Catching the Best Views of Skyline: A Semantic Approach Based on Decisive Subspaces *

Jian Pei[1] Wen Jin[1] Martin Ester[1] Yufei Tao[2]

[1] Simon Fraser University, Canada, {jpei, wjin, ester}@cs.sfu.ca
[2] City University of Hong Kong, Hong Kong, taoyf@cs.cityu.edu.hk

Abstract

The skyline operator is important for multi-criteria decision making applications. Although many recent studies developed efficient methods to compute skyline objects in a specific space, the fundamental problem on the *semantics* of skylines remains open: Why and in which subspaces is (or is not) an object in the skyline? Practically, users may also be interested in the skylines in any subspaces. Then, what is the relationship between the skylines in the subspaces and those in the super-spaces? How can we effectively analyze the subspace skylines? Can we efficiently compute skylines in various subspaces?

In this paper, we investigate the semantics of skylines, propose the subspace skyline analysis, and extend the full-space skyline computation to subspace skyline computation. We introduce a novel notion of *skyline group* which essentially is a group of objects that are coincidentally in the skylines of some subspaces. We identify the *decisive subspaces* that qualify skyline groups in the subspace skylines. The new notions concisely capture the semantics and the structures of skylines in various subspaces. Multidimensional roll-up and drill-down analysis is introduced. We also develop an efficient algorithm, *Skyey*, to compute the set of skyline groups and, for each subspace, the set of objects that are in the subspace skyline. A performance study is reported to evaluate our approach.

1 Introduction

It has been well recognized that the *skyline operator* is important for multi-criteria decision making applications. A (classic) illustrative example of skyline queries is to search for hotels in Nassau (Bahamas) which are cheap and close to the beach [2]. Suppose each hotel has two attributes: the price and the distance to the beach. Hotel A *dominates* hotel B (or, A is a better choice than B in the context of this example) if $A.price \leq B.price$, $A.distance \leq B.distance$ and at least one inequality holds. Those hotels not dominated by others in terms of price and distance to the beach form the skyline. In other words, the skyline hotels are all possible trade-offs between price and distance to the beach that are superior to other hotels.

There are many recent studies on efficient methods for skyline computation (see Section 5 for a brief review). However, the fundamental questions about the semantics of skyline remain open.

Example 1 (Intuition)**.** Consider a set of 5 objects in 2-d space (X, Y) as shown in Figure 1. It is easy to verify that objects a, b and c are in the skyline in space (X, Y) since each of them is not dominated by any other objects.

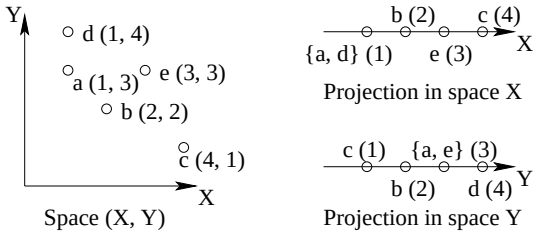


Figure 1: An example showing the intuition.

In the same figure, we also plot the projections of the objects on dimensions X and Y, respectively. In

* The research of Jian Pei is supported in part by NSERC Grant 312194-05 and NSF Grant IIS-0308001. The research of Yufei Tao is supported in part by Grant CityU 1163/04E from the RGC of HKSAR. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

subspace X, the projections of a and d collapse. Both are in the subspace skyline of X. In subspace Y, the projection of c is in the subspace skyline. Since all objects collapse in the trivial subspace $\emptyset$, hereafter, we use the term "subspace" to refer to only non-empty ones except for specifically mentioned.

Although a, b and c all are skyline objects in the full space (X, Y), there are some tricky differences among them. Both a and c have some projections which are in some subspace skylines (i.e., subspaces X and Y, respectively), but no projection of b is in subspace skyline of any proper subspace. A closer look finds that a taking value 1 on dimension X is already sufficient to qualify it as a skyline object. Similarly, the value 1 of c on dimension Y is critical for the skyline membership of c. On the other hand, b is a skyline object only if both dimensions X and Y are considered – it needs two dimensions to qualify.

Although both d and e are not in the skyline in space (X, Y), they are still subtly different if we look at the subspaces. The projection of d is in the skyline in subspace X but e has no projection belonging to a subspace skyline. d is dominated by a, nevertheless, the dominance is "partial" – d takes the same value as a in dimension X and thus has the chance to be in the skyline in subspace X. ∎

While the skyline in Example 1, which involves a two dimensional space and only a few objects, is simple and easy to be perceived, the general situation may be much more complicated when many dimensions are involved. Nevertheless, the observations in Example 1 illustrate one important intuition. Whether an object is in the skylines of the full space or of some subspaces is determined by the values of the object in some *decisive subspaces*. The decisive subspaces and the values in those subspaces vary from object to object in the skyline. For a particular object, the values in its decisive subspaces justify *why and in which subspaces the object is in the skyline* – the *semantics* of the object with respect to skyline.

Why should we care about the semantics of skylines? Semantics is important to understand the data. For example, Section 6.1 analyzes a real data set which contains 17,226 records of Great NBA Players' seasonal performance from 1960 to 2001. Wilt Chamberlain's performance in 1960 is in the skyline of the full space, which can be identified by the conventional skyline computation methods. However, one may wonder which merits really make Wilt that outstanding. The semantics analysis in Section 6 shows that Wilt was outstanding in total rebounds in the season of 1960 by achieving the record of 2149 in the NBA history. The attribute of total rebounds is the decisive subspace that establishes his superior status. In fact, he was not exceptional in any other factors such as total assists. As another example, Michael Jordan does not hold any record in any single attribute. However, his performance in 1988 is in the skyline of subspaces (total points, total rebounds, total assists) and (games played, total points, total assists), and also in the skyline of the full space. Those two subspaces are decisive, and explain why Michael Jordan is an outstanding player. Clearly, such information cannot be captured by the traditional skyline computation.

The concepts of skyline groups and decisive subspaces can also be used immediately for efficiently answering the *skyline membership queries*: given an object or a group of objects, determine the subspaces where the object(s) are in the subspace skylines.

The investigation of skylines in subspaces naturally introduces the problem of *subspace skyline analysis and computation*: for a set of subspaces, find the objects and their projections that are in the skylines of the subspaces, and analyze their relationship. This type of queries is interesting and useful in practice since, more often than not, a user may want to interactively examine the skylines with respect to different combinations of attributes.

Motivated by the above observations, in this paper, we study the semantics of skyline objects and its applications on subspace skyline computation. We make the following contributions.

- *We develop a theoretical framework to answer the question about semantics of skyline: Why and in which subspaces is an object in the skyline?* The semantics of skyline objects is concisely captured by the novel notions of *skyline groups* and the corresponding *decisive subspaces*. The subspaces where an object (or a set of objects) is in the skyline can be effectively determined by the skyline groups that the object belongs to and their decisive subspaces.

- *We investigate the problem of subspace skyline analysis.* Skylines in subspaces can be concisely summarized by skyline groups. Moreover, skyline objects in the full space can be selected as the representatives in skyline groups. They catch the "contour" (i.e., technically, the projections) of the skylines. The multidimensional roll-up and drill-down analysis is useful to support the online analytic processing of skylines.

