade-offs into the skyline computation and show how the computational obstacles of trade-off skylines can be overcome in order to render the Skyline paradigm semantically meaningful.

2. THEORETICAL FOUNDATIONS

In order to realize trade-off skylines, it is necessary to formalize the basic notions and establish the required theoretical ground work. These initial steps are given by our early publications [15] [16] on the topic from an order-theoretical point of view and have been summarized in [17].

In this basic theory, so-called *base preferences* P_1 to P_n on the individual attribute domains are given as strict partial orders. If an attribute value a of the ith-attribute is considered better than b, we write $(a, b) \in P_i$ or $a >_i b$. Analogously, respective compatible *base equivalences* Q_1 to Q_n are given as equivalence relations, representing a user's indifference with respect to some attribute values (e.g. "a black car is as good as a similar blue car", equivalence is in general written as $(a, b) \in Q_i$ or $a \approx_i b$). Finally, we can define a shorthand notation for a being better or equal to b (i.e. $(a, b) \in P_i \vee (a, b) \in Q_i$), denoted as $a \gtrsim_i b$.

These base preferences and base equivalences can be aggregated into a full *Pareto order* P (also called object order), and full *object equivalence* Q using an aggregation function based on the Pareto semantics. These orders encode all domination relationships between all database objects in $O \subseteq D_1 \times \ldots \times D_n$ with D_i being the domain of the ith attribute. I.e. if an object o_1 is dominating an object o_2 with respect to the Pareto semantics, then $(o_1, o_2) \in P$ (or also denoted $o_1 >_P o_2$). The object equivalence Q is computed in a similar fashion. In order to ensure that both aggregation can succeed without any contradictions, a *compatibility criterion* is also defined in [15] for both base preferences and equivalences as well as for the resulting object preferences and equivalences. The two object orders will later be extended with the additional domination relationships inferred from user trade-offs.

Now, the skyline set is given as all those objects which are not dominated by any other object wrt. to the Pareto semantics and users' trade-offs. Using the Pareto order P, this leads to the following definition:

$$Sky := \{o_2 \in R \mid \neg\exists\, o_1 : (o_1, o_2) \in P\};\ o_1, o_2 \in O.$$

Traditional Skyline algorithms not incorporating trade-offs can be based on a different definition of the Skyline set which does not need the object order P. This is because in normal Pareto orders show the characteristic of *separability* [18] with respect to the individual attributes (i.e. the object order without any trade-offs can be decomposed losslessly into its respective base preferences). In particular, this characteristic allows for building efficient skyline algorithms: when testing any two objects A and B for domination, a skyline algorithm does not have to materialize the separable Pareto order P to perform that test. Instead, the test for domination can be replaced by *component-wise* comparisons: if attribute values of object A are better or equal with respect to each attribute than object B's (and 'strictly better' in at least one), then A dominates B and B can be pruned from the skyline.

However, as we introduced the concept of trade-offs in [16] (called *amalgamated preferences*), we have shown that trade-offs will induce addional relationships in the object preference and equivalence order. In the general case, these newly added relationships will violate the separability and will render algorithms based on simple component-wise comparison useless, requireing a full materilization.

This 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 subset of the available attributes. For example, considering the domain of cars, a user could focus on the attributes color and price. A trade-off then describes in a *qualitative* fashion how much a user is willing to sacrifice in some dimension(s) to gain better performance in some other dimension(s) on the basis of a practical example. A possible trade-off t_1 could be: "I would prefer a car for \$18000 with a metallic blue paint job over a car for \$16000 with a plain blue paint job." Then we write $t_1 := \big((\$18000, blue\ metallic) \rhd (\$16000, blue)\big)$.

More formally, a trade-off t is always defined over a set of attributes given by respective indexes denoted as $\mu \subseteq \{1, \ldots, n\}$. Then, a trade-off is a relationship between two tuples $x, y \in \bigtimes_{i \in \mu} D_i$ denoted as $t := (x \rhd y)$.

