From Stars to Galaxies: skyline queries on aggregate data

Matteo Magnani
Aarhus University
Aabogade 34
8200 Aarhus N., DK
magnanim@cs.au.dk

Ira Assent
Aarhus University
Aabogade 34
8200 Aarhus N., DK
ira@cs.au.dk

ABSTRACT

The `skyline` operator extracts relevant records from multidimensional databases according to multiple criteria. This operator has received a lot of attention because of its ability to identify the best records in a database without requiring to specify complex parameters like the relative importance of each criterion. However, it has only been defined with respect to single records, while one fundamental functionality of database query languages is *aggregation*, enabling operations over sets of records. In this paper we introduce *aggregate skylines*, where the skyline works as a filtering predicate on sets of records. This operator can be used to express queries in the form: *return the best groups depending on the features of their elements*, and thus provides a powerful combination of grouping and skyline functionality. We define a semantics for aggregate skylines based on a sound theoretical framework and study its computational complexity. We propose efficient algorithms to implement this operator and test them on real and synthetic data, showing that they outperform a direct SQL implementation of up to two orders of magnitude.

Categories and Subject Descriptors

H.2 [**Database management**]: Miscellaneous

1. INTRODUCTION

Aggregate skyline queries merge the functionality of two basic database operators, `skyline` and `group by`. In the following we introduce this new kind of query starting from these two basic operators. Then, we discuss the difference between aggregate skyline queries and a sequential execution of `skyline` and `group by`, that has been object of previous studies, and outline the main contributions and the structure of this paper.

1.1 Skyline queries

The *skyline* is a database query operator used to identify records of interest *according to multiple criteria* [5]. This task is in general complex because the result of a query based on multiple criteria may change depending on how much each criterion is weighed, i.e., depending on the *record scoring function*. For example, consider the `Movie` relation in Figure 1, containing a set of movies from the Internet Movie Database[1], with year, director, popularity (expressed in thousands of votes) and quality (expressed as the average user evaluation on a $[0, 10]$ range). Assume we want to extract the best movies, and we have a preference for movies that are popular and received good evaluations. If we are more interested in popularity our first option will be `Pulp Fiction`, while if we are more interested in quality our first choice will be `The Godfather`. This example highlights how it may not be possible to identify a single best answer in presence of multiple criteria: `Pulp Fiction` and `The Godfather` are said to be *incomparable* because the former is better with respect to popularity and the latter is better with regard to quality.

Title	Year	Director	Pop	Qual
Avatar	2009	Cameron	404	8.0
Batman Begins	2005	Nolan	371	8.3
Kill Bill	2003	Tarantino	313	8.2
Pulp Fiction	1994	Tarantino	557	9.0
Star Wars (V)	1980	Kershner	362	8.8
Terminator (II)	1991	Cameron	326	8.6
The Godfather	1972	Coppola	531	9.2
The Lord of the Rings	2001	Jackson	518	8.7
The Room	2003	Wiseau	10	3.2
Dracula	1992	Coppola	76	7.3

Figure 1: `Movie` table

The `skyline` operator [5] addresses this problem by guaranteeing that the best option will always be included in the result whatever the record scoring function is :

EXAMPLE 1 (SKYLINE QUERY). *The best movies according to their popularity and quality can be extracted using the following query (the result is indicated in Figure 2):*

SELECT *
FROM Movie
SKYLINE OF Pop MAX, Qual MAX

The semantics of this operator is based on the concept of *dominance* [5]: it selects all records that are *not dominated* by any other record, where *dominance* is defined as follows ($\leq$ and $<$ can also be used if for some attributes lower values are preferred, but for

[1]http://www.imdb.com

Title	Year	Director	Pop	Qual
Pulp Fiction	1994	Tarantino	557	9.0
The Godfather	1972	Coppola	531	9.2

Figure 2: Traditional record-wise skyline on the `Movie` table

Director	MAX(Pop)	MAX(Qual)
Cameron	404	8.6
Nolan	371	8.3
Tarantino	557	9.0
Kershner	362	8.8
Coppola	531	9.2
Jackson	518	8.7

Figure 3: Traditional aggregate query performed on the `Movie` table

simplicity we will assume to prefer higher values throughout the paper without loss of generality):

DEFINITION 1 (DOMINANCE). *Let r and s be two records in d dimensions. r dominates s iff* $\forall i \in [1, d]\ r_i \geq s_i \wedge \exists i\ r_i > s_i$

Notice how `The Room` is not selected by the previous skyline query because its values of popularity and user evaluation ($\langle 10, 3.2\rangle$) are strictly lower than, e.g., the ones achieved by `The Godfather` ($\langle 531, 9.2\rangle$), or said in another way, `The Godfather` *dominates* `The Room`.

1.2 Aggregate queries

In the previous example we have computed a skyline of *single records*. However, in general queries do not just target single records, e.g., movies can be grouped by director, year or country. To answer queries regarding directors instead of single movies, or more generally *groups of records* instead of *single records*, we can use aggregate queries:

EXAMPLE 2 (AGGREGATE QUERY). *The maximum popularity and quality achieved by directors with a maximum quality of at least* 8.0 *can be extracted using the following* aggregate *query (the result is indicated in Figure 3):*

SELECT Director, max(Pop), max(Qual)
FROM Movie
GROUP BY Director
HAVING max(Qual) $>=$ 8.0

Director
Coppola
Tarantino

(a)

Director
Coppola
Jackson
Kershner
Tarantino

(b)

Figure 4: Skyline (a) and Aggregate Skyline (b) directors

1.3 Aggregate skyline

In the two previous sections we have seen a skyline query extracting the *most interesting records* from a table, and an aggregate query summarizing the features of *groups of records*. In this paper we introduce *aggregate skyline queries* combining the *group by* and *skyline* functionality and enabling the computation of skylines over groups of records. Aggregate skyline queries allow us to formulate queries such as: *What are the most interesting directors (according to the features of their movies)?* Or, more in general: *What are the most interesting groups (according to the features of their elements)?*

EXAMPLE 3. *We want to know who are the best directors in Table 1. To do this we compare their movies according to popularity and user evaluation. The corresponding aggregate skyline query is:*

SELECT director
FROM movies
GROUP BY Director
SKYLINE OF Pop MAX, Qual MAX

In summary, while a traditional skyline can be seen as a powerful `where` SQL clause that filters out irrelevant *single* records, an aggregate skyline corresponds to a `having` clause and is used to filter out irrelevant *groups of records*.

Interestingly, aggregate skyline queries have not been studied so far. In the literature some works have already focused on the optimization of SQL queries involving both skyline and grouping, but in these works the two operators are executed one after the other in isolation [10, 2, 1]. Differently, aggregate skylines are not just the result of the execution of a `group by` followed by a `skyline`, or vice-versa. We discuss these options in the following.

[skyline $\rightarrow$ group by] Computing a skyline before grouping corresponds to a different kind of query, because the skyline is applied to single records without knowledge of the group structure. For example, consider directors `Tarantino` and `Jackson`. `Tarantino` is present in the `Movie` relation with two movies, one better and one worse than the only movie by `Jackson` (according to our two criteria). Therefore, from the overall comparison we cannot conclude that `Tarantino`'s movies are necessarily better than `Jackson`'s movies or vice-versa. However, if we compute a skyline on the `Movie` relation, `Pulp Fiction` filters out most of the other movies (and thus directors, including `Jackson`) without considering any aggregate information about them. From this discussion it appears how a skyline followed by other operators, e.g., `group by`, returns information on *the directors of the most interesting movies* and not *the most interesting directors.*

[group by $\rightarrow$ skyline] Similarly, a skyline after a `group by` has only access to the summary information computed by the aggregate functions. As a consequence, performing a skyline on a partial relation obtained after grouping (Figure 3) may produce unwanted results. For example, in Figure 3 director `Cameron` appears to be strictly better than `Nolan`, while looking at the original data in Figure 1 we see that no movie by `Cameron` is strictly better than `Nolan`'s only movie.