- *We present efficient algorithms for subspace skyline computation.* We develop an algorithm to compute both the set of skyline groups and, for each subspace, the set of objects that are in the subspace skyline. The algorithm applies the findings in the semantics research and recursively reduces the set of objects to be searched. Moreover, a local sorting technique is developed so that computing skylines in subspaces can be substantially faster than a naïve method running a skyline computation algorithm on every subspace from scratch.

Objects	A	B	C	D
x	1	4	5	7
y	1	3	6	7
z	2	3	5	8

Table 1: A set of objects as our running example.

- *A performance study using both synthetic and real data sets is conducted to evaluate our approach.* We showcase some interesting findings in the skyline semantic analysis using the real data set about technical statistics of NBA players, and justify why they are meaningful in practice. Moreover, we use benchmark synthetic data sets to test the efficiency and the scalability of our algorithm.

The rest of the paper is organized as follows. In Section 2, we introduce the notion of skyline groups and decisive subspaces, and justify how the new notions capture the semantics of objects with respect to skyline. In Section 3, we tackle the problem of subspace skyline analysis. In Section 4, we present an algorithm for subspace skyline computation. We review related work in Section 5. An extensive performance study is reported in Section 6. The paper is concluded in Section 7.

2 Semantics

In this section, we first illustrate the ideas. Then, we introduce the new notions. Last, we elaborate on how the new notions capture the semantics and answer skyline membership queries.

2.1 Ideas

If an object u is in the skylines of subspaces $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{C}_1 \subset \mathcal{C}_2$, can we declare that u is also in the skyline of any subspace $\mathcal{C}$ in between, i.e., $\mathcal{C}_1 \subseteq \mathcal{C} \subseteq \mathcal{C}_2$? This property is appealing since it may extremely simplify the determination of skyline membership in subspaces. Unfortunately, the general situation is far from being so simple.

Example 2 (Ideas)**.** Consider the objects in Table 1 as our running example. We obtain the following two observations.

First, object x is in the skylines of full space (A, B, C, D) and of subspace A. However, it is not in the skyline of subspace (A, B), since its projection, $(1, 4, *, *)$, is dominated by $(1, 3, *, *)$, the projection of y. This demonstrates that, in general, for subspaces $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{C}_1 \subset \mathcal{C}_2$, *even though an object is in the subspace skylines of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$*, it may not be in the subspace skyline of* $\mathcal{C}$ *in between.*

Second, objects x and y collapse in subspace (A, D). The projection $(1, *, *, 7)$ is in the subspace skyline of (A, D). Thus, any dimension values in subspaces A, D or (A, D) qualifying x as a subspace skyline object in those subspaces also establish the same qualification for y, and vice versa. In other words, *if a group of objects collapse in a subspace* $\mathcal{B}$ *and the shared projection is in the subspace skyline of* $\mathcal{B}$*, then the objects in the same group share the skyline membership in all subspaces of* $\mathcal{B}$*.* ∎

The observations in Example 2 lead to the following ideas.

- The skyline membership is not monotonic – being in the skyline of subspace $\mathcal{B}$ does not automatically qualify an object in the skyline of super-spaces of $\mathcal{B}$. The object may be dominated in the super-spaces of $\mathcal{B}$ by some other objects which have the same values in $\mathcal{B}$.
- Objects coincide and form groups in subspaces. The skyline memberships in subspaces are shared by all objects in the same group. The convergence and divergence of groups from subspace to subspace play critical roles in forming skylines of various subspaces. Therefore, it is critical to capture groups of objects that the shared projections are in the skylines of the subspaces.

2.2 Concepts

Hereafter, by default we consider a set of objects S in an n-dimensional space $\mathcal{D} = (D_1, \ldots, D_n)$, where dimensions $D_1, \ldots, D_n$ are in the domain of numbers. For the sake of brevity, we often do not explicitly mention S and $\mathcal{D}$ when they are clear in the context.

For objects $u, v \in S$, u is said to *dominate* v if $u.D_i \leq v.D_i$ for $1 \leq i \leq n$ and there exists at least one dimension D_{i_0} such that $u.D_{i_0} < v.D_{i_0}$. Object u is a *skyline object* if u is not dominated by any other objects in S.

The notion of skyline can be intuitively extended to subspaces.

Definition 1 (Subspace skyline)**.** A subset of dimensions $\mathcal{B} \subseteq \mathcal{D}$ $(\mathcal{B} \neq \emptyset)$ forms a (non-trivial) $|\mathcal{B}|$-*dimensional subspace* of $\mathcal{D}$. For an object u in space $\mathcal{D}$, the *projection* of u in subspace $\mathcal{B}$, denoted by $u_{\mathcal{B}}$, is a $|\mathcal{B}|$-tuple $(u.D_{i_1}, \ldots, u.D_{i_{|\mathcal{B}|}})$, where $D_{i_1}, \ldots, D_{i_{|\mathcal{B}|}} \in \mathcal{B}$ and $i_1 < \cdots < i_{|\mathcal{B}|}$.

The projection of an object u $(u \in S)$ in subspace $\mathcal{B} \subseteq \mathcal{D}$ is in the *subspace skyline* (of $\mathcal{B}$) if $u_{\mathcal{B}}$ is not dominated by any $w_{\mathcal{B}}$ in $\mathcal{B}$ for any other object $w \in S$. u is also called a *subspace skyline object (of* $\mathcal{B}$*).* ∎

For example, in Figure 1, the projections of both a and d are in the subspace skyline in subspace X, and the projection of c is in the subspace skyline in subspace Y.

Now, let us consider objects that collapse in subspaces. They form groups that are critical in our skyline analysis.

Definition 2 (C-group)**.** Let $G \subseteq S$ be a subset of objects and $\mathcal{B} \subseteq \mathcal{D}$ be a subspace. $(G, \mathcal{B})$ is a *coincident group* (or *c-group* for short) if all objects in G share the same values on all dimensions in $\mathcal{B}$. The *projection* of the group in $\mathcal{B}$, denoted by $G_{\mathcal{B}}$, is $u_{\mathcal{B}}$ where $u \in G$.

A c-group $(G, \mathcal{B})$ is *maximal* if no any other objects $v \in (S - G)$ share the same values as those in G on dimensions in $\mathcal{B}$, and objects in G do not share the same value on any other dimension $D \in (\mathcal{D} - \mathcal{B})$. ∎

Example 3 (C-group)**.** Consider objects x and y in Table 1. They share the same value on dimension A, and no other objects have the same value on A. Thus, x and y form a coincident group (or c-group for short) on A.

Although we cannot add new objects into the group $(\{x, y\}, A)$, it can be expanded by including more dimensions. x and y share the same values in A and D. Thus, we can maximize the group to include dimension D. The maximal c-group is $(\{x, y\}, (A, D))$. ∎

Given a subset of objects G, we define $\mathcal{I}(G)$ as the maximal set of dimensions that all objects in G share the same values. That is,

$$\mathcal{I}(G) = \{D | D \in \mathcal{D}, \forall u, v \in G : u.D = v.D\}.$$

Moreover, for a subspace $\mathcal{B}$ and a set of objects G, we define $\mathcal{O}(G, \mathcal{B})$ as the maximal set of objects that have the same values on dimensions in $\mathcal{B}$ as objects in G. That is,

$$\mathcal{O}(G, \mathcal{B}) = \{v | v \in S, \forall D \in \mathcal{B} \forall u \in G : v.D = u.D\}.$$

The above two operators can be used to derive maximal c-groups for any given subset of objects, or a subset of objects and a subspace. The following lemma gives the derivation, and can be shown based on the related definitions immediately.