Working towards a computation scheme for trade-off Skylines, we established rules defining which relationships are added by each trade-off. Basically, to integrate a trade-off into the object order P, a trade-off t will induce a trade-off domination relationship between any object o_1 and o_2 (denoted as $o_1 >_t o_2$) for which o_1 is better or equal than x with respect to μ, y is equal or better to o_2 with respect to μ, and finally o_1 is better or equal than o_2 with respect to all attributes not in $\bar{\mu}$, whereas $\bar{\mu}$ contains all

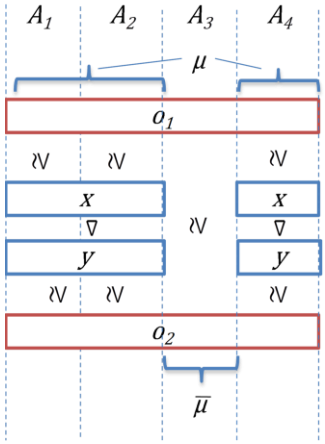

Figure 2. Trade-Off domination relationship $o_1 \succ_T o_2$

attribute indices not in μ. Refer to Figure 2 for a visualization of this concept.

Of course, all domination relationships are transitive and *P* must finally be closed transitively. This transitive closure may generate complex domination relationships spanning several trade-offs which we call *trade-off chains* (discussed in the next section).

These considerations lead to an incremental computation scheme [16]: basically, trade-off Skylines can be computed starting with the initial Skyline based on *P* and then be refined by incrementally adding individual trade-offs. Each trade-off induces additional relationships into *P* (then denoted as P^*, see Figure 4), and thus the newly dominated objects are to be removed from the previous Skyline result set. This allows for an interactive compuation of the trade-off skyline in direct cooperation with the user.

However, this computation approach still relies on materializing the object orders and proved to be computationally prohibitive in later experiments.

3. SIMPLE TRADE-OFF SKYLINES

As mentioned in the previous section, the main reason for the need of an inefficient materialization of the object order is the loss of the object order's separability. To design better performing algorithms, a fundamental technique is to rely as far as possible on basic component-wise attribute comparisons and just materialize a minimal subset of the total object order *P*. A general algorithm based on this idea is covered in the next section. In this section, we will present a trade-off skyline computation algorithm initially presented in [19] which additionally restricts the semantics of allowed trade-offs in order to enforce object orders which can very easily be materialized partially.

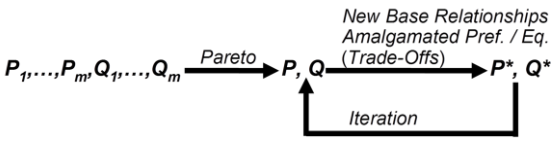

Figure 4. Incrementally extending the object order

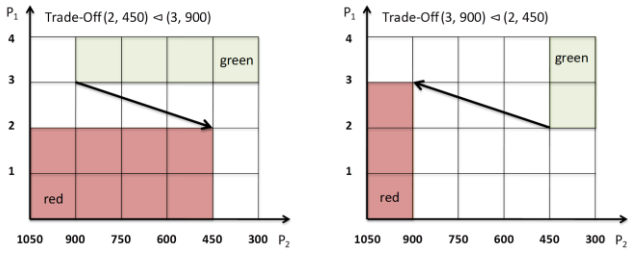

Figure 3. Red and Green sets for simplified trade-off computation

The underlying rationale is as follows: most problems with computing trade-off enhanced product orders arise from *trade-off chains*, i.e. domination relationships which are induced by not one trade-off alone but are the result of the transitive closure of multiple trade-off induced domination relationships and ordinary Pareto domination relationships. Thus, a possibility for simplification of the trade-off computation problem is to restrict the complexity of the possible trade-off chains. A good heuristic which works well with many real-world scenarios is to allow only trade-offs on pairs of two antagonistic attributes each (e.g. power and fuel efficiency for cars, or display size and weight for laptop computers). Furthermore, those attribute pairs must be disjoint. If both conditions are fulfilled, all resulting trade-off chains are of a simpler nature and thus allow for algorithms showing high performance due to the possibility of primarily relying on attribute comparisons.