Using other aggregate functions does not improve the situation. For example, we may try to compare the average values instead of the maxima. In this case, consider the average performance of

Tarantino's movies, having popularity and quality $\langle 435, 8.6 \rangle$. If we now check the movies by `Jackson` and `Kershner` (respectively, `The Lord of the Rings` and `Star Wars (V)`) we can see how both are better than one movie by `Tarantino` and worse than the other, and they are mutually incomparable. This means that we cannot make any distinction between `Jackson` and `Kershner` according to the *dominance* relation. However, once we compare them against the average values for `Tarantino` we notice how the former (`Jackson`) is better while the latter (`Kershner`) is incomparable. On a more theoretical level, using aggregate functions like `avg` makes the resulting operator *not stable*, which is one of the main features of the original skyline operator. We will discuss this in more detail in the technical part of the paper.

Figure 4 exemplifies the previous discussion: both sequential applications of `group by` and `skyline` lead to the selection of only two skyline records (`Tarantino` and `Coppola`), while an aggregate skyline also retrieves, e.g., `Jackson`, whose general performance as a director is not dominated by any other directors (although his movies, taken one by one, are). Please notice that in the general case an aggregate skyline is not necessarily a superset of a traditional skyline, as we will prove through a theoretical study of the operator.

As a final remark, notice that an aggregate skyline can be computed whenever groups of records are involved, independently of how they have been generated. However, in this paper we will consistently focus on SQL queries involving the `group by` operator. In the remaining of the paper we use the movie database as a working example. In general, the application fields of aggregate skylines are numerous, including all contexts where relevant groups of records have to be identified. Relevant examples are the identification of virtuous hospitals/wards or problematic diseases in medical databases, or the identification of successful/popular kinds of products in on-line selling sites or warehouses.

1.4 Contributions and outline

The main contributions of this paper are:

- the introduction of the aggregate skyline operator,
- the definition of its semantics, based on a sound theoretical framework and on properties characterizing the result of aggregate skylines,
- efficient algorithms, with an extensive experimental evaluation showing significant improvements of the presented algorithms with respect to a direct SQL implementation.

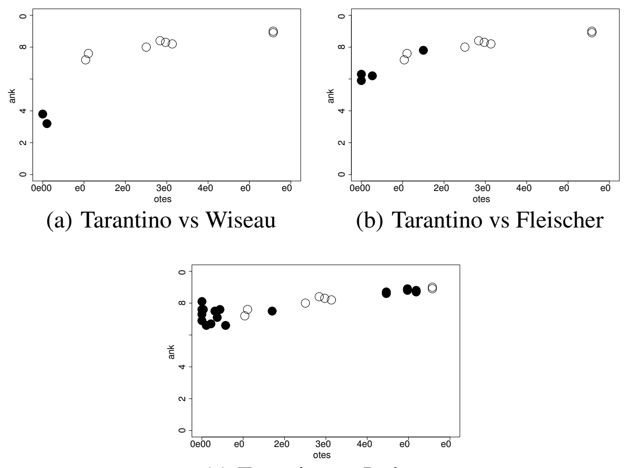
The paper is organized as follows. In Section 2 we define the aggregate skyline operator. This includes a set of properties guaranteeing that the operator returns the expected result and a theorem and discussion emphasizing the difficulty of this problem with respect to traditional skyline queries. Then, we study the computational complexity of the operator, showing that it is super-linear with respect to both the number and size of groups. Therefore, in Section 3 we define efficient algorithms. Section 4 presents the result of a thorough experimental evaluation performed both on real and synthetic data, showing that our approach outperforms a direct SQL implementation. We conclude the paper with a summary of our main results and a review of related work.

2. THE AGGREGATE SKYLINE

The original definition of *skyline* as the set of *records not dominated by other records* naturally extends to more complex entities, like groups of records. We can in fact define an aggregate skyline as the set of *groups not dominated by other groups*. However, while the definition of record dominance is straightforward, extending it to groups presents some challenges. We highlight them through some examples using real data taken again from the IMDB archive, starting from a clear case of group dominance.

The first step we take toward the definition of aggregate skylines is the introduction of two kinds of dominance, that we call *strict* and *weak* — the first is a special kind of the second. The existence of different kinds of dominance in real aggregate data, differently from the single-record case where dominance is a unique and clearly defined concept, is highlighted by the following two examples.

EXAMPLE 4 (STRICT DOMINANCE). *In Figure 5(a) we compare two directors, namely Quentin Tarantino (white points) and Tommy Wiseau (black points) according to their movies, where higher popularity and quality are preferred. The objective is to answer the question:* given the available information, would you prefer watching a movie by Tarantino or by Wiseau? *In this example we can safely claim that* Tarantino *is the answer, and this because he* strictly dominates *Wiseau. In fact, even the worst movie by Tarantino is better than any movie by Wiseau.*


(a) Tarantino vs Wiseau (b) Tarantino vs Fleischer

(c) Tarantino vs Jackson

Figure 5: Examples of group dominance relationships between movie directors (Tarantino is always represented by white points)

Strict dominance can be thought as the aggregate counter-part of record dominance. However, the next example shows how we may in general find it difficult to compare two groups even in cases where all movies are pair-wise comparable according to the traditional definition of dominance.

EXAMPLE 5 (WEAK DOMINANCE). *Figure 5(b) shows a comparison of Quentin Tarantino (white points) and Ruben Fleischer (black points). We can no longer claim that we would necessarily* *prefer watching a movie by* Tarantino*, because there is one particular movie by Fleischer (namely,* Zombieland*) that we would prefer to two movies by Tarantino. At the same time, we cannot say that the two directors are incomparable: it is still clear that Tarantino is in general better than Fleischer.*

Table 1: Meaning of the symbols

Symbols	Meaning
U_r	Record Universe (set of all records)
U_g	Group Universe (partition of U_r)
$r, s, t \dots$	Records ($\in U_r$)
$R, S, T \dots$	Groups, i.e., sets of records) ($\in U_g$)
Sky(U_r)	Traditional record skyline ($\in U_r$)
Sky$_g(U_g)$	Aggregate skyline ($\in U_g$)
γ	Degree of dominance between groups
$\succ_g$	group-dominance relation

In Figure 5(c) we compare Tarantino (white) and the director of The Lord of the Rings *trilogy Peter Jackson (black). Now it is harder to decide if the former is better than the latter, because although Tarantino's movies are in general more popular there are three notable exceptions, leading us to the conclusion that Tarantino dominates Jackson* to a lesser extent *than he dominates Fleischer.*

These examples highlight the necessity for a more general, flexible and practical notion of group dominance. In fact, when dealing with real data we can hardly rely on a strict notion of dominance between groups, but we must manage cases where a group dominates another *up to a certain extent*. This has not been previously formalized in the literature, where the concept of dominance has been used either in a strict Boolean sense (one record dominates or not another) or in a graded but record-oriented way, e.g., in fuzzy skylines where a *single record* may dominate another (or not) depending on a fuzzy comparison function [8, 21].

Therefore, in this paper we introduce and study a new notion of skyline based on group domination (in the following we will adopt the notational conventions indicated in Table 1). This notion of group domination, notated $\succ_g$, will be defined and thoroughly studied in the next section.