Lemma 1 (C-group)**.** *For a given subset of objects G, $(\mathcal{O}(G, \mathcal{I}(G)), \mathcal{I}(G))$ is a maximal c-group. For a given subset of objects H and a subspace $\mathcal{B}$ such that all objects take the same values on all dimensions in $\mathcal{B}$, $(\mathcal{O}(H, \mathcal{B}), \mathcal{I}(\mathcal{O}(H, \mathcal{B})))$ is a maximal c-group.* ∎

We are particularly interested in maximal c-groups whose projections are in the skyline of some subspaces. Intuitively, we want to capture the subsets of values in their projections that are decisive to their skyline memberships.

Definition 3 (Skyline group and decisive subspace)**.** Maximal c-group $(G, \mathcal{B})$ is called a *skyline group* if $G_{\mathcal{B}}$ is in the subspace skyline of $\mathcal{B}$.

For skyline group $(G, \mathcal{B})$, a subspace $\mathcal{C} \subseteq \mathcal{B}$ is called *decisive* if (1) $G_{\mathcal{C}}$ is in the subspace skyline of $\mathcal{C}$; (2) $\mathcal{O}(G, \mathcal{C}) = G$; and (3) there exists no proper subspace $\mathcal{C}' \subset \mathcal{C}$ such that conditions (1) and (2) also hold for $\mathcal{C}'$.

The *signature* of skyline group $(G, \mathcal{B})$ is written as $Sig(G, \mathcal{B}) = \langle G_{\mathcal{B}}, \mathcal{C}_1, \ldots \mathcal{C}_k \rangle$, where $\mathcal{C}_1, \ldots \mathcal{C}_k$ are all decisive subspaces of the skyline group. ∎

Conditions (1) and (3) are straightforward. Condition (2) requires that the decisive subspace is exclusive to the group G. This reflects our intension to catch the decisive factors for a group of objects that are in the (subspace) skylines. We will revisit this point soon when we discuss the semantics.

Example 4 (Skyline group)**.** Consider the objects in Table 1 again. In the full space, x, y and z are unique and each of them is a skyline object, therefore, each of them forms a maximal c-group in space (A, B, C, D). Each group contains only one object.

For group $(x, ABCD)$, where x and $ABCD$ are shorthands for set $\{x\}$ and subspace (A, B, C, D), respectively, subspace CD is decisive. Please note that AD is not a decisive subspace for the group, since x collapses with y in AD and the maximal c-group in AD contains two objects, i.e., $\mathcal{O}(x, AD) = xy$. In other words, condition (2) in the definition is violated. Another decisive subspace for this group is AC. Thus,

$$Sig(x, ABCD) = \langle (1, 4, 5, 7), AC, CD \rangle.$$

As another example, for group (xy, AD), its projection is in the subspace skyline of AD. The group has two decisive subspaces, namely A and D. Thus,

$$Sig(xy, AD) = \langle (1, *, *, 7), A, D \rangle.$$

Similarly, we have $Sig(xz, C) = \langle (*, *, 5, *), C \rangle$. ∎

2.3 Semantics of (Subspace) Skyline Objects

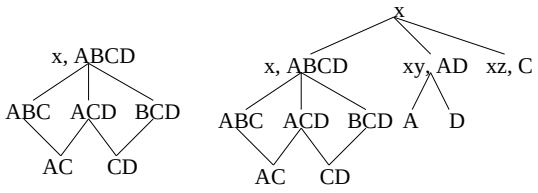
The question about the semantics asks: *For a given object or a group of objects, can we determine the subspaces where the projections of the object(s) are in the subspace skyline?*

Theorem 1 (Decisive subspace)**.** *For skyline group $(G, \mathcal{B})$, if $\mathcal{C}$ is decisive, then for any subspace $\mathcal{C}'$ such that $\mathcal{C} \subseteq \mathcal{C}' \subseteq \mathcal{B}$, $G_{\mathcal{C}'}$ is in the subspace skyline.*

Proof. We prove by contradiction. Suppose $G_{\mathcal{C}'}$ is not in the subspace skyline, and is dominated by $w_{\mathcal{C}'}$. Then, $w \notin G$. For each dimension $D \in \mathcal{C}'$, $w.D \leq G_{\mathcal{B}}.D$ and at least one inequality holds. On the other hand, since $\mathcal{C}$ is decisive, $G_{\mathcal{C}}$ is not dominated by projections of any other objects. Thus, $G_{\mathcal{C}} = w_{\mathcal{C}}$. That means, $\mathcal{O}(G, \mathcal{C}) \supset G$, which violates condition (2) in Definition 3. ∎

Theorem 1 indicates how decisive subspaces capture the semantics of skyline objects: *The skyline membership of an object or a group of objects is established by its decisive subspaces.*

Example 5 (Semantics)**.** As shown in Example 4, $Sig(x, ABCD) = \langle (1, 4, 5, 7), AC, CD \rangle$. Thus, x is



(a) Subspaces determined by group (x, ABCD)

(b) Subspaces where x is in the subspace skyline

Figure 2: The subspaces where object x in Table 1 is in the skyline.

in the skyline of subspaces inclusively bordered by $ABCD$, AC and CD, as shown in Figure 2(a). This also explains why we opt for the representation of signature. ∎

The signature of skyline group $(x, ABCD)$ explains why and in which subspaces x is in the skyline without any accompanying coincident objects. x coincides with y and z in some subspaces and thus may jointly be in some subspace skylines. This is captured by the corresponding skyline groups.

Theorem 2 (Semantics)**.** *An object u is in the skyline of subspace $\mathcal{C}$ if and only if there exists a skyline group $(G, \mathcal{B})$ and its decisive subspace $\mathcal{C}'$ such that $u \in G$ and $\mathcal{C}' \subseteq \mathcal{C} \subseteq \mathcal{B}$.*

Proof sketch. (Direction if). Following Theorem 1, $G_{\mathcal{C}}$ is in the subspace skyline. Since $u \in G$, $u_{\mathcal{C}}$ is also in the subspace skyline.

(Direction only-if). Consider the group of objects $\mathcal{O}(u, \mathcal{C})$. All objects in the group are in the subspace skyline of $\mathcal{C}$ since they share the same values as u on dimensions in $\mathcal{C}$. Following Lemma 1, $(\mathcal{O}(u, \mathcal{C}), \mathcal{I}(\mathcal{O}(u, \mathcal{C})))$ is a maximal c-group. Furthermore, it can be shown that the group must be in the skyline of subspace $\mathcal{I}(\mathcal{O}(u, \mathcal{C}))$. Thus, the group is a skyline group. We note that subspace $\mathcal{C}$ satisfies conditions (1) and (2) of Definition 3. Thus, if $\mathcal{C}$ is minimal, then $\mathcal{C}$ itself is decisive, i.e., $\mathcal{C}' = \mathcal{C}$. Otherwise, there must exists a $\mathcal{C}' \subset \mathcal{C}$ that $\mathcal{C}'$ is decisive. ∎

2.4 Answering Skyline Membership Queries

Theorem 2 comes with a generic framework of answering *skyline membership queries*: given an object or a group of objects, determine the subspaces where the object(s) are in the subspace skylines.