The resulting algorithm can be summarized as follows: first, we start with the Pareto skyline without any trade-offs. Trade-offs are then incrementally elicited and the skyline is refined accordingly. This refinement takes advantage of the restrictions on two attributes: a trade-off ($x \rhd y$) can be visualized on a two-dimensional plane by two areas (see figure Figure 3): the area containing all objects projected on the respective trade-off attributes dominating the head x of the trade-off (denoted the *green* set, objects in this area are potential candidates for dominating other objects via the trade-off) and the area being dominated by the tail y of the trade-off (called the *red* set containing all objects which are potentially dominated by objects in the green set). The objects in the red and green sets can easily be selected from the database using a simple SQL statement. Afterwards, each object in the green set needs to be compared to all objects in the red set; if the green object dominates or is equal to any red object with respect to all attributes (expect the two attributes used to define the trade-off), the red object is dominated by the green object via the trade-off and is thus removed.

In order to capture trade-off chains, multiple of these area checks have to be executed consecutively. Usually, the red and green sets contain only few objects, thus the runtime measured in our experiments is usually well below 500ms.

Effects on the Skyline

During our work on [19], we also evaluated the effects of simple trade-offs on the properties of resulting trade-off refined skyline sets. As an example, we will present an evaluation performed on a real-life data set containing 20,537 sale offers for notebook computers. After providing 7 base preferences, the resulting Pareto skyline was computed containing still 182 notebook offers, in-

cluding all types of notebooks from lightweight netbooks to large and heavy desktop replacements.

In this evaluation, we assume that the user is willing to sacrifice mobility in favor for performance and display size. Using a trade-off elicitation heuristic [19], this results in 13 trade-offs (which have been inferred from just two user interactions). Incorporating these trade-offs reduces the skyline set to just 59 notebooks (32% of the original skyline's size). Furthermore, the *focus* and quality of the result is increased: instead of containing large numbers of notebooks from all different categories, the result focuses on 17"-screen notebooks (a cluster containing the majority of all desktop replacement machines) while many of the smaller notebooks have been removed as a result of the provide trade-offs (see Figure 5). However, in contrast to simple filters, this refinement retained important characteristics of the Pareto semantics as all remaining notebooks, even those in the cluster of smaller netbooks, are especially interesting and non-dominated objects which are potentially still a good deal even after the user refined his intentions by proving trade-offs.

4. FULL TRADE-SKYLINES

In this section, we will present our latest works on trade-off enhanced skyline computation. In contrast to the simplified algorithm of the previous section, no semantic restrictions are forced on the structure of possible trade-offs, i.e. trade-offs are possible on any combination and number of attributes and may thus form trade-off chains of high complexity. Especially, this enforces thorough considerations with respect to *consistency* of trade-offs.

In previous sections, we implicitly assumed that trade-offs are provided in a consistent and non-contradicting fashion. However, this is not necessarily true as users may easily specify trade-offs which will form a cyclic trade-off chain and render the whole set of trade-offs inconsistent and unusable. Obviously, inconsistencies must be detected and resolved. This job is performed by our consistency check algorithms presented in [20] [21]. In a later work, this algorithm has been further expanded to also cover the actual computation of trade-off skylines [22].

The base idea of our full trade-off skyline computation and consistency check algorithm is to abstract from materializing the prohibitively complex object order and build a tree data-structure (called *TTree*) which represents only possible trade-off chains (in a generic way independent of the actual content of the database). If any contradictions can be detected within this data structure, the provided trade-off set is rejected as being *inconsistent*. If no inconsistency is found, the tree can later be used for the actual skyline computation.

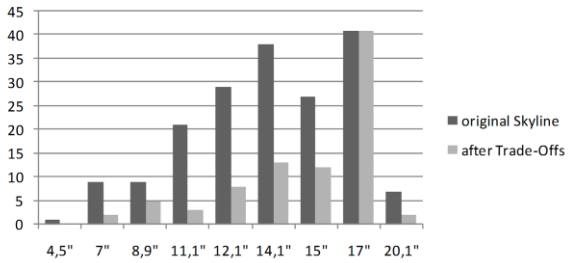

Figure 5. Skyline Result Focus
Notebook dataset; user is looking for a desktop replacement
y-axis: number of result objects, *x-axis*: display size of notebook

To better explain the concept of TTrees, consider Figure 6 showing a TTree build from three trade-offs t_1 to t_3: on the lowest branch t_3, we can see that a trade-off chain from t_3 to t_1 over t_2 and again t_1 is possible (and of course, also all shorter chains).

