Exploiting Indifference for Customization of Partial Order Skylines

Wolf-Tilo Balke[1] Ulrich Güntzer[2] Wolf Siberski[1]
[1]L3S Research Center
University of Hannover, Germany
{balke, siberski}@l3s.de
[2]Institute of Informatics,
University of Tübingen, Germany
ulrich.guentzer@informatik.uni-tuebingen.de

Abstract

Unlike numerical preferences, preferences on attribute values do not show an inherent total order, but skyline computation has to rely on partial orderings explicitly stated by the user. In such orders many object values are incomparable, hence skylines sizes become unpractical. However, the Pareto semantics can be modified to benefit from indifferences: skyline result sizes can be essentially reduced by allowing the user to declare some incomparable values as equally desirable. A major problem of adding such equivalences is that they may result in intransitivity of the aggregated Pareto order and thus efficient query processing is hampered. In this paper we analyze how far the strict Pareto semantics can be relaxed while always retaining transitivity of the induced Pareto aggregation. Extensive practical tests show that skyline sizes can indeed be reduced about two orders of magnitude when using the maximum possible relaxation still guaranteeing the consistency with all user preferences.

1. Introduction

Retrieval techniques taking human preferences into account play an essential role in today's information systems. As a result, database retrieval has moved beyond pure SQL-style retrieval, i.e. exact matches, towards more powerful ranked retrieval techniques. All of these techniques involve scoring hits according to the users' preferences. As these are expressed with respect to a set of predicates, query processing becomes a multi-objective optimization problem. For instance, top-k approaches use a numerical utility function to compensate between (individually weighted) query predicates. However, practical applications show that such numerical compensation often causes problems in the querying process. This is because users cannot sensibly decide a-priori for a suitable function or the most adequate weightings to express their information needs, i.e. without having at least an overview over the contents of the database.

As remedy to this shortcoming, the notion of skyline queries was developed, see e.g. [18], [7], [9], [15]. Here, query predicates are considered to be completely independent. Consequently, no weighting function combining individual predicate scores can be used. Instead, all possibly optimal objects are returned to the user, based on the notion of Pareto optimality. It states that all objects have to be considered optimal with respect to a collection of objects and a set of preferences, if they are not dominated by any other object. To be more exact, an object is considered to dominate another object if it is considered better in one preference and better or at least equivalent with respect to all other preferences. The skyline paradigm has been proven to be quite useful in practical applications and has already inspired a large number of applications in a lot of different areas, e.g. data warehousing [22] or location-based services [14].

Approaches in database retrieval up to now focused on efficient algorithms for skyline computation over numerical predicates, where a preference always defines a total order on predicate values, see [7], [18], [21], [4]. Due to using only total orders and the intrinsic transitivity of the combined domination relation, big parts of the database can efficiently be pruned during query processing, see e.g. [4] or [7]. But total orderings cannot really capture user preferences for non-numeric predicates. For instance a user might be indifferent between buying a blue or red car, but might prefer both over a yellow car. Therefore, novel algorithms have been devised enabling the processing of partial order preferences as well, e.g. [15], [10], [3], [8], [18]. Since partial orders can introduce a large amount of incomparability between objects, these algorithms all face the problem of needing a transitive aggregation to allow for efficient query processing.

The straightforward option is given by strict Pareto aggregation that is always transitive for transitive base preferences. But this approach considerably increases the average sizes of the skyline. In fact, [6] shows that

the average number of objects in strict Pareto skylines grows exponentially with the number of query predicates. This is supported by experiments in [4] showing that for independently distributed data skyline sizes are getting unmanageable for as little as six query predicates, usually retrieving around 25% of the whole database. Therefore, it is very attractive to relax the strict Pareto aggregation and exploit some more user-provided information about incomparable objects. We allow users to explicitly state equivalences between attribute values in partial order preferences and then customize the Pareto aggregation accordingly.

In this paper we investigate how far the Pareto aggregation can be relaxed without sacrificing transitivity and thus still allowing for efficient evaluation algorithms with pruning techniques.

2. Query Processing with Partial Order Skyline Semantics

The efficient processing of skyline queries usually relies on a sorting of database objects implicitly induced by the preferences with respect to each query predicate. Recent instance-optimal algorithms on total order preferences (as presented in [4]) show that arranging database objects in sorted lists for each single preference allows for effective pruning: it is sufficient to scan lists until a common object is found, because by exploiting transitivity within the sorted list this object has to dominate all objects remaining in the lists. Thus, all yet unseen objects (usually a large portion of the database) can be pruned.

A similar result does hold for partial order preferences. Also in this case sorted lists can be constructed using topological orderings of the partial order preference graphs, but the condition for pruning is somewhat more complex, because in topological orders no transitivity can be readily exploited. Consider for instance the two preferences P_1 (about car types) and P_2 (about colors) in Figure 1. Both preferences induce a sorting on the database objects. Table 1 shows the corresponding ranked lists for some sample objects o_i.