The framework is simple. As preprocessing, we materialize the set of skyline groups and their signatures, for which the algorithm will be given in Section 4. Then, *instead of searching all possible subspaces, we only need to check the skyline groups in which the object is a member.* This is effective since only the signatures of the skyline groups are needed. Moreover, the skyline groups can be indexed by their signatures to speed up the search. The optimization of the algorithmic framework is an interesting topic for future study, but is beyond the scope of this paper.

Example 6 (Semantics – continued)**.** Continued from Example 5, x is a member of skyline group (xy, AD), which has decisive subspaces A and D. Thus, x is also in the subspace skylines of A, D and AD. Similarly, as a member of group (xz, C), x is in the subspace skyline of C. The complete set of subspaces where x is in the skyline is shown in Figure 2(b). ∎

3 Subspace Skyline Analysis

The notion of skyline groups naturally leads us to explore skylines in subspaces. When skylines in all subspaces are considered, it is imperative to ask: *How are the subspace skylines formed and what is the relationship among them?*

3.1 Intuition

We try to decipher some elegant structures embedded in the subspace skylines.

Skylines in subspaces consist of projections of objects. For a projection that is in the skyline of a subspace, the set of objects that share the same projection form a c-group. By the c-group containment relationship, the projections in subspace skylines form a lattice called the *skyline projection lattice* (Theorem 3 in Section 3.2), which is a concise structure.

The projection lattice may contain redundant information. The critical point here is that some projections in skylines of different subspaces may be made by the same maximal group of objects. Conceptually, a skyline group is a maximal group of objects that coincide in some subspaces and whose projections are also in the subspace skyline. Therefore, we can use skyline groups to derive a concise representation. The lattice of skyline groups is called the *skyline group lattice* and is a quotient lattice of the skyline projection lattice (Theorem 4 in Section 3.2).

Manipulating groups of objects all the time is still inconvenient. Ideally, we would like to select some representatives for the skyline groups. Fortunately, this is achievable since each skyline group must contain at least one object that is in the skyline of the full space. This indicates that the full space skyline casts the contours of skylines in subspaces.

3.2 Skyline Group Lattice

A projection $u_{\mathcal{B}}$ where $u \in S$ is called a *skyline projection* if it is in the skyline of $\mathcal{B}$. We can define a relation $\sqsubseteq$ on the set $\mathcal{P}$ of all skyline projections: for $p, q \in \mathcal{P}$ that are in the subspace skylines of $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively, $p \sqsubseteq q$ if $\mathcal{B}_1 \supseteq \mathcal{B}_2$ and $p_{\mathcal{B}_2} = q$.

Theorem 3 (Skyline projection lattice)**.** *Let $\mathcal{P}$ be the set of all skyline projections with respect to a set of objects S. $(\mathcal{P}, \sqsubseteq)$ is a complete lattice if $(*, *, \ldots, *)$ and $\emptyset$ are treated as the two trivial skyline projections for the unit element and the zero element, respectively.*

Proof sketch. Obviously, $\sqsubseteq$ is a partial order on $\mathcal{P}$. We also note that $\emptyset$ is the projection of any objects in subspace $\emptyset$ (the trivial subspace), and $(*, *, \ldots, *)$ is the projection of an empty set of objects on all dimensions. They are trivial and just technically make up the lattice. The completeness of the lattice follows from the fact that the number of skyline projections is limited. ∎

To characterize that multiple skyline projections may be made by one maximal group of objects, we define an equivalence among skyline projections as follows. For any projection p in subspace $\mathcal{B}$, define the *pre-image* of p as the set of objects that have p as the projection in $\mathcal{B}$, denoted by $pre(p) = \{u | u \in S, u_{\mathcal{B}} = p\}$. For two skyline projections p and q in subspaces $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively, they are equivalent (in terms of being generated by the same group of objects), denoted by $p \sim q$, provided $pre(p) = pre(q)$.

Theorem 4 (Skyline group lattice)**.** *Let $\mathcal{SG}$ be the set of all skyline groups. $(\mathcal{SG}, \sqsubseteq)$ forms a complete lattice where $\sqsubseteq$ is on the projections in the groups. Moreover, $(\mathcal{SG}, \sqsubseteq) = (\mathcal{P}, \subseteq)/\sim$.*

Proof sketch. The quotient lattice claim follows from the fact that a skyline group also expands to include all possible dimensions where the objects share the same projections. Details are omitted due to limited space. ∎

Theorem 4 shows that skyline groups capture skyline projections in subspace skylines effectively, and the signatures of skyline groups serve as the summarization. Immediately, we know that the number of skyline groups is at most the number of skyline projections.

Practically, is the summarization using skyline groups meaningful? In practice, data is more or less correlated. Thus, objects may share values in some dimensions and form groups. In addition to capturing the semantics of skyline objects, skyline groups also summarize data records collapsing in some subspaces and appearing in some subspace skylines.

3.3 Skyline Groups and Skyline Objects

Although skyline groups provide a succinct summarization of the skylines in various subspaces, it still can be inconvenient and costly to manage all group members if many objects exist in a data set. *Can we select some representative objects from the skyline groups?*

Encouragingly, we observe that each skyline group contains at least one skyline object in the full space.

Theorem 5 (Skyline object)**.** *For any skyline group $(G, \mathcal{B})$, there exists at least one object $u \in G$ such that u is in the skyline of full space $\mathcal{D}$.*

Proof. Let u be an object in G such that, in the full space $\mathcal{D}$, u is not dominated by any other objects in G. Such an object exists provided $G \neq \emptyset$. We show that u is a skyline object in $\mathcal{D}$ with respect to the set of all objects S. Otherwise, if u is dominated by v, two cases may arise. First, $u_{\mathcal{B}} = v_{\mathcal{B}}$, then $v \in G$ and it contradicts the assumption that u is not dominated by any other objects in G. Second, if there exists a dimension $D \in \mathcal{B}$ that $u.D > v.D$, then given u is dominated by v in the full space, $u_{\mathcal{B}}$ is dominated by $v_{\mathcal{B}}$. That leads to a contradiction to the assumption that $u_{\mathcal{B}}$ is in the subspace skyline. ∎

Theorem 5 indicates that the skyline objects in the full space play critical roles in the construction of subspace skylines – their projections are sufficient to represent the "*contour*" of the skyline, i.e., the dimension values of the projections in the subspace skyline. In other words, *an object that is not in the skyline of full space can be in the skyline of some subspace only if it collapses to some full space skyline object(s).*

For a data set S, we can obtain the set SK of skyline objects in the full space. A projection is in the skyline of subspace $\mathcal{B}$ in S if and only if it is also in the skyline of the same subspace in SK. Moreover, we can construct skyline group lattices $\mathcal{SG}_S$ and $\mathcal{SG}_{SK}$ on data sets S and SK, respectively. We can show that $\mathcal{SG}_{SK}$ is a quotient lattice of $\mathcal{SG}_S$. Limited by space, we omit the details here.