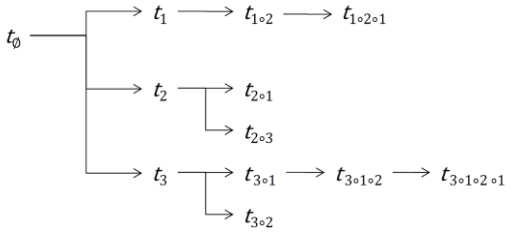

Figure 6. Basic TTree for $T = \{t_1, t_2, t_3\}$

To decide if two given trade-offs t_1 and t_2 can be chained in general (written as $t_1 \circ t_2$), their respective tail and head part is tested for Pareto domination ($y_1 \gtrsim_{\mu_1 \cap \mu_2} x_2$, see Figure 7). This concept can also be extended to cover trade-off chains involving an arbitrary number of trade-offs (please refer to [22] for the full theoretical foundations on general trade-off chains).

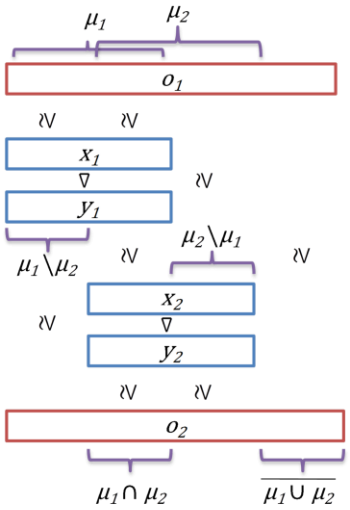

Figure 7. $o_1 >_T o_2$ with two trade-offs $t_1 \circ t_2$

The TTree is constructed incrementally, starting with an empty root node. Then, trade-offs are added to the tree one by one by testing for each existing node (representing a trade-off chain each) whether the current trade-off can be appended to that node or not to form a longer trade-off chain (using the extended chaining criteria from [22]). If the node can be appended, the current branch is checked for consistency (see [21]). After adding all provided trade-offs, also all possible trade-off chains have been constructed.

Unfortunately, a TTree constructed in such a fashion grows quickly in size. This effect can mainly be accounted to redundantly stored information as several trade-off chains can carry the same or even weaker semantic information than other trade-off chains. A further optimization technique during tree construction is thus to avoid or remove all trade-off chains which are *subsumed* by another trade-off chain (i.e. carry redundant information) [22]. This technique significantly decreases the size of TTrees (then

Figure 8. Interface for example comparisons

called *pruned TTree*), especially for extreme trade-off sets which would otherwise generate overly complex chains.

Computing the Skyline

Each trade-off chain in the TTree can be represented by a single, integrated trade-off [22]. By generating the TTree, the complex problem of domination relations via longer trade-off chains can thus be reduced to simple domination tests involving just single trade-offs (each representing a possible chain).

Based on this consideration, computing the trade-off skyline can be performed as follows: first, the Pareto skyline without any trade-offs is computed. Based on the resulting skyline set, additional trade-offs are elicited from the user and a respective pruned TTree is generated. Each branch (i.e. possible trade-off chain) of the TTree is condensed into a single trade-off and stored in an indexed data structure [22]. Finally, a standard skyline algorithm which is based on object comparisons can be applied on the initially computed Pareto skyline (e.g. Block-Nested Loops (BNL) Algorithm [1]). However, instead of using a simple Pareto domination criterion when testing for dominance of objects o_1 and o_2, the domination test is replaced by two index look-ups on the index of trade-offs previously generated from the TTree: it has to be tested if there is a trade-off such that its head is dominated by o_1 and its tail dominates o_2 (as shown previously in Figure 2). If such a trade-off can be found, o_2 is dominated by o_1 via this trade-off and removed.