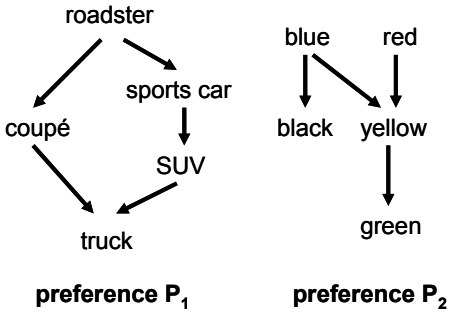

Figure 1. Example preferences

In the totally ordered case each such sorting is unique (except for ties, which can be sorted in arbitrary order). But for the partially ordered case a lot of different topological orderings exist, due to incomparability between objects. For example, regarding predicate 1 we could have put o_4, the black coupe, on rank 1 instead of o_1, the red sports car, because they are incomparable in predicate 1. This also has an impact on how far we have to access the lists to guarantee correct result sets. We find that object o_1 occurs in both lists after accessing the first two ranks, but we still cannot prune the rest of the lists, because for example o_4 is not dominated by o_1, but has not been accessed yet. Moreover, how should we deal with an object like a white limousine, not even occurring in any of the preferences? Formally it is a valid skyline object, because it is not dominated. Thus, there can be no guarantee that all relevant objects for the skyline computation have been seen by just ensuring that there exists a common object in both lists.

The problem here is that working with partially ordered sets there is no way to arrange objects linearly such that every object dominates all its successors. Consider for example preference P_1 in Figure 1: the coupé and the sports car and the coupé and the SUV are incomparable and due to incomparability not being an equivalence relation (the sports car and the SUV are comparable, although both are incomparable to the coupé) cannot be ordered uniquely expressing this relation. We have investigated a sufficient pruning condition for partial order preferences in [3]. There analytical proof is presented that no relevant skyline object can be missed (i.e. there can be no false negatives), if a certain set of objects (called *l*-cut) has been accessed before pruning. But this efficient algorithm only works for transitive aggregations; otherwise some unseen object, although not part of the skyline itself, may still dominate some already seen object. Thus, although there will never be false negatives, in intransitive aggregations there is no way of eliminating false positives once the pruning has taken place. Maintaining a transitive aggregation hence is vital for the efficiency of query processing.

Table 1. Ranked object list example

rank	predicate 1		predicate 2	
	oid	value	oid	value
1	o1	sports car	o3	blue
2	o3	SUV	o1	red
3	o4	coupé	o4	black
4	…	…	…	…

3. Pareto Aggregation and Transitivity

Since skylines merely based on domination information expressed by strict partial order base preferences tend to be too big for practical use, in this section we want to discuss possible customizations of the Pareto optimal result set that users can create by adding equivalences to reduce incomparability between skyline result objects. Generally speaking all such customizations decrease the skyline result size and the more equivalences are specified, the more the set is reduced.

3.1. The Customized Pareto Aggregation

With respect to Pareto optimal results on *m* query predicates considering base preferences $P_1,\ldots,P_m$ for the respective domains only give part of the domination information. Adding equivalence information $eq_1,\ldots,eq_m$ between objects in each predicate is an important tool for the user to explicitly express indifference between choices or equally desirable object values. We define the class of customized Pareto optimal results as:

Definition 1: (Customized Pareto Aggregation)
Let O be a set of database objects and $P_1,\ldots,P_m$ a set of strict partial order preferences about attribute values on *m* different predicates describing objects in O. Each preference P_i is based on a partial order H_i on attribute values $att_i(o)$ with respect to the i-th predicate of objects $o \in O$: $\mathbf{(o_1, o_2) \in P_i}$ **, iff** $\mathbf{att_i(o_1) <_{Hi} att_i(o_2)}$.

Let further $eq_1,\ldots,eq_m$ denote *m* equivalence relations such that $\forall\ 1 \le i \le m$: $P_i \cap eq_i = \emptyset$, i.e. users cannot be indifferent between two objects, if they have expressed a preference between them and vice versa. Then we can define Pareto aggregation customized by equivalence relations as:

$\mathbf{Pareto(O, P_1,\ldots,P_m, eq_1,\ldots,eq_m) := \{(o_1, o_2) \mid \forall\ 1 \le i \le m: (o_1, o_2) \in P_i \cup eq_i\ \wedge \exists\ 1 \le j \le m: (o_1, o_2) \notin eq_j\}}$

The definition means that an object o_1 dominates an object o_2, if and only if it is explicitly preferred to o_2 with respect to at least a single preference and either is also preferred to o_2 with respect to all remaining preferences, or can be considered equally desirable using the binary relation eq_j.

Example: Consider the numerical case where each query predicate is evaluated over the interval [0,1] assigning a score value to each object's attribute values based on the respective degree of match. The result of any *m*-dimensional skyline query like e.g. [7] or [21], can be easily modeled by Pareto(O, $P_1,\ldots,P_m$, $eq_1,\ldots,eq_m$) by choosing suitable scoring functions $S_1,\ldots,S_m$, where S_i: $O \rightarrow [0,1]$, and defining he preferences $P_1,\ldots,P_m$ as $P_i := \{(o_1, o_2) \mid S_i(o_1) < S_i(o_2)\}$ and equivalence relations $eq_1,\ldots,eq_m$ as $eq_i := \{(o_1, o_2) \mid S_i(o_1) = S_i(o_2)\}$.