DEFINITION 2 (AGGREGATE SKYLINE). *Let U_g be a set of groups of records. An* aggregate skyline *is the set of all groups in U_g not dominated by any other group:*

$$Sky_g(U_g) = \{R \in U_g \mid \nexists S \in U_g \text{ such that } S \succ_g R\}$$

Definition 2 provides a clean extension of the skyline operator to groups, but does not formalize the concept of group-domination. Now the objective is to turn this concept into a well defined operator with a formal semantics clearly describing its properties. The first challenge we have to face is to provide a definition corresponding to the intuition behind group dominance that has emerged from the two previous examples. This will allow us to formally study the properties of our proposal and unveil other hidden challenges regarding its computation, that we will show to be independent of our specific definition and related to the general problem of group dominance comparisons. But first, we start by introducing our definition of group-dominance, that we call γ-dominance.

2.1 γ-dominance

The intuition behind our definition is the following: if we want to compare Tarantino with Wiseau we just assume to pick a random movie for both directors and check how likely it is for one to dominate the other. As a consequence, our proposal consists in relating γ to the probability that, given a random pair of records from the two groups that we are comparing (in our case, a random choice of a movie by director A and a movie by director B), the first record dominates the second[2]:

DEFINITION 3 (γ-DOMINATION). *Let $p(S \succ R)$ be the probability that, given a random pair of records $(r, s) \in R \times S$, $s \succ r$. We say that $S \succ_\gamma R$ iff $p(S \succ R) = 1 \vee p(S \succ R) > \gamma$.*

The fact that this definition is consistent with common-sense can be checked looking at Table 2, where we indicate the values of γ with respect to the three directors in Figure 5. For example, if we consider Tarantino and Fleischer, we can see that three movies by Fleischer are dominated by all movies directed by Tarantino, and one movie by Fleischer is dominated by six Tarantino movies. Therefore, Tarantino dominates Fleischer on $3 \cdot 8 + 1 \cdot 6 = 30$ out of 32 possible combinations of movies. The ratio between these two numbers gives the probability of domination .94 indicated in the table. It is worth noting that the probabilities that Tarantino dominates Jackson and vice-versa do not add to one, showing that for some pairs of movies none is strictly better than the other.

In the following we discuss which values of γ should be chosen when computing an aggregate skyline, then we prove that aggregate skylines satisfy two general notions of stability, making them robust with respect to limited variations in the data. In addition, we will discuss properties trivially satisfied in a single-record context but that no longer hold when we deal with groups of records. We will thus point out the resulting challenges and requirements for new algorithmic approaches.

Table 2: $p(S \succ R)$ for the examples in Figure 5

S	R	$p(S \succ R)$
Tarantino	Wiseau	1.00
Tarantino	Fleischer	.94
Tarantino	Jackson	.68
Wiseau	Tarantino	.00
Fleischer	Tarantino	.06
Jackson	Tarantino	.26

2.2 Setting γ

Let us consider again Figure 5. In Table 2 we indicate the values of γ for different pairs of groups, e.g., `Tarantino` γ-dominates `Fleischer` for all $\gamma < .94$. Now, we have to decide which groups to include in the skyline according to these values, i.e., we need to define a threshold to determine when the degree of domination is large enough to exclude the dominated group. Interestingly, not all values make sense. For example, if we set γ so that `Fleischer` dominates `Tarantino`, e.g., $\gamma < .06$ we would create the inconsistent situation where `Tarantino` would be strictly preferred to `Fleischer` and `Fleischer` would be

[2]The strict $>$ in the definition is necessary to guarantee asymmetry, a property studied later in this section

strictly preferred to `Tarantino`. To prevent this kind of inconsistencies, we require a definition of group-dominance to be *asymmetric*:

PROPERTY 1 (ASYMMETRY). *Let $R, S \in U_g$ be groups of records. If $R \succ_g S$ then $S \not\succ_g R$.*

Asymmetry is obviously satisfied when single records are concerned, because if $r \succ t$ there is an attribute i such that $r.i > t.i$, and consequently t cannot be strictly preferred to r. This is why this property was not emphasized in studies on the traditional skyline. However, it turns out to be relevant when we shift from a record to a group perspective, and constrains the range of acceptable values for γ (the proof of the following proposition follows directly from Definition 3):

PROPOSITION 1. *Definition 3 is asymmetric if $\gamma \geq .5$.*

This proposition has two main consequences. First, from now on we limit the definition of γ-dominance to the cases where $\gamma \geq .5$. From this point of view, γ can be seen as a tool to control the size of the result: we can first execute a query with $\gamma = .5$, which returns the smallest (most selective) set of results, and if we want to see more relevant groups we can increase the value of this parameter. Or, we can compute all groups that *can be* in an aggregate skyline, corresponding to $\gamma = 1$, and return them in sorted order according to the minimum value of γ for which they are in the group skyline. Notice that with $\gamma = .5$ a group dominates another whenever the probability of domination is higher than .5. With respect to higher values of γ, this may result in more dominances, and thus more groups excluded from the aggregate skyline and a smaller result.

At the same time, it is worth noting that $\gamma = .5$ represents a characteristic value with a clear and natural semantics: when $\gamma = .5$, group A dominates group B if, taking a random element a from A and a random element b from B, it is more probable that a dominates b than the contrary. Therefore, Proposition 1 identifies a value that can be used as a default if we want a parameter-free aggregate skyline operator, like in the SQL syntax presented in Example 3.

2.3 Stability of aggregate skylines

In the introduction we have discussed some problems related to the alternative approach of computing aggregate functions for every group and executing a skyline afterwards. These problems are related to a general notion of *stability* indicating that the result of a method is not influenced by several kinds of data transformation, as detailed in the following.

A first kind of stability regards the insertion and deletion of records. This is important to control the behavior of the operator. For example, assume that a very good director who has directed a large number of popular and high quality movies comes out with a very bad movie. The insertion of this single new record may certainly influence our overall judgment about the director, but should not make us forget about all the good movies that she directed before. We may thus have a change in γ, but this change should be bounded, that is, the degree cannot change arbitrarily when the change in the group is small. *Stability to updates* states that small changes in a group have a small effect on γ.

PROPERTY 2 (STABILITY TO UPDATES). *Let $R \succ_\gamma S$ (resp., $S \succ_\gamma R$), $R' \subseteq R$ and $\epsilon = \frac{(|R|-|R'|)}{|R|}$. For every $|R|, |R'|$ and $|S| \neq 0$, the group domination predicate is* robust *if $R' \succ_{\gamma'} S$ (resp., $S \succ_{\gamma'} R'$) with $\gamma(1-\epsilon) \leq \gamma' \leq \gamma(1+\epsilon)$.*

PROOF. We need to consider the cases where we remove or add records to the first group ($R \succ_\gamma S, R' \subseteq R$) or to the second ($S \succ_\gamma R, R' \subseteq R$). Here we prove the first case, while the proof of the second follows similarly and is not reported.

Let us notate $|R \succ S|$ as the number of pairs $(r, s) \in R \times S$ such that $r \succ s$. We also indicate the cardinality of group R as $|R|$. Therefore, we can write γ before and after record removal as:

$$\gamma = \frac{|R \succ S|}{|R| \cdot |S|} \tag{1}$$

$$\gamma' = \frac{|R' \succ S|}{|R'| \cdot |S|} \tag{2}$$

We first consider the case where the removal of records from R may cause an increase of γ, i.e., $\gamma' \geq \gamma$. From the previous equation we can see that the maximum possible increase happens when $|R \succ S| = |R' \succ S|$ — the case when no removed records were previously dominating any records in S. It follows that:

$$|R'| \cdot |S| \cdot \gamma' = |R' \succ S| \leq |R \succ S| = |R| \cdot |S| \cdot \gamma$$