To further improve the runtime performance of our trade-off skyline computation scheme, we also designed a family of parallel skyline algorithms based on the BNL algorithm [23]. These algorithms are perfectly suited to compute our trade-off skylines efficiently on modern multi-core hardware.

Trade-Off Elicitation

Up to now, we assumed that users will directly provide trade-offs to the system. However, this task requires a significant cognitive effort. To overcome this disadvantage, we designed two trade-off elicitation heuristics: the first heuristic initially published in [19] proposes trade-offs to the user who may accept or dismiss the suggestions. A major cause for unmanageable large skyline result sets is object incomparability resulting from anti-correlated attributes. Accordingly, this heuristic analyzes the correlation and clustering properties of the data objects to suggest trade-offs which will minimize the incomparability between strongly anti-correlated attribute clusters.

Our second heuristic [24] is based on simple item comparisons: the user is presented with two example items from the database (e.g. most popular items, or most distinctive items, etc; see Figure 8 for an example) and simply decides which of the two items he prefers. This decision is used by the heuristic to deduce a set of so-called conceptional trade-offs which generalize the decision. The conceptual trade-offs resulting from our heuristic approach can be proven to be qualitative representations of more complex quantitative utility functions as used in ranking or top-k retrieval, while at the same time avoiding the elicitation overhead generally imposed by quantitative approaches.

5. SUMMARY AND CONCLUSIONS

In paper we summarized our efforts to extend the well-established skyline paradigm with the concept of *preference trade-offs*, which allow for compensation between individual attribute dimensions in a *qualitative* fashion. Compensating (or trading) between different choices is indeed a very natural concept, frequently encountered in every days' decision processes. Trade-offs help to increase the focus and manageability of skylines without any arbitrary assumptions beyond the control of the user. However, up to now it was not possible to augment the strict Pareto semantics of traditional skylines with the compensating semantics of trade-offs.

In our previous works, we established the theoretical foundations for trade-off enhanced skylines from an order-theoretical point of view. Unfortunately, we could prove that trade-offs will break the convenient property of separability when performing object domination tests, an integral component within any existing skyline algorithm. Accordingly, at least some parts or even the whole object order must be materialized. In order to avoid the full materialization of the object order, we first presented a simplified trade-off computation scheme which enforces some semantic restrictions on possible trade-offs and which resulted in a very efficient algorithm. Finally, we presented a general solution to the trade-off enhanced skyline challenge: our latest algorithms cover the full semantic expressiveness of preference trade-offs while at the same time providing good runtime performance. Furthermore, these latest algorithms also deal with detecting inconsistencies and contradicting user input and have even been extended with elicitation heuristics reducing the cognitive load during trade-off elicitation.

6. ACKNOWLEDGMENTS

Part of this work was supported by a grant of the German Research Foundation (DFG) within the Emmy Noether Program of Excellence.

7. Bibliography

[1] Stephan Börzsonyi, Donald Kossmann, and Konrad Stocker, "The Skyline Operator," in *Int. Conf. on Data Engineering (ICDE)*, Heidelberg, Germany, 2001.

[2] Donald Kossmann, Frank Ramsak, and Steffen Rost, "Shooting Stars in the Sky: An Online Algorithm for Skyline Queries," in *Conf. on Very Large Data Bases (VLDB)*, Hong Kong, China, 2002.

[3] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger, "An Optimal and Progressive Algorithm for Skyline Queries," in *Int. Conf. on Management of Data (SIGMOD)*, San Diego, USA, 2003.

[4] M. Lacroix and P. Lavency, "Preferences: Putting more Knowledge into Queries," in *Conf. Very Large Databases (VLDB)*, Brighton, UK, 1987.

[5] Chee Yong Chan, Pin-Kwang Eng, and Kian-Lee Tan, "Stratified Computation of Skylines with Partially Ordered Domains," in *Int. Conf. on Management of Data (SIGMOD)*, Baltimore, MD, USA, 2005.

[6] Wolf-Tilo Balke, Jason Zheng, and Ulrich Güntzer, "Approaching the Efficient Frontier: Cooperative Database Retrieval Using High-Dimensional Skylines," in *Conf. on Database Systems for Advanced Applications (DASFAA)*, Beijing, China, 2005.