This model is also suitable for expressing the different semantics used for defining the Pareto-optimal set in literature. The question of customizing Pareto order can be brought down to the question of how to deal with equivalence and incomparability. For any two objects o_1 and o_2 in the classic definition of Pareto order two objects have to be considered equivalent with respect to the i-th predicate, if they feature *exactly the same* i-th attribute value, i.e. the objects cannot be distinguished as far as this attribute is concerned.

However, in the case of indifference between two objects their equivalence (or more precisely which objects to group into equivalence classes) is problematic. Conditions for defining objects' equivalence range from 'objects really have to have exactly the same predicate value to be considered equivalent' to 'every two predicate values that are equal or incomparable can be considered equivalent' (i.e. $eq_i := \{(o_1,o_2) \mid \neg(att_i(o_1) <_{Hi} att_i(o_2)) \wedge \neg(att_i(o_2) <_{Hi} att_i(o_1))\}$). The evaluation of complex preferences under Pareto semantics in database retrieval spans the entire range: the first definition has been called 'Pareto accumulation' (cf. [15]), whereas the latter has been used as 'Pareto composition' (cf. [10]). Please note that in complete total order preferences like used in conventional skyline queries (as in the example above) this problem does not arise, because any two objects o_1 and o_2 with different attribute values can be compared. But dealing with partial order preference semantics we do have to tackle the problem of incomparability for finding efficient evaluation algorithms.

Obviously the two cases stated above are the extreme cases, where either none or all incomparable objects are put together in an equivalence class. As one of the main differences it has also been shown that the first definition always induces a transitive Pareto preference [15], whereas depending on the base preferences for the second definition the induced preference's transitivity is generally violated [3], [10], since choosing eq_i as all incomparable value pairs does not define an equivalence relation. For effective pruning (and thus efficient query processing) we should rely on transitive Pareto relations, but of course we also want lean skyline sizes. Thus, the question is: What definition of equivalence puts the maximum of incomparable objects into equivalence classes, while still maintaining transitivity and respecting (of course) the user's intentions?

3.2. The Customized Pareto Aggregation

In the following we will formalize the possible definitions of equivalence relations eq_i and will show which definition is optimal in decreasing skyline sizes while always maintaining transitivity of the combined preference. There are already several instantiations of eq_i in literature:

a) **Pareto Accumulation [15]:** $(o_1,o_2) \in eq_i$, iff $att_i(o_1) =_{Hj} att_i(o_2)$, i.e. the attribute values have to be identical
b) **(Maximal) Substitute Value (SV) Semantics [16]:** $(o_1,o_2) \in eq_i$, iff $att_i(o_1)$ and $att_i(o_2)$ share the same sets of parents and descendants with respect to H_j (This includes the case $att_i(o_1) =_{Hj} att_i(o_2)$)
c) **Pareto Composition [10]:** $(o_1,o_2) \in eq_i$, iff $att_i(o_1)$ and $att_i(o_2)$ are incomparable with respect to H_i (This also includes the case $att_i(o_1) =_{Hi} att_i(o_2)$, since then neither $att_i(o_1) <_{Hi} att_i(o_2)$, nor $att_i(o_2) <_{Hi} att_i(o_1)$ holds. Note that in general eq_i is not an equivalence relation.)

Obviously the definitions of equivalence get weaker from a) to b) and from b) to c). Also in case b) the aggregated preference is always transitive [16]. But of course there are many more possible definitions of equivalence relations between a) and c) (as we will discuss in section 4). The exact interrelationship between P_i and eq_i will determine the characteristics of the Pareto aggregation. A rather strong connection is given by the following definition:

Definition 2: (compatibility of eq_i)
A binary relation eq_i is called compatible with a partial order preference P_i, if for all objects $o_1, o_2, o_3 \in O$ holds: $(o_1,o_2) \in eq_i \wedge (o_2,o_3) \in P_i \Rightarrow (o_1,o_3) \in P_i$ and $(o_1,o_2) \in P_i \wedge (o_2,o_3) \in eq_i \Rightarrow (o_1,o_3) \in P_i$.

That means a relation eq_i is compatible, if the domination relation between two objects always extends to all other members of their respective classes ([16] introduced this definition under the name 'SV relation'). As a first observation we can now show that the compatibility characteristic is necessary for certain transitivity (the sufficiency is shown in [16]):

Lemma 1: (compatibility and transitivity)
For any preference P and equivalence relation eq as defined in Definition 1 the partial order $(P \cup eq)$ is transitive, if and only if eq is compatible with P.

Proof: '=>' Let o_i be database objects (i = 1,2,3). Consider $(o_1,o_2) \in eq$ and $(o_2,o_3) \in P$, then we can conclude that $(o_1,o_2), (o_2,o_3) \in (P \cup eq)$ and due to the transitivity of $(P \cup eq)$, also $(o_1,o_3) \in (P \cup eq)$. For the sake of contradiction assume $(o_1,o_3) \in eq$, then by the symmetry and transitivity of equivalence relation eq we also get $(o_2,o_3) \in eq$ which contradicts $P \cap eq = \emptyset$ in Definition 1. Hence, it always follows $(o_1,o_3) \in P$, which is the first part of the compatibility definition. The other part follows analogously by assuming $(o_1,o_2) \in P$ and $(o_2,o_3) \in eq$.