On the other hand, if we are only concerned with the projections in the subspaces skyline, only the skyline objects in the full space are necessary for the analysis. In such a case, we do not need to manipulate all objects. This potentially leads to a significant reduction in the computational cost.

Theorem 5 immediately has two practically useful applications. First, it gives rise to efficient algorithms for subspace skyline computation, which will be discussed in Section 4. Second, it can also lead to a novel OLAP style analysis of subspace skylines, which will be showcased in Section 3.4.

We would like to point out that the other direction of Theorem 5 does not hold. Generally, a maximal c-group that is not a skyline group still may have a skyline object in the full space as a member. For example, in Figure 1, the group (ae, Y) is a maximal c-group and a is a skyline object in the full space (X, Y), but the group itself is not a skyline group.

3.4 OLAP Analysis on Skylines

Since the skyline groups form a complete lattice, it is natural to introduce the multidimensional roll-up and drill-down analysis on skyline groups.

Example 7 (OLAP analysis)**.** Figure 3 shows the skyline group lattice in our running example (Table 1). For each node in the lattice, the projection, the skyline objects as representatives, and the decisive subspaces are shown.

By browsing Figure 3, the following structural information about the subspace skylines can be presented.

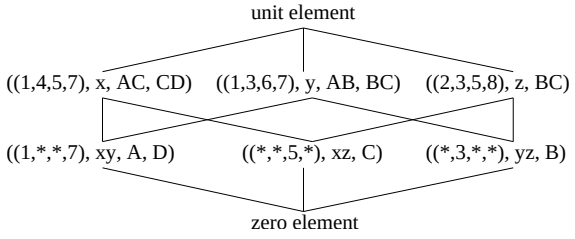


Figure 3: Skyline group lattice for Table 1.

- Subspace skylines. The information is recorded in the signatures.
- Relationships between skylines in subspaces. For example, from the figure, we know that an object is in the subspace skyline of C if it has value 5 on C. There are two ways to further qualify the object as a skyline object in the full space: either having value 3 on B (i.e., object z), or having value 1 on A or 7 on D (i.e., object x). The latter two values ($A = 1$ and $D = 7$) always come together.
- Closure information. From the figure, we can learn that it is impossible to have an object in the subspace skyline of BCD, but not in the subspace skyline of $ABCD$. Although a naïve method to derive this information has to check all objects in the data set, we derive this information from only the skyline groups. ■

In practice, why are such roll-up and drill-down operations useful? Suppose all objects in our running examples are stocks. When a user examines subspace C, she finds that both x and z are subspace skyline objects. This is interesting to her, but she would like to find out further in what other subspaces x is also good and is better than z. Then, she finds AC and CD and their super-spaces through a roll-up.

Clearly, the online roll-up and drill-down analysis is not available in the traditional skyline analysis.

4 Subspace Skyline Computation

Given a data set, the problem of *subspace skyline computation* is to compute, for each non-empty subspace, the set of objects that are in the skyline of the subspace. At the same time, we also want to compute the complete set of skyline groups and their signatures as the summarization of the skylines.

4.1 Finding Skyline by Sorting

A lexicographic order can be defined on the set of objects S. For any objects $u, v \in S$. $u \prec v$ if there exists an i_0 ($1 \leq i_0 \leq n$) such that $u.D_i = v.D_i$ for ($1 \leq i < i_0$) and $u.D_{i_0} < v.D_{i_0}$. $u \preceq v$ if $u \prec v$ or $u = v$. Apparently, the lexicographic order $\preceq$ is a total order.

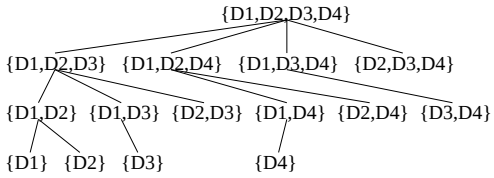


Figure 4: A top-down subspace enumeration tree.

As shown in [3], skyline objects in the full space $\mathcal{D}$ can be found in two steps, as illustrated in the following example.

Example 8 (Skyline computation by sorting)**.** Let us compute skyline objects in space (X, Y) for the objects in Figure 1. In the first step, we sort all objects in the lexicographic order. The sorted list is $a(1, 3)$, $d(1, 4)$, $b(2, 2)$, $e(3, 3)$, $c(4, 1)$.

The second step is as follows. We initiate the set of skyline objects as empty. Then, we scan the sorted list once. For each object u in the list, we compare u against the current set of skyline objects. If u is not dominated, then u is a skyline object and is inserted into the set.

For example, since $a(1, 3)$ is the first one in the sorted list, it is not dominated and is inserted into the set. The next object, $d(1, 4)$, is dominated by $a(1, 3)$ in the current set of skyline objects, and thus is discarded. The third object, $b(2, 2)$, is compared with $a(1, 3)$, and is not dominated. Thus, b is also inserted into the set as a new skyline object. $e(3, 3)$ is discarded since it is dominated by $a(1, 3)$. Last, $c(4, 1)$ is inserted in to the set since it is not dominated by either a or b. The set of skyline objects $\{a, b, c\}$ is returned. ■

4.2 Top-down Subspace Enumeration Tree

In order to search skylines of all subspaces thoroughly, we search subspaces in a depth-first manner. The complete set of subspaces can be enumerated systematically using a (top-down) subspace enumeration tree. For example, Figure 4 shows a tree enumerating subspaces of space (D_1, D_2, D_3, D_4).

A top-down subspace enumeration tree differs from a conventional set enumeration tree [14] in the way of enumeration. In a conventional set enumeration tree, search starts from the empty set, and each child adds a new element to the parent set. It is bottom-up. In the top-down subspace enumeration tree here, we start from the full space, and each child explores a proper subspace with one dimension less. The reason for this arrangement is that the search from super-spaces to subspaces enables us to recursively use Theorem 5 to prune the set of objects under consideration. This point will become clear in Section 4.3.

4.3 Algorithm *Skyey*

Algorithm *Skyey* takes a set of objects S in space $\mathcal{D}$ as input, and returns the set of skyline groups (in the

form of signatures) and, for each non-empty subspace, a list of objects that are in the corresponding subspace skyline. We first describe the algorithm. An example (Example 10) will follow.

4.3.1 Finding Skyline Objects in Full Space

In the first step, *Skyey* sorts all objects in the lexicographic order and finds the set of skyline objects in the full space, as illustrated in Example 8. The list of skyline objects in the full space can be output. Every distinct skyline object forms a skyline group. In other words, if two objects u and v have exactly the same value in all dimensions of $\mathcal{D}$, and both of them are in the skyline, then they share the same skyline group.