Dividing both sides by the positive number $|R'| \cdot |S|$ we obtain:

$$\gamma' \leq \frac{|R| \cdot |S|}{|R'| \cdot |S|} \cdot \gamma$$

Noticing that we can rewrite $|R| \cdot |S|$ as $|R'| \cdot |S| + (|R| - |R'|)|S|$ and setting $\frac{(|R|-|R'|)}{|R|} = \epsilon$ we conclude that

$$\gamma' \leq \gamma(1 + \epsilon)$$

Let us now consider the case where γ may decrease, i.e., $\gamma' \leq \gamma$. Similarly to the previous case, we can see that the maximum possible decrease happens when $|R \succ S| = |R' \succ S| + (|R| - |R'|)|S|$ — the case when all removed records were previously dominating all records in S. It follows that:

$$|R'| \cdot |S| \cdot \gamma' = |R' \succ S| \geq |R \succ S| - (|R| - |R'|)|S|$$

$|R \succ S|$ can be rewritten as $|R| \cdot |S| \cdot \gamma$ (Eq. 1). In addition, we can rewrite $|R| \cdot |S|$ as $|R'| \cdot |S| + (|R| - |R'|)|S|$. Setting $\frac{(|R|-|R'|)}{|R|} = \epsilon$ we conclude that

$$\gamma' = |R' \succ S| \geq |R \succ S| - (|R| - |R'|)|S| = (1 + \epsilon)\gamma - \epsilon$$

Because $\gamma \geq \frac{1}{2}$ (to enforce asymmetry), $\gamma(1 - \epsilon) \leq (1 + \epsilon)\gamma - \epsilon$. As a consequence,

$$\gamma(1 - \epsilon) \leq \gamma'$$

The case where $S \succ_\gamma R$ follows similarly. □

Now consider two movies, one with a very high quality, say 10, and another of lower quality, say 5. If we take an average of these two values, as it is done in alternative approaches, we obtain a quality of 7.5. We can already notice that the same value is returned by averaging two movies whose actual quality is 7.5, reducing two very different scenarios to the same generic description.

However, a more subtle problem consists in the fact that taking the value *in the middle* introduces the assumption that users can be described by a *linear* utility function, stating for example that a quality of 10 is exactly *twice as a quality of* 5. On the contrary, we can think of users for which movies whose quality is higher than 9 are good and those of quality less than 9 are just bad. In this case, the "average" would not correspond to the *arithmetic* average. In this context, *stability to monotone data transformations* indicates that the result of a method is not based on any assumption regarding the users, except the fact that they consistently prefer higher values to lower ones (monotonicity). This kind of stability is one of the basic requirements of the original definition of record-wise skyline [5], even if it was only informally stated in the original paper and later formalized only for linear scoring functions [16].

DEFINITION 4 (STAB. TO MONOTONE TRANSFORMATIONS). *Let R and S be groups of records in d dimensions. Let $R' = \{\phi(r) \mid r \in R\}$ and $S' = \{\phi(s) \mid s \in S\}$ where $\phi = \langle \phi_1, \ldots, \phi_d \rangle$ is a tuple of d monotone scalar functions and $\phi(\langle r.1, \ldots, r.d \rangle) = \langle \phi_1(r.1), \ldots, \phi_d(r.d) \rangle$. A group domination predicate $\succ_g$ is* robust *if $R \succ_g S \Leftrightarrow R' \succ_g S'$.*

PROPOSITION 2. *Definition 3 is stable to monotone transformations.*

PROOF. Definition 3 relies on counting pairs of records satisfying traditional record-based dominance, which is itself stable to monotone transformations. As all the single record-based relationships remain unchanged one by one, so does their number. $\square$

3. COMPLEXITY AND ALGORITHMS

A general approach that has been used to compute the skyline is to apply some heuristics to prune dominated records and limit the number of dominance checks. However, dealing with groups introduces an additional source of complexity. In fact, there are properties that hold for single-record skyline queries but are not satisfied by aggregate skylines. This prevents us from applying some basic optimization techniques, making the aggregate skyline problem computationally challenging.

We start this section by discussing two of these properties: skyline containment and transitivity. Skyline containment would allow us to take advantage of a record-wise skyline in computing the aggregate one, and transitivity has a potentially tremendous impact on the optimization of aggregate skyline queries. In fact, it is a basic assumption of most traditional skyline computation methods. Unfortunately, both properties do not hold in general for groups of records. However, we will see that we can define a weaker concept of transitivity that can be practically used for optimization.

A possible way to speed up the computation of a (group) skyline consists in identifying an initial set of groups belonging to the skyline. Then, these groups can be used to prune others. In our working example, we might identify some skyline records, i.e., movies, then select the directors of these movies and use them for pruning. However, this is possible only if directors with a skyline movie are also in the group skyline:

PROPERTY 3 (SKYLINE CONTAINMENT). *Let $r \in U_r, r \in R, R \in U_g$. If $r \in Sky(U_r)$ and $r \in R$ then $R \in Sky_\gamma(U_g)$.*

Unfortunately, this is not true for our definition:

PROPOSITION 3. *For Definition 3 skyline containment does not hold.*

PROOF. We can prove this by providing a counterexample. Let $G_1 = \{(5,5),(1,1),(1,2)\}, G_2 = \{(2,3)\}$. We can see that $p(G_2 \succ G_1) = \frac{2}{3}$, so G_1 is not in the group skyline if $\gamma < \frac{2}{3}$. However, G_1 contains the skyline record $(5,5)$. $\square$

It is worth noticing that this might motivate us to look for a different definition satisfying also this property. However, the following theorem shows that we cannot expect any single definition to be strictly better than ours from this point of view, whenever the definition is "reasonable". Reasonable means that if all records of R dominate all records of S then $R \succ_g S$, i.e., strict dominance holds – which is the most basic requirement for a definition of aggregate-skyline:

THEOREM 1. *No definition of group-domination where strict dominance holds satisfies both properties 3 and 2.*

PROOF. Let us consider two groups R' and S where all records in S dominate all records in R'. Now, let us add a skyline record to R, dominating all records in S. Let us assume that both Properties 3 and 2 hold. Than, by Property 3 R belongs to the group skyline, implying that $S \succ_\gamma R \implies \gamma < .5$. At the same time, by Property 2 we can state that $1 = \gamma' \leq \gamma(1+\epsilon) < .5(1+\epsilon)$. As $\epsilon \in [0,1]$, we can conclude that $1 < 1$, showing that the assumption that both properties hold is absurd. $\square$

A second property with a very relevant practical impact on the computation of aggregate skylines is *transitivity*. Transitivity is one of the key concepts enabling pruning in traditional skyline algorithms and making aggregate skylines challenging to compute.

PROPERTY 4 (TRANSITIVITY). *If $R \succ_g S$ and $S \succ_g T$ then $R \succ_g T$.*

This property is fundamental for optimization reasons, because it is used to avoid comparisons between groups: if we verify that R dominates S we can discard S without any effect on the result. However, if transitivity does not hold, there may be a third group T that is dominated by S and not by R. Discarding S before comparing it to T would result in incorrectly including T in the result. As a consequence, without transitivity we basically have to check dominance between *all pairs of groups*. Unfortunately, like for asymmetry this property clearly holds for single records but is no longer satisfied in general when we compare groups.