'<=' For this direction we have to show that all compositions of elements in $(P \cup eq)$ are again in $(P \cup eq)$, i.e. $(P \cup eq) \circ (P \cup eq) \subseteq (P \cup eq)$. Thus we need to consider four cases:

a) $P \circ P \subseteq (P \cup eq)$, follows directly from the transitivity of P
b) $P \circ eq \subseteq (P \cup eq)$, follows from the compatibility of P and eq
c) $eq \circ P \subseteq (P \cup eq)$, also follows from the compatibility of P and eq
d) d) $eq \circ eq \subseteq (P \cup eq)$, follows directly from the transitivity of eq ■

This result on single preferences and equivalence relations leads easily to a related result for the m-dimensional customized Pareto aggregation:

Lemma 2: (compatibility and transitivity of customized Pareto aggregation)
Let preference P_1 and equivalence relation eq_1 be defined as in Definition 1. Consider preferences $P_2,..,P_m$ and equivalence relations $eq_2,.., eq_m$ with $P_i \cap eq_i = \emptyset$ and eq_i is compatible with P_i. If there is an integer m such that for all possible choices of such preferences and equivalence relations the customized Pareto aggregation $Pareto(O, P_1,\ldots,P_m, eq_1,\ldots,eq_m)$ is transitive, then also eq_1 has to be compatible with P_1.

Proof: The proof is by contradiction. Assume that the customized Pareto aggregation is always transitive for the given m, but eq_1 would not be compatible with P_1. Then there exist objects $o_1,o_2,o_3 \in O$ such that either: $(o_1,o_2) \in P_1 \wedge (o_2,o_3) \in eq_1$, but $(o_1,o_3) \notin P_1$, or $(o_2,o_1) \in P_1 \wedge (o_2,o_3) \in eq_1$, but $(o_3,o_1) \notin P_i$.

Without loss of generality let us assume the first case. Then also $(o_1,o_3) \notin eq_1$ holds, because otherwise due to $(o_2,o_3) \in eq_1$ and eq_1's transitivity also $(o_1,o_2) \in eq_1$ would hold in contradiction to $(o_1,o_2) \in P_1$. Now construct $P_2,..,P_m$ such that for $2 \le i \le m : (o_1,o_2) \in P_i$ and $(o_2,o_3) \in P_i$ (and thus also $(o_1,o_3) \in P_i$) and choose arbitrary equivalence relations $eq_2,\ldots,eq_m$ such that for $2 \le i \le m : eq_i \cap P_i = \emptyset$. Then we have

$(o_1,o_2) \in Pareto(O, P_1,\ldots,P_m, eq_1,\ldots,eq_m)$,
because $(o_1,o_2) \in P_1$ and for $2 \le i \le m : (o_1,o_2) \in P_i$

and $(o_2,o_3) \in Pareto(O, P_1,\ldots,P_m, eq_1,\ldots,eq_m)$,
because $(o_2,o_3) \in eq_1$ and for $2 \le i \le m : (o_2,o_3) \in P_i$,

but $(o_1,o_3) \notin Pareto(O, P_1,\ldots,P_m, eq_1,\ldots,eq_m)$,
because neither $(o_1,o_3) \in P_1$ nor $(o_1,o_3) \in eq_1$.

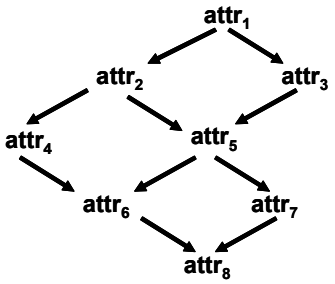

Figure 2. Preference with natural level order

Thus, the resulting customized Pareto aggregation would not be transitive. In summary, we have shown that compatibility of eq_i with P_i for all i is really necessary for transitivity of the Pareto aggregation. ■

It is easy to construct examples where the equivalence relations are not compatible, but the customized Pareto aggregation is nevertheless transitive. But compatible relations are the only possible semantics for the eq_i relation that *always* retains transitivity in the induced Pareto aggregation. Thus sticking to the maximal SV semantics as given in [16] seems indeed to be the best choice in customizing Pareto aggregations. However, this is only true, if we restrict ourselves to compatible relations. In the next section we will explore a more powerful set of equivalence relations.

4. Customizing Pareto Aggregation beyond the Compatible Case

To customize the Pareto aggregation by compatible relations is only of limited value, because it based on the requirement that the simple set-theoretic union of the preference and equivalence relation has to be transitive. Generally speaking, this does happen only for equivalence relations that are close to the equality relation. Hence, valuable potential to substantially decrease skyline sizes by dealing with incomparability is lost. Consider the following simple example.

Example: Let P be the preference given in Figure 2 with a total of eight attribute values and eq be an equivalence relation. If we insist on compatibility of eq, the equality relation is the only possible choice for eq and thus the skyline size cannot be decreased by stating equivalences at all. On the other hand it seems quite intuitive for the user to consider a replacement of attribute values by similarly preferred attribute values in the query as being of equal value with respect to the result (for instance the attribute value $attr_2$ could be considered equally desirable to $attr_3$, and similarly $attr_4$ to $attr_5$ and $attr_6$ to $attr_7$).