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

[8] Parke Godfrey, Ryan Shipley, and Jarek Gryz, "Preference Structures and Their Numerical Representations," in *Conf. on Very Large Databases (VLDB)*, Trondheim, Norway, 2005.

[9] Wolf-Tilo Balke, Ulrich Güntzer, and Wolf Siberski, "Restricting Skyline Sizes using Weak Pareto Dominance," *Informatik - Forschung und Entwicklung (IFE)*, vol. 21, no. 3, 2007.

[10] C.-Y. Chan, H.V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *Int. Conf. on Management of Data (SIGMOD)*, Chicago, USA, 2006.

[11] Y. Yuan et al., "Efficient computation of the skyline cube," in *Int. Conf. on Very Large Data Bases (VLDB)*, Trondheim, Norway, 2005.

[12] Jian Pei, Wen Jin, Martin Ester, and Yufei Tao, "Catching the best views of skyline: a semantic approach based on decisive subspaces," in *Int. Conf. on Very Large Databases (VLDB)*, Trondheim, Norway , 2005.

[13] Jongwuk Lee, Gae-won You, and Seung-won Hwang, "Telescope: Zooming to Interesting Skylines," in *Conf. on Database Systems for Advanced Applications (DASFAA)*, Bangkok, Thailand, 2007.

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

[15] Wolf-Tilo Balke, Ulrich Güntzer, and Christoph Lofi, "Incremental Trade-Off Management for Preference Based Queries," *Intl. Journal of Computer Science & Applications (IJCSA)*, vol. 4, no. 2, 2007.

[16] Wolf-Tilo Balke, Christoph Lofi, and Ulrich Güntzer, "User Interaction Support for Incremental Refinement of Preference-Based Queries," in *Int. Conf. on Research Challenges in Information Science (RCIS)*, Ouarzazate, Morocco, 2007.

[17] Wolf-Tilo Balke, Christoph Lofi, and Güntzer Ulrich, "Eliciting Matters - Controlling Skyline Sizes by Incremental Integration of User Preferences," in *Int. Conf. on Database Systems for Advanced Applications (DASFAA)*, Bangkok, Thailand, 2007.

[18] Sven Ove Hansson, "Preference Logic," *Handbook of Philosophical Logic* , vol. Volume 4 , no. 2, pp. 319-393, 2002.

[19] Christoph Lofi, Wolf-Tilo Balke, and Ulrich Güntzer, "Efficient Skyline Refinement using Trade-Offs," in *3rd Intl. Conf. on Research Challenges in Information Science (RCIS)*, Fez, Morocco, 2009.

[20] Christoph Lofi, Ulrich Güntzer, and Wolf-Tilo Balke, "Efficiently Performing Consistency Checks for Multi-Dimensional Preference Trade-Offs," in *IEEE Int. Conf. on Research Challenges in Information Science (RCIS)*, Marakech, Morocco, 2008.

[21] Christoph Lofi, Ulrich Güntzer, and Wolf-Tilo Balke, "Consistency Check Algorithms for Multi-Dimensional Preference Trade-Offs," *International Journal of Computer Science & Applications (IJCSA)*, vol. 5, no. 3b, pp. 165 - 185, 2008.

[22] Christoph Lofi, Ulrich Güntzer, and Wolf-Tilo Balke, "Efficient Computation of Trade-Off Skylines," in *13th Intl. Conf. Extending Database Technology (EDBT)*, Lausanne, Switzerland, 2010.

[23] Joachim Selke, Christoph Lofi, and Wolf-Tilo Balke, "Highly Scalable Multiprocessing Algorithms for Preference-Based Database Retrieval," in *15th International Conference on Database Systems for Advanced Applications (DASFAA)*, Tsukuba, Japan, 2010.

[24] Christoph Lofi and Wolf-Tilo Balke Ulrich Güntzer, "Eliciting Skyline Trade-Offs using Example-Based Heuristics for E-Commerce Applications," in *Technical Report on http://www.ifis.cs.tu-bs.de/staff/christoph-lofi*, 2010.