PROPOSITION 4. *Definition 3 is not transitive.*

PROOF. By counterexample: consider Figure 6, with $\gamma = .5$. We can see that R does not dominate T because $p(R \succ T) = .5$. However, $R \succ_\gamma S$ and $S \succ_\gamma T$. $\square$

To overcome this problem, we have found a weaker notion of transitivity that still holds for groups and can thus be used for pruning:

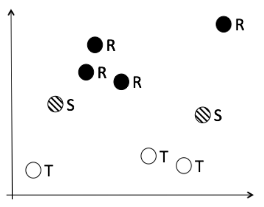


Figure 6: Transitivity does not hold among groups: we represent three groups R, S and T where $R \succ_\gamma S$, $S \succ_\gamma T$ but $R \not\succ_\gamma T$ ($\gamma = .5$)

PROPOSITION 5 (WEAK TRANSITIVITY). *If $R \succ_{\overline{\gamma}} S$ and $S \succ_{\overline{\gamma}} T$ then $R \succ_\gamma T$, where $\overline{\gamma} = 1 - \frac{\sqrt{1-\gamma}}{2}$.*

PROOF. The main idea behind this proof is to reduce the transitivity property to a case of matrix multiplication. Given any enumeration of the elements of groups R and S, we can model the dominance relation between R and S as a $|R| \times |S|$ matrix $RS = (m_{ij})$, where:

$$m_{ij} = \begin{cases} 1 & \text{if } r_i \succ s_j \\ 0 & \text{otherwise} \end{cases}$$

The same can be done for groups S and T, and we call this matrix ST. We now state two fundamental facts:

1. The percentage $pos(RS)$ of non-zero entries in matrix RS corresponds to $p(R \succ S)$, indicating that $R \succ_\gamma S$ for every $\gamma < pos(RS)$. We call such a matrix a *Domination Matrix* for R and S.
2. If RS and ST are Domination Matrices, $RT = RS \times ST$ is also a Domination Matrix for R and T.

This link between record domination, group domination and matrix multiplication gives us a powerful tool to study transitivity relations. As an example, the following are respectively the matrices RS, ST and RT for the groups in Figure 6:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

We can in fact see that $pos(RS) = \frac{5}{8}$, $pos(ST) = \frac{2}{3}$ and $pos(RT) = .5$, and then conclude that $R \succ_{.5} S$, $S \succ_{.5} T$ but $R \not\succ_{.5} T$.

Now we can prove Proposition 5. Let us consider the generic matrices in Figure 7 (upper part), where $pos(RS) = pos(ST) = \frac{1+(1-\alpha)}{2} = 1 - \frac{\alpha}{2}$. First, we claim that, fixed α, this is the configuration of values in RS and ST that determines the *smallest* number of positive entries in RT. The reason of this claim is to show that given some probability of domination between R and S and between S and T we have a guaranteed lower bound for the probability that R dominates T, corresponding to the percentage of non-zero entries ($pos(RT)$) in the product matrix RT. The claim that this configuration leads to the smallest gray area in RT (therefore, a lower bound) can be verified by noticing that any change of zero and non-zero cells in the first two matrices leads to additional positive values in RT. This is shown in the lower part of Figure 7, where some positive (gray) values in RS are moved to the null (white) part of the matrix and this increases the gray area in RT.

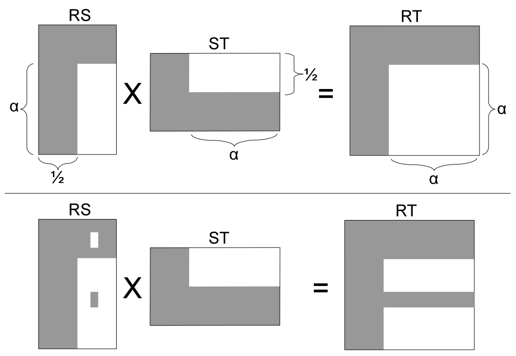


Figure 7: Domination Matrices: gray parts represent non-zero values, white parts represent zero values. On top, a configuration leading to the smaller probability of domination (size of the gray area) between R and T.

We can thus state that given *any* pair of Dominance Matrices with at least $1 - \frac{\alpha}{2}$ positive values, $pos(RT)$ is *at least* $1 - \alpha^2$, which is the relative size of the gray area inside RT.

Using the facts that:

- for a Dominance Matrix RS, $pos(RS) = p(R \succ S)$, then
- $p(R \succ T) = 1 - \alpha^2$, therefore $\alpha = \sqrt{1 - p(R \succ T)}$
- $p(R \succ S) = p(S \succ T) = 1 - \frac{\alpha}{2} = 1 - \frac{\sqrt{1-p(R \succ T)}}{2}$

we may set $\overline{\gamma} = 1 - \frac{\sqrt{1-\gamma}}{2}$ and conclude that if $R \succ_{\overline{\gamma}} S$ and $S \succ_{\overline{\gamma}} T$ then $R \succ_\gamma T$. $\square$

3.1 SQL and Nested Loop

Aggregate skyline queries can be expressed in SQL as in Algorithm 1. Here we are also assuming the presence of an attribute (`num`) storing the cardinality of each class, that can be pre-computed if not available, and the query is only in 2 dimensions. Despite the simplified scenario, this direct SQL implementation is inefficient, as illustrated in Figure 8.

Algorithm 1 SQL aggregate skyline

```
Require: For every record, a num attribute with the number of
    movies for that director
Ensure: X directors are in the γ-skyline
    select distinct director
    from movies
    where director not in (
    select X.director
    from movies X, movies Y
    where ((Y.votes > X.votes and Y.rank >= X.rank) or (Y.votes
    >= X.votes and Y.rank > X.rank))
    group by X.director, Y.director
    having 1.0*count(*)/(X.num*Y.num)> .5)
```

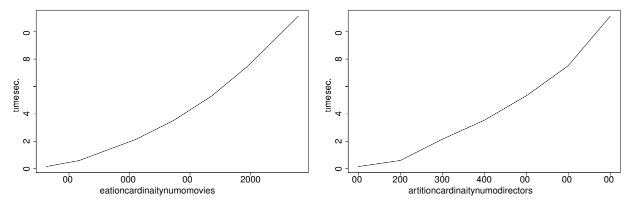


Figure 8: Scalability of SQL implementation on sqlite

In fact, this query basically corresponds to a nested loop to compare every class against each other, and an internal nested loop to compare all records from one class to all records of the other, this for every pair of classes. Algorithm 2 presents a slightly optimized version of this approach, that we will use as our baseline.

Algorithm 2 Nested Loop aggregate skyline

```
Input: $G$ (set of groups), $\gamma$
for $g1 \in G$ do
    for $g2 \in G$ do
        if $g1 \leq g2$ then
            skip this comparison;
        end if
        if $g1 \succ_{\gamma} g2$ then
            mark $g2$ as dominated
        else if $g2 \succ_{\gamma} g1$ then
            mark $g1$ as dominated
        end if
    end for
end for
```

The complexity of Algorithm 2 is defined by the number of dominance checks, which in turn depends on dimensionality d. Consider a dataset with n groups $\{c_1, \ldots, c_n\}$. The time complexity of the algorithm is:

$$\sum_{i,j \in [1,n], i<j} \text{cost}(c_i, c_j) \tag{3}$$

where

$$\text{cost}(c_i, c_j) = O(\sum_{k=1}^{|c_i|} \sum_{l=1}^{|c_j|} d) \tag{4}$$

This formula highlights two nested quadratic components: one, external, involves the comparison of pairs of groups – Equation (3), and for each pair of groups we compare all pairs of records from the two groups – Equation (4). This emphasizes three main areas of optimization:

- **External**, to reduce the number of comparisons of groups.
- **Internal**, for every given pair of groups that we could not prune "externally", to reduce the number of comparisons of pairs of records.
- **Global**, taking into consideration the dependencies between the number of pairs of groups and the number of pairs of records.

3.2 External optimization

We propose three algorithmic approaches to reduce the number of comparisons between groups. The first consists in applying our definition of weak transitivity to the basic nested loop approach: this is indicated in Algorithm 3 (lines 10-22).