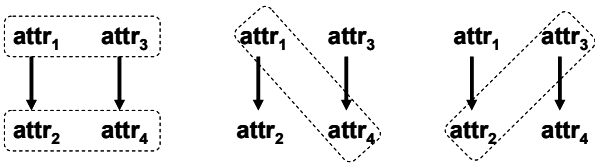

Figure 3. Possible consistent equivalences in two branches of a preference

To cater also for such equivalences a simple possibility is to use the level of objects as given by the longest path from any maximal object (in the preference of the example above objects can be assigned to 5 levels):

Definition 3: (level equivalence relation of a preference)
Let P be a partial order preference. An object $o \in O$ is said to belong to level i (or level(o) = i) with respect to preference P, if the longest path from any maximum object in P to o consists of (i - 1) edges. The level equivalence relation of preference P is then defined as: $lev_p ::= \{(o_1,o_2) \mid o_1, o_2 \in O \wedge level_P(o_1) = level_P(o_2)\}$.

This level equivalence relation has some nice properties. First, it defines an equivalence relation and naturally enforces $P \cap lev_p = \varnothing$. Second, all compatible equivalence relations are a subset of the level equivalence relation and only for total order base preferences the level equivalence relation is itself compatible. In a nutshell, the level equivalence relation can deal with incomparability more efficiently than compatible relations and thus is a promising and intuitive candidate for decreasing skyline sizes. However, we can easily see that in general $(P \cup lev_p)$ will not be transitive and thus also our Pareto aggregation will not be transitive. The solution to this problem is moving to the transitive closure, denoted by $(P \cup lev_p)^+$, for evaluation purposes. Here we get the best of both worlds: better possibilities of customizing Pareto aggregations to reduce skyline sizes and always transitive aggregations for efficient evaluation. Note that this step does not violate any user constraints, either. By expressing e.g. a preference on o_1 over o_2 and expressing the equal desirability of o_1 and o_3, a user implicitly also expresses a preference on o_3 over o_2. Only this implicit relationship is added by the transitive closure.

But this definition can not only be applied to level equivalence relations. A user can declare as many equivalences as are justified by the individual context of the query. An example can be seen in Figure 3: between independent branches of a preference (here shown with four attribute values, P is shown as arrow, eq is shown as dashed box) depending on the intended semantics a user can define several equivalences. One example is the level equivalence where $attr_1$ is compa-

rable to $attr_3$ and $attr_2$ to $attr_4$. But of course a user could also state that $attr_1$ should be comparable with att_4 or that $attr_2$ should be comparable to $attr_3$. Please note that making $attr_1$ comparable to att_4 and at the same time $attr_2$ to $attr_3$ is not a sensible option, because it would immediately create conflicts with the strict preference P. Since there, thus, are a lot such non-compatible equivalence relations besides the level order that can be readily used for customization, we will adapt our Pareto aggregation definition to always consider transitive closures:

Definition 4: (Customized Pareto Aggregation using transitive closures)

Let O be a set of database objects and preferences $P_1,\ldots,P_m$ and equivalence relations $eq_1,\ldots,eq_m$ be defined as in Definition 1 including $\forall\ 1 \le i \le m$: $P_i \cap eq_i = \emptyset$. Then we can define Pareto aggregation using transitive closures and customized by equivalence relations as:

$Pareto(O, P_1,\ldots,P_m, eq_1,\ldots,eq_m) := \{(o_1, o_2) \mid \forall\ 1 \le i \le m: (o_1, o_2) \in (P_i \cup eq_i)^+ \wedge \exists\ 1 \le j \le m: (o_1, o_2) \notin eq_j\}$

This definition includes also Definition 1, because for compatible relations as used in Definition 1 always $(P_i \cup eq_i)^+ = (P_i \cup eq_i)$. Moreover, the customized Pareto aggregation always defines a transitive order, if all eq_i are consistent with the P_i, i.e. no element in $(P_i \cup eq_i)^+$ contradicts an element in any P_i

We have thus enabled the user to perform all suitable customizations by adding more and more equivalence relations and thus reduce skyline sizes by additional semantic information. But how far can a user go when adding equivalences for customization? The next lemma will show that a maximum possible customization consistent with the user's preferences is indeed the level equivalence relation. If a user adds only one more equivalence on top of the level equivalences, the Pareto aggregation would become inconsistent.

Lemma 3:(maximum equivalence customization)

Let P_1 be a preference and let q_{P1} be a symmetric binary relation with $P_1 \cap q_{P1} = \emptyset$. Further let q_{P1} be larger than the respective level equivalence lev_{P1}, i.e. $lev_{P1} \subset q_{P1}$. Then we can always construct a preference P_2 and equivalence relation eq_2 such that the customized Pareto aggregation $Pareto(O, P_1,P_2, q_{P1},eq_2)$ contains a tuple (o_1,o_2), while P_1 already contains an inverse tuple (o_2,o_1).