After the set of skyline objects in the full space is found, *Skyey* makes one scan of all objects that are not in the skyline of full space. Each non-skyline object is compared with the skyline objects. For each non-skyline object u and skyline object v, if the maximal set of dimensions on which u and v share common values (i.e., the set of dimensions $\mathcal{I}(uv)$) is non-empty, then a tag of $(u, \mathcal{I}(uv))$ is attached to v. The tag means u and v coincide in subspace $\mathcal{I}(uv)$.

After this step, all non-skyline objects can be discarded. We do not need to access them anymore in the rest of the algorithm. The rest of *Skyey* searches the subspaces by a depth-first traversal of the subspace enumeration tree (Section 4.2).

4.3.2 Efficient Local Sorting

Suppose the current node corresponds to a subspace $\mathcal{B}$. To identify the objects in the subspace skyline of $\mathcal{B}$, a naïve method is to sort the skyline objects in the parent node. However, repeatedly sorting objects for different subspaces can be expensive if there are many objects and many subspaces. Interestingly, we can reuse the sorted list in the parent node, which can reduce the sorting cost substantially.

Example 9 (Local sorting)**.** Suppose the skyline objects in the full space (D_1, D_2, D_3, D_4) is evaluated by sorting. In order to find the objects in the subspace skyline of (D_1, D_2, D_3) (i.e., the first child of the root node in Figure 4, we do not need to sort the objects – they are already sorted since a sorted list by (D_1, D_2, D_3, D_4) is also a sorted list by (D_1, D_2, D_3). In fact, we do not need to do any sorting when we search the subspaces in the leftmost branch of the subspace enumeration tree.

Now, let us consider the second leftmost leaf node in Figure 4, D_2. The objects are sorted by (D_1, D_2) in the parent node. Instead of a complete sorting, merge sort can serve the same purpose and save. The idea is as follows.

The sorted list by (D_1, D_2) can be regarded as divided into groups according to D_1. Within each group, objects are sorted by D_2. We only need to merge the groups according to D_2. ∎

4.3.3 Finding Objects in Subspace Skylines

Once the objects are sorted in the subspace, then the objects in the subspace skyline can be identified. Please note that we only sort the list of skyline objects in the parent node. To make the list of objects in the subspace skyline of the current node complete and the decisive subspace accurate, we need to find the objects that are not in the skyline of the parent node, but are in the skyline of the current subspace. This can be achieved by examining the information recorded in the tags using the following two rules.

- *Identifying objects that are not in the skyline of the parent node, but are in the skyline of the current subspace.* For an object v that is in the skyline of the current subspace $\mathcal{B}$, we check the tags attached to v. For any tag $(u, \mathcal{C})$ such that $\mathcal{B} \subseteq \mathcal{C}$, u should also be output as an object in the skyline of the current subspace, and is put in the same skyline group where v is in. The reason is that u share the same values as v on all dimensions in $\mathcal{B}$.

- *Identifying decisive subspaces.* For each skyline group G, we check whether the number of objects in the skyline of the current subspace is the same as the number of objects in the parent node subspace. If the number of objects changes, that means the group changes, i.e., $\mathcal{O}(G, \mathcal{B}) \supset \mathcal{O}(G, \mathcal{B}')$. Then, we add the parent subspace as a temporary decisive subspace to group G, and create a new group for $G' = \mathcal{O}(G, \mathcal{B})$, and $G_{\mathcal{B}}$ is recorded in the signature of the new group. If the group already has a temporary decisive subspace that is a super-space of the newly inserted one, then the super-space should be removed.

The depth-first search proceeds recursively. According to Theorem 5, only objects that are in the skyline of the current subspace should be passed to the children. Tags should be created for those objects that are in the skyline of the parent subspace but not in the skyline of the current subspace.

Example 10 (Algorithm *Skyey*)**.** We demonstrate the algorithm *Skyey* using our running example (Table 1).

As the first step, we sort all three objects in the lexicographic order and identify all three objects in the skyline of the full space (A, B, C, D). We create a skyline group for each object.

Then, we go to subspace (A, B, C). We do not need to sort the objects since the sorted list in the root node can be reused. Again, in this subspace, all objects are in the subspace skyline, and the groups do not change.

We further go to subspace (A, B). Again, we reuse the sorted list. In this subspace, x and z are dominated by y. Thus, ABC should be added to the

Input: A set of objects S in space $\mathcal{D}$;
Output: (1) the set of skyline groups, and (2) for each subspace, the set of objects in the subspace skyline;
Method:
Call *Skyey*(S, $\mathcal{D}$);

Function *Skyey*(S', $\mathcal{D}'$)
1: Sort S' in lexicographic order, try to reuse the existing sorted list as shown in Section 4.3.2;
2: identify objects in the subspace skyline of $\mathcal{D}'$;
3: for each skyline group in the parent node, check whether a temporary decisive subspace should be added; if necessary, create new groups and tags, remove minimal temporary decisive subspaces (Section 4.3.3);
4: let S'' be the set of subspace skyline objects in the current subspace;
5: for each $(|\mathcal{D}'|-1)$-d subspace $\mathcal{D}''$ call *Skyey*(S'', $\mathcal{D}''$)
6: return;

Figure 5: The *Skyey* algorithm.

groups of x and z, respectively, as temporary decisive subspace. Two tags, (x, AD) and (z, B), should be created and attached to y. Later, temporary decisive subspace ABC for group x will be removed when another decisive subspace AC is inserted into the group.

We turn to subspace A, where y is still in the subspace skyline. However, the group needs to be expanded, since the tag (x, A) indicates x is also in the subspace skyline. Thus, the parent subspace, AB should be added to group y as a temporary decisive subspace. A new group, xy is created. Since x and y share values on dimensions A and D, the signature of group xy should include dimension D as well, i.e., $(xy, (1, *, *, 7))$.

The other subspaces are searched recursively. Limited by space, we omit the details here. ∎

The algorithm is summarized in Figure 5. The correctness and completeness of the algorithm follows from the above discussion. Limited by space, we omit the formal results here.

Comparing to a brute-force search, algorithm *Skyey* has two major advantages. First, by Theorem 5, *Skyey* recursively reduces the set of skyline objects that need to be searched – only those objects in the skyline of the current node will be passed to the children. Often, only a small subset of objects in a set is in the skyline. The recursive reduction is effective in practice. Second, the adaptive local sorting can reuse the sorted list of the parent node to avoid sorting at all in some nodes, and also use merge sort which is practically more efficient than sorting from scratch.

Since *Skyey* needs to output the skyline of each nonempty subspace, it has to search every subspace once. In fact, if a user is interested in only the skyline groups but not the detailed lists of objects in the skyline of every subspace, the search can be further sped up. Limited by space, we omit the details here.

5 Related Work

To the best of our knowledge, this is the first study on the semantics and structure of subspace skylines. In the following, we review some related work on skyline query processing, and formal concept analysis and its applications in multidimensional data analysis.