Algorithm 3 Transitive aggregate skyline

```
 1: Input: $G$ (set of groups), $\gamma$
 2: for $g1 \in G$ do
 3:     if $g1$ is strongly dominated then
 4:         skip $g1$
 5:     end if
 6:     for $g2 \in G$ do
 7:         if $g1 \leq g2$ then
 8:             skip this comparison
 9:         end if
10:         if $g2$ is strongly dominated then
11:             skip $g2$
12:         end if
13:         if $g1 \succ_{\overline{\gamma}} g2$ then
14:             mark $g2$ as strongly dominated
15:         else if $g1 \succ_{\gamma} g2$ then
16:             mark $g2$ as dominated
17:         else if $g2 \succ_{\overline{\gamma}} g1$ then
18:             mark $g1$ as strongly dominated
19:             end processing of $g1$
20:         else if $g2 \succ_{\gamma} g1$ then
21:             mark $g1$ as dominated
22:         end if
23:     end for
24: end for
```

The second main method consists in accessing the first groups in Equation (3), i.e., those indicated as c_i, in sorted order. This mimics an approach already used in traditional skyline methods, where however only single records must be ordered. In the context of aggregate skylines we have to sort *groups*, and we do this according to the sum of the distances between the origin and the minimum and maximum corners of the minimum bounding box containing all the records in the group. This is shown in Algorithm 4.

Algorithm 4 Sorted aggregate skyline

```
 1: Input: $G$ (set of groups), $\gamma$
 2: for $g \in G$ do
 3:     insert $g$ into Priority Queue $Q$
 4: end for
 5: while $Q$ is not empty do
 6:     $g1 = Q$.poll
 7:     if $g1$ is strongly dominated then
 8:         skip $g1$
 9:     end if
10:     Proceed like in Algorithm 3, line 6
11: end while
```

The third and final *external* optimization consists in using a spatial index so that given a group we choose the groups to compare to it only among the groups that can dominate it. This is indicated in Algorithm 5, line 11. Here $g.min$ and $g.max$ indicate respectively the minimum and the maximum corners of the minimum bounding box of the records in group g. In Figure 9(a), g's bounding box is shown as a dashed rectangle, the window query is represented in gray and we visualize four potentially dominating groups (with their maximum corners inside the window).

Algorithm 5 Indexed aggregate skyline

```
1:  Input: $G$ (set of groups), $\gamma$
2:  for $g \in G$ do
3:      insert $g$ into Priority Queue $Q$
4:      insert $g$.max into spatial index $I$
5:  end for
6:  while $Q$ is not empty do
7:      $g1 = Q$.poll
8:      if $g1$ is strongly dominated then
9:          skip $g1$
10:     end if
11:     $G' =$ I.windowQuery(space dominating g1.min)
12:     for $g2 \in G'$ do
13:         if $g1 = g2$ then
14:             skip this comparison
15:         end if
16:         if $g2$ is strongly dominated then
17:             skip $g2$
18:         end if
19:         Proceed like in Algorithm 3, line 13
20:     end for
21: end while
```

3.3 Internal optimization

Now assume that we have not been able to avoid a comparison between groups g_1 and g_2. In some cases it is not necessary to check the dominance relationship between all pairs of records from the two groups. The first consideration is that after having checked some pairs it may be possible to stop if one of the following conditions holds. Let δ_{12} be the percentage of pairs $r_1, r_2 \in g_1 \times g_2$ such that $r_1 \succ r_2$, δ_{21} the percentage of pairs where $r_2 \succ r_1$, and c the percentage of pairs of records compared so far. Then:

- if $\delta_{12} + 1 - c <= \gamma \wedge \delta_{21} + 1 - c <= \gamma$, then g_1 and g_2 are incomparable.
- if $\delta_{12} > \gamma \wedge \delta_{12} + 1 - c <= \overline{\gamma}$, then $g_1 \succ_\gamma g_2$
- if $\delta_{12} > \overline{\gamma}$, then $g_1 \succ_{\overline{\gamma}} g_2$
- if $\delta_{12} > \gamma \wedge \delta_{21} + 1 - c <= \overline{\gamma}$, then $g_2 \succ_\gamma g_1$
- if $\delta_{21} > \overline{\gamma}$, then $g_2 \succ_{\overline{\gamma}} g_1$

In addition to this stopping rule, we can also compare the extreme points of the minimum bounding boxes of the groups before starting the pair-wise comparison, with the objective of pruning some alternatives. Figure 9(b-c) shows two possible cases. When g_2's minimum corner dominates g_1's maximum corner, this indicates strict domination without any need to compare any records (9(b)). With regard to Figure 9(c), we can state that all records in the area marked as A are dominated by all the records in g_2, and all records in the area marked as C dominate all the records in g_1, without any additional comparison. On the contrary, records, e.g., in the area marked as B are processed using a normal nested loop approach.

3.4 Global optimization

In Section 3.2 we have discussed how to reduce the number of comparisons between groups. This general idea is one of the basic optimization tools in traditional skyline query processing. However when we deal with groups of records the cost of comparing different pairs can vary dramatically. For example, assume we have

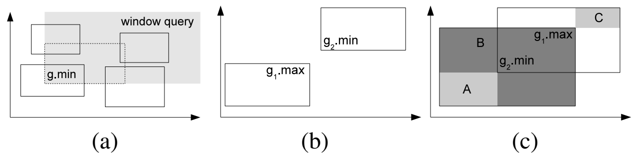


Figure 9: Window query, to obtain possible dominating groups, and comparison of groups of records approximated by their bounding boxes

four groups g_1, g_2, g_3 g_4 with cardinality respectively 1, 5, 6 and 7. The sequence of comparisons (g_1, g_2), (g_1, g_3), (g_1, g_4), (g_2, g_3) is longer than the sequence (g_1, g_2), (g_2, g_3), (g_3, g_4). However, the first sequence may result in at most 48 dominance checks, while the second can lead to up to 77 checks. The reason lies in the quadratic nature of the cost to compare two groups of records, suggesting that a good heuristic can reduce the complexity of the internal term of Equation (3) by reducing the comparisons between high-cardinality groups.

This is particularly important knowing that in many real datasets data follow some kind of long tail distribution (e.g., Zipf or Pareto distributions) and would then present a very small number of problematic (i.e., large) groups and many less problematic small groups.

3.5 Summary of improvements

In summary, in this section we have introduced the following optimization strategies:

1. Exploiting weak transitivity (**external**).
2. Sorted access to the groups (**external**).
3. Spatial indexing (**external**).
4. Stopping rule (**internal**).
5. Approximation by bounding boxes (**internal**).
6. Accessing small groups first (**global**).

In the next section we perform an experimental evaluation of these strategies on both synthetic and real data.

4. EXPERIMENTAL ANALYSIS

In this section we evaluate the efficiency of our algorithms on synthetic and real data. Synthetic data is used to test the impact of the variation of several parameters: the number of attributes (dimensionality), the amount of overlapping of the minimum bounding boxes of the groups, the total number of records and the number of records per class. The algorithms included in the evaluation are: Nested Loop with stop condition (NL), Transitive with stop condition (TR), Sort-based sorting on the size and distance from the origin of the minimum corner of the minimum bounding box (SI), Index-based (IN) and Index-based with approximation by bounding boxes (LO). Where not specified, the default values for the experiments are: 10 000 records, 100 average records per class, records in each class spread over 20% of the data space, dimensionality 5, and $\gamma = .5$.

4.1 Synthetic data

Figure 10 shows the effect of dimensionality on anti-correlated, independent and correlated data distributions, that are the typical data distributions used to evaluate skyline algorithms — anti-correlated is often the most challenging because a large part of the input is in the skyline. This figure emphasizes the superiority of the index-based approaches, that are consistently more efficient than the others. On independent and correlated data also the transitive and sort-based methods improve significantly.

On the contrary. Figure 11 shows how a large overlapping of the classes makes the method based solely on indexing less efficient than the others, including the Nested Loop approach. Also in this case easier datasets reduce the differences between the different approaches.

Figures 12, 13(a) and 13(b) present the result of scalability tests on the number of records, emphasizing different interesting behaviors. Figure 12 shows results similar to the ones obtained varying the dimensionality of the dataset: index-based methods outperform the others on anti-correlated data and the gap is reduced on the other data distributions. However, Figure 13(a) shows that when the distribution of records inside classes follows an exponential law (Zipfian, in this case) the approach based on sorting (global optimization) improves its performance, still remaining less efficient than index-based methods. Finally, Figure 13(b) shows the performance of index-based methods on a wider range of values, and Figure 13(c) shows the effect of varying the number of records per class.