Proof: Since $lev_{P1} \subset q_{P1}$ there have to be objects o_1, $o_2 \in O$ such that $(o_1,o_2) \in q_{P1}$, but $(o_1,o_2) \notin lev_{P1}$, i.e. $level(o_1) \neq level(o_2)$. Without loss of generality we can assume $level(o_1) < level(o_2)$. Then there exists $o_3 \in O$ with $level(o_3) = level(o_1)$ and $(o_2,o_3) \in P_1$ as follows from Definition 3. Now choose P_2 such that $(o_3,o_2) \in P_2$ and for eq_2 choose the equality relation. Then, also holds $(o_3,o_2) \in Pareto(O, P_1,P_2, q_P,eq_2)$, which obviously conflicts with $(o_2,o_3) \in P_1$. Thus the customized Pareto aggregation $Pareto(O, P_1,P_2, q_P,eq_2)$ is inconsistent with an explicit user preference. ■

5. Evaluation of Maximum SV Semantics

The goal of our evaluation was to find out how the equivalence relation choice influences skyline sizes. Pareto accumulation obviously is the minimal equivalence relation, and, as we have shown, level equivalence is a maximal transitive equivalence. Therefore, the user can expect skyline reductions by step-wise adding equivalence information up to the reduction brought by level equivalence. We measured skyline sizes for these two relations. To compare with existing work, we also added the substitute value (SV) equivalence relation. For these relations we conducted experiments with different settings and compared respective skyline sizes.

In order to avoid bias all experiments were performed on synthetically created data and averages over multiple runs are reported in our graphs. Both data and preferences are generated randomly for each measurement. The data is uniformly distributed (except for the skylines sizes on Zipf-distributed data) and preferences are generated based on certain realistic parameters. Table 2 shows our default configuration. In all scenarios, parameters not explicitly changed are set to these default values.

We will now briefly explain the parameters used and in the next sections, we describe each scenario and its outcome in detail:

- *Preference size*: The number of distinct attribute values modeled by a preference
- *Preference depth*: The longest path within a preference graph
- *Edge ratio*. The ratio between nodes and edges in a preference graph

Table 2. Default evaluation settings

Parameter	Value
Database size	25000
Number of preferences	5
Preference size	15
Preference depth	5
Edge Ratio	1.2

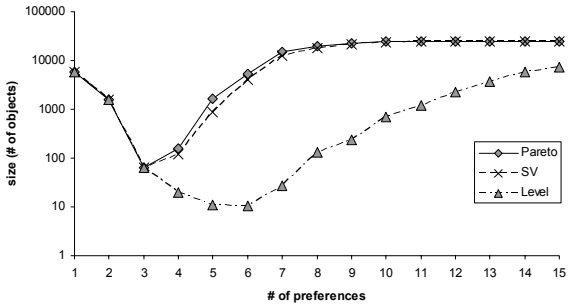

Figure 4. Effect of query dimensionality for uniformly distributed data

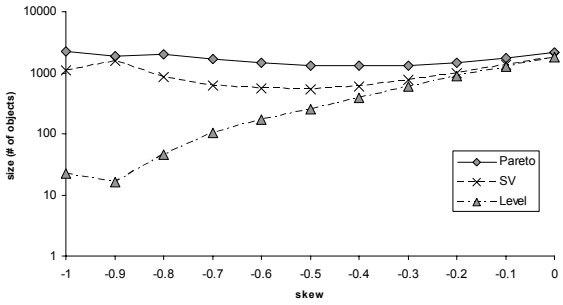

Figure 5. Effect of query dimensionality for Zipf distributed data

5.1. Influence of Query Dimensionality

The goal in this scenario was to quantify how the number of preferences (or independent query predicates) affects different skyline sizes. That means to what number of dimensions skylines can be expected to be usually still manageable. SV and Pareto accumulation skyline are both touched by the curse of dimensionality, like also comparable work shows: In the case of uniformly distributed data the Pareto accumulation skyline sizes increase substantially when more than 5 query predicates are considered, as shown in Figure 4 (this and the following figures use a logarithmic scale for skyline sizes). After a steep incline for 5 to 8 predicates almost the entire database is retrieved as skyline. SV skylines generally are only slightly smaller and suffer the same degradation with increasing numbers of dimensions. The SV semantics thus does not add enough equivalence relations to facilitate the computation of high dimensional skylines. In contrast, level equivalence skylines are significantly smaller for dimensions from 4 upward, and stay sufficiently small to be manageable for a human user for around 10 dimensions.

We also considered the case of Zipfian data distributions that are highly common for real world content distributions, for instance in the WWW [1]. In this case, a few attribute values for items are often assumed by objects, whereas all other attribute values are rather rare. We varied skews from uniform (skew parameter -1.0) to highly skewed (0.0). In the latter case, the most preferred attribute values in each preference is already assumed by 14% of all database objects. The results are shown in Figure 5. With growing skew the different skyline types coincide more and more. With so many objects having top attribute values, the chance for incomparability gets lower, and a set of rather similar top objects will dominate the whole rest of the database. For usual skews (up to -0.5) the level order skyline is still one or two orders of magnitude smaller than SV and Pareto accumulation skylines.