Numerous algorithms have been proposed for skyline retrieval and other related problems (e.g., multiobjective optimization [15], maximum vectors [8, 16, 11], etc.). In the database context, Borzsonyi et al. [2] develop the solutions based on divide-and-conquer (DC) and block nested loops (BNL). Specifically, DC divides the dataset into several partitions that can fit in memory. The skylines in all partitions are computed separately using a main-memory algorithm, and then merged to produce the final skyline. BNL essentially compares each tuple in the database with all the other records, and outputs the tuple only if it is not dominated in any case. The sort-first-skyline (SFS) [3] sorts the input data according to a (monotone) preference function, after which the skyline can be found in another pass over the sorted list. Tan et al. [17] propose a solution that deploys the highly CPU-efficient bit-operations (e.g., calculating the AND/OR of two binary vectors) by computing the skyline from some bitmaps capturing the original dataset. The authors also provide another method based on some clever observations on the relationships between the skyline and the minimum coordinates of individual points. Kossmann et al. [7] present an algorithm that finds the skyline with numerous nearest neighbor searches. An improved approach following this idea appears in [12]. Balke et al. [1] study skyline computation in web information systems, applying the "threshold" algorithm of [4]. Recently, Lin et al. [10] consider the skyline maintenance over data streams.

The maximal c-group lattice is levered by the formal concept analysis [5]. However, very different from previous studies on database and data mining studies using formal concept analysis, such as [13, 9], we are interested in a small part of the lattice – only the skyline groups. Moreover, the challenge is how to compute the skyline groups and the objects in the subspace skylines. These issues are far beyond the traditional studies on formal concept analysis.

There have been some recent studies on finding the determinant factors in multidimensional subspaces for some critical data. Two typical examples are [6, 9]. In [6], Knorr and Ng propose an approach to find the minimal sets of factors that explain the distance-based outliers. In [9], the quotient cube lattice is developed to identify groups of aggregates that share the same set of base tuples in a data warehouse and thus the semantics of the aggregates can be explained and summarized concisely. While the philosophy in this research

Player	GP	PTS	REB	AST	Decisive subspaces
Wilt Chamberlain (1960)	79	3033	2149	148	(REB)
Wilt Chamberlain (1961)	80	4029	2052	192	(PTS), (GP, REB), (REB, AST)
Chuck Williams (1973)	90	1113	250	557	(GP)
John Stockton (1990)	82	1413	237	1164	(AST)
Michael Jordan (1988)	81	2633	652	650	(PTS, REB, AST), (GP, PTS, AST)
Gary Payton (2001)	82	1815	396	737	(GP, PTS, REB, AST)

Table 2: Some skyline players and the corresponding decisive subspaces.

Dimensionality	# of players
1	4
2	41
3	102
4	67

Table 3: Number of skyline players in subspaces with different dimensionality.

share some similarity to those studies, the technical problems and the approaches are essentially different.

Simultaneous to our study, Yuan et al. [18] study the problem of computing the skylines in all subspaces and develop efficient algorithms. Both [18] and this study investigate the skylines in various subspaces. However, there are two critical differences. First, the methods in [18] search the skyline in every subspace and do not explore the structure of the skylines. In this paper, we study the structure of skylines in subspaces, and use the concepts of skyline groups and decisive subspaces to capture the semantics of subspace skylines. Second, the algorithms in [18] and algorithm *Skyey* in this paper compute skylines for every subspace. In addition, algorithm *Skyey* computes skyline groups and their signatures. Interestingly, both studies suggest that a top-down depth-first search framework may favor efficient computation. [18] further develops a novel data structure, skylist, to store skyline objects in different subspaces compactly.

6 Experimental Results

We conducted an empirical study of our method using both a real data set and the benchmark synthetic data sets. We evaluated the meaningfulness of skyline groups and their decisive subspaces, as well as the efficiency and the scalability of our approach to subspace skyline computation.

All algorithms were implemented using Microsoft Visual C++ V6.0. Experiments were conducted on a PC with an Intel Pentium 4 1.6 GHz CPU, 512 M main memory and a 40 G hard disk, running the Microsoft Windows XP Professional Edition operating system.

6.1 Results on Real Data Set *Great NBA Players' Statistics*

We downloaded from the NBA official website (www.nba.com) the Great NBA Players' technical statistics from 1960 to 2001. Totally there are $17,266$ records. Each record is the statistics of a player in a season. We selected four attributes: the number of games played (GP), total points (PTS), total rebounds (REB) and total assists (AST). In this data set, the larger the attribute values, the better. That is, player A dominates player B if A's attribute values are not less than B's, and A has at least one attribute better than B.

Finding the skyline in this players' statistics data set makes excellent sense in practice. People are often interested in finding the skyline players – players who have some outstanding merits that are not dominated by some other players. Moreover, finding the semantics of skyline in this application is of great interest – we not only want to know who are the great players, but also want to know exactly on which combinations of factors a player is dominating the other players.

The knowledge of subspace skylines has immediate applications. For example, if a coach wants to find a player good at total points and total rebounds, he should look at the skyline players in the subspace (PTS, REB), instead of all skyline players.

In this data set, we found 67 skyline records in the full 4-d space. The total number of corresponding decisive subspaces is 146, and the average dimensionality of the decisive subspaces is 2.21. We list some skyline players and their decisive subspaces in Table 2.

The first four records are in the skyline since each of them takes the maximum value in one dimension. Interestingly, Wilt Chamberlain's performance in 1961 was also outstanding in some combinations of attributes. Michael Jordan's performance in 1988 was not exceptional in terms of any single attribute. However, it is in the skyline once attribute combinations (PTS, REB, AST) or (GP, PTS, AST) are considered. Gary Payton's performance in 2001 is in the skyline only if all the attributes are considered.

Clearly the decisive subspaces provide more insightful information than just the list of skyline players in the full space.

We found skyline records in all non-empty subspaces. Some of them may not be skyline records in the full space. The numbers of subspace skyline groups are listed in Table 3. These numbers can be explained by the different number of subspaces associated with the given dimensionality and by the fact that the number

of skyline records increases with increasing dimensionality.

We also counted the total number of subspaces where a record is in the skyline. This is an interesting measure. Intuitively, if a player is in the skylines of more subspaces, he has a better overall capability in terms of combinations of attributes. We found that, in addition to dominating all others in total points (PTS), Wilt Chamberlain's performance in 1961 has the highest number, 13, of subspaces where it is in the skyline. On the other hand, although John Stockton's performance in 1990 dominates all others in total assists (AST), which is decisive, it is only in the skyline of 5 subspaces.

From the preliminary analysis on the real data set, we obtained the interesting and meaningful observations that cannot be derived from the traditional skyline computation. This demonstrates the meaningfulness of the proposed subspace skyline analysis.

6.2 Results on Synthetic Data Sets

Using the data generator provided by the authors of [2], we generated three types of data sets as described in [2]:

- *Independent data sets* where the attribute values of the generated records are uniformly distributed;
- *Correlated data sets* where if a record is good in one dimension, likely it is also good in other dimensions; and
- *Anti-correlated data sets* where if a record is good in one dimension, it is unlikely to be good in other dimensions.

For the details of the data generator, please refer to [2]. For each type of data distribution, we generated data sets with different sizes (from $100,000$ to $1,000,000$ tuples) and with dimensionality varying from 2 to 6.