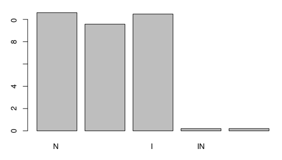
4.2 Real data

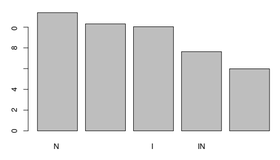
Finally, we present the results of the execution of our algorithms on a real dataset of about 15 000 records containing statistics for all basketball players and regular seasons since 1979[3]. On these data, we computed a group skyline maximizing attributes *points, rebounds, assists, steals, blocks, field goals, free throws* and *three points* (per game). As grouping attributes, we used both single and multiple attributes. The results are indicated in Figure 14 and confirm previous evidence obtained on synthetic datasets. The proposed algorithms are consistently more efficient than the baseline implementation. In one case (bottom left), the improvement is only around 15%. This is a case considering many skyline attributes (8) and a very large number of classes with a few records, i.e., a case not very different from a traditional record skyline where group-related optimizations are less relevant. In other cases, improvements of up to two orders of magnitude are achieved.

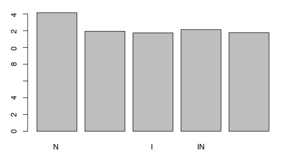
5. RELATED APPROACHES

The skyline operator was proposed in [5], bringing the existing concept of *Pareto front* to the attention of the database community. Since then, two main research directions have been followed. Some researchers have defined new variations of this operator, including representative [14], reverse [7] and uncertain skylines [18]. Others have focused on its efficient execution, using several approaches like data presorting [6], anytime processing [15], indexing [17] or distributed computing [9]. In this context, some research efforts have been devoted to the improvement of skyline computation when the operator is used in conjunction with data transformations, as in the case of dynamic and spatial skylines [17, 20], and also skylines executed together with other SQL operators, in particular joins and grouping.

[3]`http://www.databasebasketball.com`


(a) Team, 4 attr.


(b) Team, Year, 4 attr


(c) Player, 8 attr.

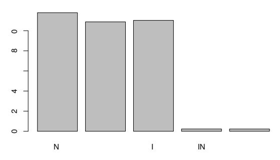

(d) Year, 8 attr.

Figure 14: Efficiency on a real dataset (NBA data), grouped by different attributes. We also indicate the number of skyline attributes used in each query

In all previous works on skyline processing together with other SQL operators the semantics of skyline was unchanged, and defined in isolation with respect to the other operators: a skyline with a `group by` was just a skyline obtained before or after the result of a `group by`. This is the case in [10], where grouping is computed on the result of a skyline, and in [2] and [1], where the skyline is computed after grouping. As a result, in all these works the skyline is computed on single records of scalar values, as in its original definition, and not on groups of records as in our work.

Some works have also used the keyword *group* in relation with skyline queries to indicate a different problem: [23] and [12] show how to find groups of k records not dominated by other groups, without using a *group by* operator. In addition, domination is defined after reducing the groups to single records, so no concept of skyline of groups is introduced in this work. [3] uses the keyword *aggregation*, but only to refer to the arithmetic combination of attributes, e.g., the sum or difference of values – also in this case, no grouping is involved.

Some related papers have considered to compute the skyline on subsets of a relation: [17, 4] introduce the concept of skyline *inside* groups, also applied in [22] to data warehousing. Also in this case skylines are computed over single records.

[11] introduced the concept of *thick skyline*, where the skyline is extended with similar records and thus records not in the skyline can also be returned. However there is no `group by` involved, records are still treated one by one to check dominance, and in addition this approach is not robust: expressing movie popularity using the number of votes (e.g., 362 000) instead of the thousands of votes (e.g., 362) may change the result of the query.

An apparently unrelated but still relevant set of works concerns skylines over uncertain data, first proposed in [18] and then extended to other kinds of queries like contextual [19], reverse [13] and top-k skylines [23]. In these works, every record may have multiple alternative values, therefore algorithms developed to manage uncertain records have already faced the problem of dealing with multiple

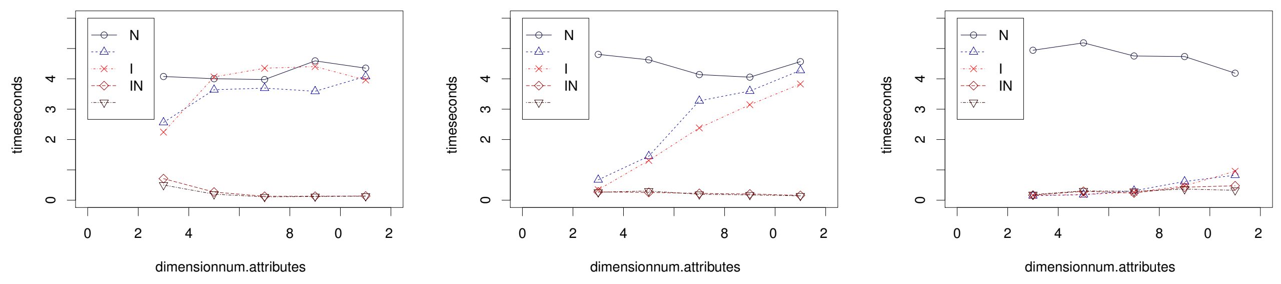


Figure 10: Varying dimensionality for anti-correlated, independent, and correlated distributions. Records are uniformly distributed into classes

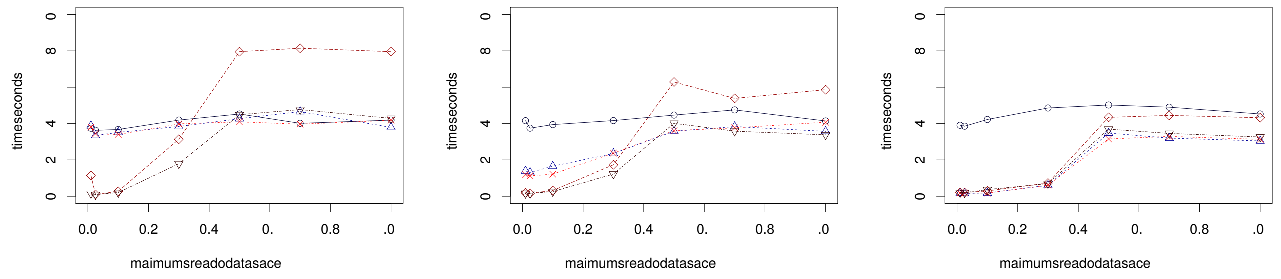


Figure 11: Varying group overlapping for anti-correlated, independent, and correlated distributions. Records are uniformly distributed into classes