5.2. Influence of Database Size

To determine the influence of the database size in terms of scalability, we fixed the number of predicates to 5 dimensions and varied the number of database objects between 50,000 and 250,000. Figure 6 reports our results.

The resulting skylines sizes in all cases are rather constant, growing only slowly with increasing database size. This effect can be explained by the fact that most objects that are generated to fill the database will be dominated by already existing top objects, whereas the probability to create such a top object is relatively small. In fact, skyline sizes have been shown to grow only logarithmically with the number of database items. Again we can see that the SV skyline is only slightly smaller than the Pareto accumulation skyline. The level equivalence skyline again is constantly two orders of magnitude smaller than both the Pareto accumulation and SV skylines.

In summary, the scalability of the skyline paradigm thus is rather less dependent on the total database size, but rather more on the query dimensionality as we have seen in section 5.1. Depth and edge ratio of the underlying preference graphs in our evaluation scenario definitely influence the factor of skyline growth for increasing database sizes, but an analytical investigation of this factor is beyond the scope of this paper.

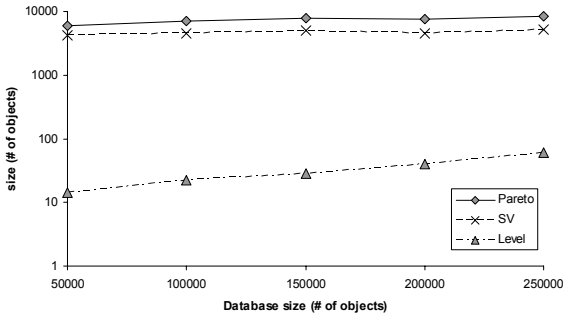

Figure 6. Effect of database size on skyline sizes

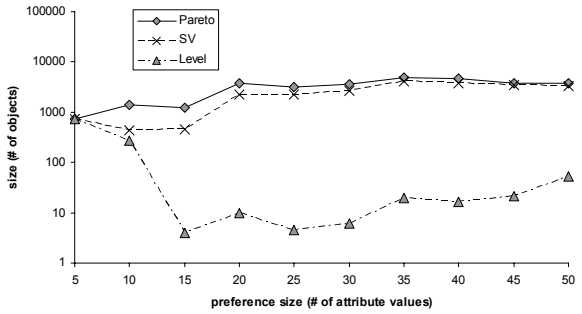

Figure 7. Preference size effect on skyline size

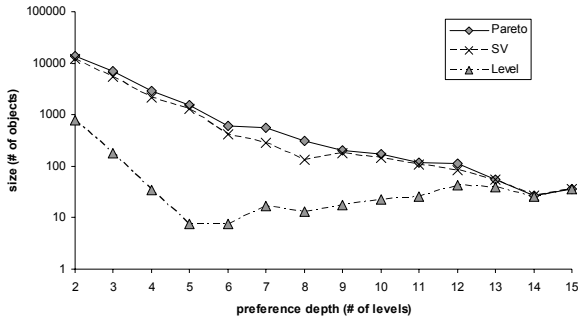

Figure 8. Preference depth effect on skyline size

5.3. Influence of Preference Size and Shape

In a last set of experiments, we varied the preference size and the depth of the preference graph. In the first scenario we increased the number of distinct domain values between 5 and 30. In the second scenario the preference graph depth was gradually increased from 2 to 15 while keeping the number of domain attributes fixed, i.e. the partial order preference gradually is transformed into a total order preference.

In both scenarios Pareto accumulation and SV skylines behave in a similar fashion. We can see the results in Figure 7 and 8. When increasing preferences size from 5 to 20, SV and Pareto accumulation skyline sizes grow up to 25%, respectively 30% of the database size. From 20 upward, these skylines don't grow further. Also here SV skylines stay always slightly smaller than Pareto accumulation skylines. Level equivalence skylines start with equal size for a preference size of 5, because this is already a total order due to the given depth of 5. However, in general Level equivalence again reduces skyline size by one or two orders of magnitude, only slowly growing with increased preference size.

When increasing preference depth, Pareto accumulation and SV skyline shrink notably. As could be expected, this happens due to the reduction of incomparable attribute values within each individual preference. With less incomparability also the difference between SV and Pareto accumulation skyline size decreases. Level equivalence shows a complementary behavior: with increasing preference depth, the preference converges to a level order anyway, and thus level equivalence has less and less effect. At a depth of 15 the preference finally becomes a total order and thus all three skylines coincide.

6. Summary

Today efficient processing of preference queries is essential for human-centered information systems. While computing skylines over numerical domains has made a lot of progress recently, their sizes over discrete value domains are often too large, rendering the outcome useless for the human user. This is caused by the fact that only partial orders are available to evaluate preferences, and thus attribute values often become incomparable and unnecessarily inflate skyline sizes.

To limit unnecessary incomparability and thus reduce skyline sizes, we have introduced the notion of customized preference aggregation. In addition to individual base preferences, users can explicitly express an equal desirability with respect to certain attribute values by means of equivalence relations. This notion offers a general framework into which existing proposals such as numerical skylines or SV semantics fit as special cases. Our approach offers more flexibility because the user is empowered to decide which values to treat as equally good and to add perceived equivalences one by one until the skyline is reduced to an acceptable size. Thus, skyline queries are extended by an effective trade-off management and for the entire querying process the user stays in control over the performed reductions.