For the purpose of comparison, we implemented a naïve bottom-up algorithm as a baseline method to compute the complete set of skyline groups and the skylines in every subspace. The basic idea is to compute skylines from the 1-d subspaces to the full space. A depth-first search using the traditional set enumeration tree [14] is used. For example, given a dataset of attributes (D_1, D_2, D_3, D_4), we first compute the skyline in a single dimension subspace D_1 by sorting all objects by dimension D_1. After the sorting, by one scan we can immediate determine the skyline objects in the subspace D_1. To compute the skyline objects in subspace (D_1, D_2), we need to sort the objects by dimensions (D_1, D_2). We apply the local sorting technique to reduce the cost of sorting. We also capture the skyline groups by monitoring objects splitting into different groups as new dimensions are added in. Since the search is bottom-up, once a new group is formed, the decisive subspace is caught.

Dimensionality	# of skyline groups		
	Independent	Correlated	Anti-correlated
1	5	5	12
2	50	42	300
3	60	16	1562
4	150	10	2831
5	378	9	10652
6	1320	36	22320

Table 4: Number of skyline objects in subspaces on synthetic data sets.

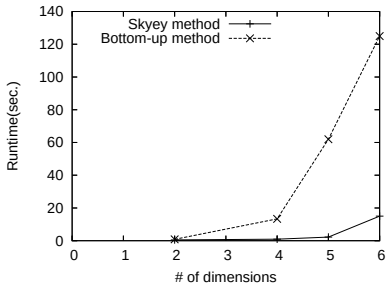


Figure 9: Runtime vs. Dimensions

We evaluated the scalability of algorithm *Skyey* and the bottom-up method with respect to the number of tuples in the data sets. The dimensionality was fixed to 4. The results on the three types of data sets are shown in Figures 6, 7 and 8, respectively. Clearly, both methods are scalable with respect to the size of data sets, but *Skyey* is far more efficient than the bottom-up method. Among the three data sets, the computation on the correlated data sets is the fastest, and the computation on the anti-correlated data sets is the slowest.

To further understand the effect of different distributions on the number of skyline groups, in Table 4, we list the number of skyline objects in subspaces found on synthetic data sets with 6 dimensions and $100,000$ tuples.

As can be seen, on the correlated data sets, the number of skyline objects in subspaces is always the smallest among the three types of data sets, while the number of skyline objects on the anti-correlated data sets is always substantially larger than the other two types. This clearly explains the difference in runtime.

We also tested the scalability of *Skyey* and the bottom-up method with respect to the dimensionality. Limited by space, we only show in Figure 9 the results on the data sets with independent data distribution. The size of the data sets was fixed to $100,000$ records.

We observed that *Skyey* is much more scalable than the bottom-up method. In our experiments, *Skyey* is 5 times faster than the bottom-up method when the dimensionality is 6. The speed-up factor is increasing

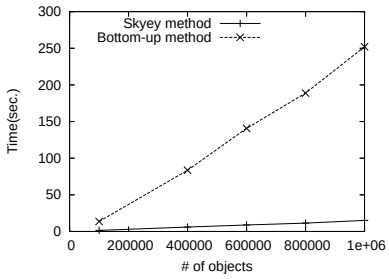


Figure 6: Runtime on Independent Data sets

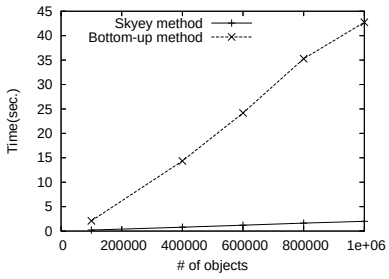


Figure 7: Runtime on Correlated Data sets

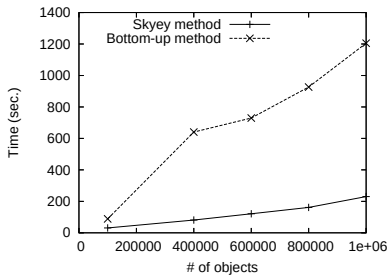


Figure 8: Runtime on Anti-Correlated Data sets

with increasing dimensionality.

While *Skyey* is quite efficient when the dimensionality is not very high, we note that neither method is linearly scalable with respect to dimensionality. This observation is consistent with the results in previous studies (e.g., [12]), which showed that the dimensionality curse is still a grand challenge for skyline computation. The same applies to subspace skyline computation. How to conduct efficient subspace skyline computation and analysis on high dimensional data (e.g., with dimensionality over 100) is still an open problem.

7 Conclusions

In this paper, we answered the questions about semantics of skyline objects by introducing the novel notions of skyline groups and decisive subspaces. We proposed the problem of subspace skyline analysis and computation. On the subspace skyline analysis side, a novel roll-up and drill-down analysis of skylines in various subspaces was introduced. On the subspace skyline computation side, an efficient algorithm *Skyey* was developed. A performance study using both real and synthetic data sets was conducted to verify the meaningfulness and the efficiency of our approach. The experimental results strongly suggest that the semantics of skyline objects and subspace skyline analysis are highly meaningful in practice, and algorithm *Skyey* is efficient and scalable.

To the best of our knowledge, this is the first study on the semantics of skylines and the subspace skyline analysis. As the future work, it would be interesting to explore how to integrate the algorithmic contributions in [18] and this paper to develop more efficient methods for skyline group computation.

Acknowledgement. We are grateful to Dr. Jiawei Han for his valuable comments on an early version of this paper, and to Dr. Donald Kossmann for providing us the synthetic data generator. We thank Dr. Xuemin Lin for sending us the submission version of [18].

References

[1] W-T. Balke, U. Güntzer, and J X. Zheng. Efficient distributed skylining for web information systems. In *EDBT*, 2004.

[2] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, 2001.

[3] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with pre-sorting. In *ICDE*, 2003.

[4] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[5] B. Ganter and R. Wille. *Formal Concept Analysis – Mathematical Foundations*. Springer, 1996.

[6] Edwin M. Knorr and Raymond T. Ng. Finding intensional knowledge of distance-based outliers. In *VLDB*, 1999.

[7] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: an online algorithm for skyline queries. In *VLDB*, 2002.

[8] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476, 1975.

[9] L. Lakshmanan, J. Pei, and J. Han. Quotient cube: How to summarize the semantics of a data cube. In *VLDB*, 2002.

[10] X. Lin, Y. Yuan, W. Wang, and H. Lu. Stabbing the sky: Efficient skyline computation over sliding windows. In *ICDE*, 2005.

[11] J. Matousek. Computing dominances in e^n (short communication). *Inf. Process. Lett.*, 38(5):277–278, 1991.

[12] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, 2003.

[13] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, 1999.

[14] R. Rymon. Search through systematic set enumeration. In *Proc. 1992 Int. Conf. Principle of Knowledge Representation and Reasoning (KR'92)*, pages 539–550, Cambridge, MA, 1992.

[15] R. Steuer. *Multiple Criteria Optimization*. John Wiley, New York, 1986.

[16] Ivan Stojmenovic and Masahiro Miyakawa. An optimal parallel algorithm for solving the maximal elements problem in the plane. *Parallel Computing*, 7(2):249–251, 1988.

[17] K. Tan, P. Eng, and B. Ooi. Efficient progressive skyline computation. In *VLDB'01*, 2001.

[18] Y. Yuan, X. LIN, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang. Efficient Computation of the Skyline Cube. In *VLDB'05*.