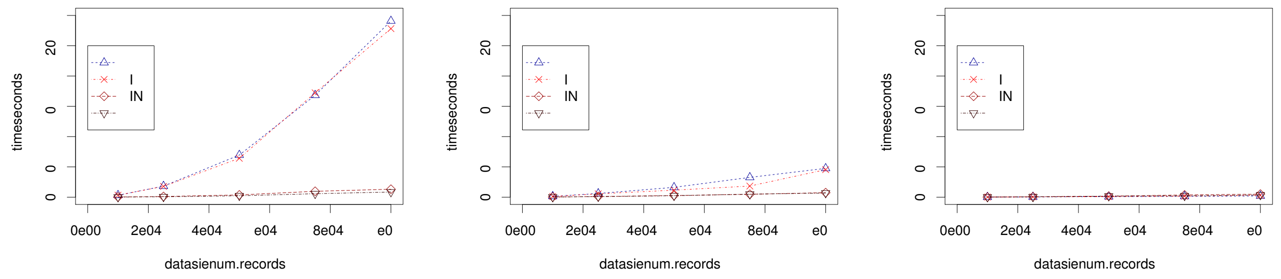
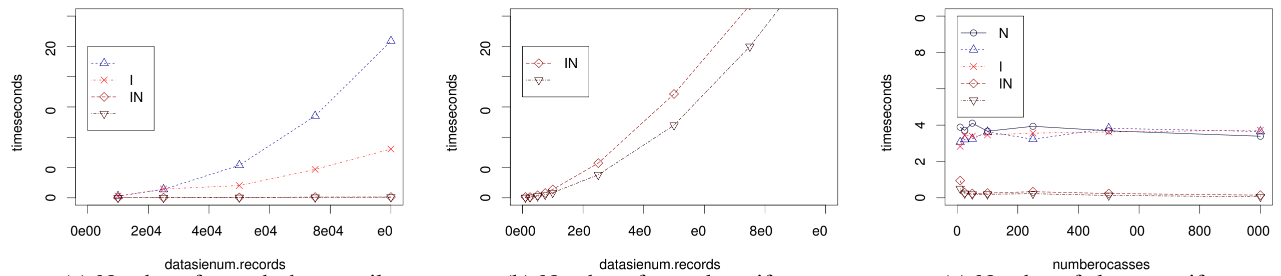


Figure 12: Scalability varying the number of records for anti-correlated, independent, and correlated distributions. Records are uniformly distributed into classes



(a) Number of records, heavy tail (b) Number of records, uniform (c) Number of classes, uniform

Figure 13: Scalability on anticorrelated data, varying the number of records (a, b) and the number of records per class (c), with records distributed to the classes with heavy tail (a) and uniform (b, c) distributions.

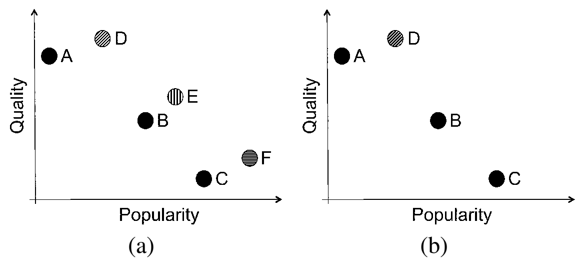


Figure 15: Difference between probabilistic skyline and aggregate skyline

instances. However, the definition of probabilistic skyline is different from the definition of aggregate skyline, because there is no concept of group.

Figure 15 illustrates the difference between aggregation and uncertainty. In Figure 15(a) we have six movies directed by four directors, with three of them directed by the same director (A, B and C, in black) and D, E and F directed by a different director each. To see if the black director should be in the group skyline or not, we should check if one of the three other directors dominates him. For example, we may start by comparing him with the director of movie D (Figure 15(b)). Clearly, D is preferable *only to one* of the three movies by the black director, and this means that depending on our definition of domination the black director may well be in the group skyline. A comparison with E and F leads to the same conclusion. On the contrary, consider a probabilistic skyline. In this context, A, B and C would represent three possible locations of a single record, only one of which is true. As a consequence, this record will never be in the skyline, because for every alternative there will always be another record dominating it.

6. CONCLUDING REMARKS

In this paper we have introduced a new kind of query called *aggregate skyline*. This query combines two basic operators: skyline and group by. After discussing the relevance and novelty of this approach, we have provided a formal definition with an intuitive interpretation and satisfying some basic properties, including asymmetry and different kinds of stability. However, we have also shown that some basic properties with a relevant impact on query optimization do no longer hold when groups are concerned. Therefore, we have introduced and experimentally tested several optimization approaches, including a new definition of weak transitivity that enables pruning at group level. The experimental results on both real and synthetic data show that our algorithmic approaches can reduce the computation time by up to two orders of magnitude with respect to a direct SQL implementation of the query. However, there are some specific data distributions that remain challenging from a computational point of view, opening toward the development of customized query optimization methods addressing them.

7. ACKNOWLEDGMENT

This work has been supported in part by the Danish Council for Strategic Research, grant 10-092316. We thank Gianluca Gozzoli for adapting the synthetic data generator presented in [5].

8. REFERENCES

[1] S. Antony, P. Wu, D. Agrawal, and A. E. Abbadi. Aggregate Skyline: Analysis for Online Users. In *2009 Ninth Annual International Symposium on Applications and the Internet*, pages 50–56. IEEE, July 2009.

[2] S. Antony, P. Wu, D. Agrawal, and A. El Abbadi. MOOLAP: Towards Multi-Objective OLAP. In *Proc. ICDE*, 2008.

[3] A. Bhattacharya and B. P. Teja. Aggregate Skyline Join Queries: Skylines with Aggregate Operations over Multiple Relations. In *COMAD*, 2010.

[4] C. Boehm, A. Oswald, C. Plant, M. Plavinski, and B. Wackersreuther. SkyDist : Data Mining on Skyline Objects. pages 461–470, 2010.

[5] S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline operator. In *Proc. ICDE*, pages 421–430, 2001.

[6] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with Presorting. In *Proc. ICDE*, pages 717–719, 2003.

[7] E. Dellis and B. Seeger. Efficient Computation of Reverse Skyline Queries. In *Proc. VLDB*, pages 291–302, 2007.

[8] M. Goncalves, L. Tineo, U. S. Bolívar, and D. D. Computación. Fuzzy Dominance Skyline Queries. In *DEXA*, pages 469–478, 2007.

[9] K. Hose and A. Vlachou. A survey of skyline processing in highly distributed environments. *The VLDB Journal*, pages 1–26, 2011.

[10] W. Jin, M. Ester, Z. Hu, and J. Han. The Multi-Relational Skyline Operator. In *Proc. ICDE*, 2007.

[11] W. Jin, J. Han, and M. Ester. Mining thick skylines over large databases. In *PKDD*, pages 255–266. Springer-Verlag New York Inc., 2004.

[12] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das. On Skyline Groups. In *CIKM*, 2012.

[13] X. Lian and L. Chen. Reverse skyline search in uncertain databases. *ACM Transactions on Database Systems*, 35(1):1–49, 2010.

[14] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting Stars: The k Most Representative Skyline Operator. In *Proc. ICDE*, 2007.

[15] M. Magnani, I. Assent, and M. L. Mortensen. Anytime skyline query processing for interactive systems. In *VLDB Workshop on Ranking in Databases*, 2012.

[16] D. Nanongkai, A. D. Sarma, A. Lall, R. J. Lipton, and J. Xu. Regret-minimizing representative databases. *Proceedings of the VLDB Endowment*, 3(1-2):1114–1124, 2010.

[17] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM TODS*, 30(1):41–82, 2005.

[18] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic Skylines on Uncertain Data. In *Proc. VLDB*, 2007.

[19] D. Sacharidis, A. Arvanitis, and T. Sellis. Probabilistic Contextual Skylines. In *Proc. ICDE*, 2010.

[20] M. Sharifzadeh and C. Shahabi. The spatial skyline queries. In *Proc. VLDB*, 2006.

[21] W. Yan, C. Zanni-merk, and F. Rousselot. Skyline Adaptive Fuzzy Query. In *Proceedings of the 15th international conference on Knowledge-based and intelligent information and engineering systems*, pages 345–354, 2011.

[22] M. L. Yiu, E. Lo, and D. Yung. Measuring the Sky: On Computing Data Cubes via Skylining the Measures. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):492–505, 2012.

[23] Y. Zhang, W. Zhang, X. Lin, B. Jiang, and J. Pei. Ranking uncertain sky: The probabilistic top-k skyline operator. *Information Systems*, 36(5):898–915, 2011.