Besides the effectiveness of our approach, transitivity of such an aggregated preference relation is important, because this also ensures efficient skyline query processing. While user-specified equivalence relations do not in general ensure transitivity (in fact, we have seen the SV relation to be the maximal equivalence relation always retaining this characteristic), we consider the transitive closure for a given consistent equivalence relation. In that way, the additional equivalence relations do not affect query processing efficiency adversely, and indeed query runtimes for all approaches were similar throughout our experiments.

For validating our approach and getting an intuition about the skyline reductions that can be expected, we have shown that level equivalence forms a maximum relation under these assumptions. Therefore, this relation also causes a maximum reduction of skyline sizes. Evaluating this reduction we found it to be quite impressive, thus in our framework the user can decrease skyline sizes by up to two orders of magnitude in a

controlled fashion. Our experiments show that controlled exploitation of indifference is a useful tool to limit skyline size, in contrast to compatible equivalence relations (as given by SV semantics) which cannot reduce skyline sizes by much, allowing also all consistent equivalence relation (as given up to the level equivalence) enables user to cut down skylines to manageable sizes.

Our approach can be put to use in interactive search processes, where users refine their queries step by step adding more and more personal relevance information. In our case, users will start out with ordinary Pareto accumulation, and add predicate value equivalences until the skyline has reached a reasonable size. Skyline queries can thus make the essential step to manageable results by personalization. Given that skyline queries' current popularity in many real world applications was only hampered by their exponentially growing result sets, the promising results gained by customized Pareto aggregations will definitely show an important impact on personalized query processing.

Acknowledgements

Part of this work was funded within the Emmy-Noether program of excellence of the German Research Foundation (DFG) as part of the APIS (Advanced Personalization in Information Services) project.

10. References

[1] L. Adamic, B. Huberman. Zipf's Law and the Internet. In *Glottometrics*, Vol. 3, 2002.

[2] W.-T. Balke, U. Güntzer. Multi-objective Query Processing for Database Systems. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, Toronto, Canada, 2004.

[3] W.-T. Balke, U. Güntzer. Efficient Skyline Queries under Weak Pareto Dominance. In *Proc. of the IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling (PREFERENCE)*, Edinburgh, UK, 2005.

[4] W.-T. Balke, U. Güntzer, J. Zheng. Efficient Distributed Skylining for Web Information Systems. In *Proc. of the Int. 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. In *Proc. of the Int. Conf. on Database Systems for Advanced Applications (DASFAA),* Beijing, China, 2005.

[6] J. Bentley, H. Kung, M. Schkolnick, C. Thompson. On the Average Number of Maxima in a Set of Vectors and Applications. In *Journal of the ACM (JACM)*, vol. 25(4) ACM, 1978.

[7] S. Börzsönyi, D. Kossmann, K. Stocker. The Skyline Operator. In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, Heidelberg, Germany, 2001.

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

[9] J. Chomicki. Querying with Intrinsic Preferences. In *Proc. of the Int. Conf. on Extending Database Technology (EDBT)*, LNCS 2287, Prague, Czech Republic, 2002.

[10] J. Chomicki. Preference Formulas in Relational Queries. In *ACM Transactions on Database Systems (TODS)*, Vol. 28(4), 2003.

[11] R. Fagin, A. Lotem, M. Naor. Optimal Aggregation Algorithms for Middleware. In *ACM Symp. on Principles of Database Systems (PODS)*, Santa Barbara, USA, 2001.

[12] P. Fishburn. Preference Structures and their Numerical Representations. *Theoretical Computer Science*, Vol. 217, 1999.

[13] U. Güntzer, W.-T. Balke, W. Kießling. Optimizing Multi-Feature Queries for Image Data-bases. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, Cairo, Egypt, 2000.

[14] X. Huang, C. Jensen. In-Route Skyline Querying for Location-Based Services. In *Proc. of the Int. Workshop on Web and Wireless Geographical Information Systems (W2GIS)*, Goyang, Korea, 2004.

[15] W. Kießling. Foundations of Preferences in Database Systems. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, Hong Kong, China, 2002.

[16] W. Kießling. Preference Queries with SV-Semantics. In *Proc. of the Int. Conf. on Management of Data (COMAD)*, Goa, India, 2005.

[17] V. Koltun, C. Papadimitriou. Approximately Dominating Representatives. In *Proc. of the Int. Conf. on Database Theory (ICDT)*, Edinburgh, UK, 2005.

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

[19] M. Lacroix, P. Lavency. Preferences: Putting more Knowledge into Queries. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, Brighton, UK, 1987.

[20] A. Motro. VAGUE: A User Interface to Relational Databases that Permits Vague Queries. In *ACM Transactions on Office Information Systems (TOIS)*, vol. 6(3), 1988.

[21] D. Papadias, Y. Tao, G. Fu, B. Seeger. An Optimal and Progressive Algorithm for Skyline Queries. In *Proc. of the Int. ACM SIGMOD Conf. (SIGMOD'03)*, San Diego, USA, 2003.

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