# Computing Compressed Multidimensional Skyline Cubes Efficiently *

Jian Pei[1] Ada Wai-Chee Fu[2] Xuemin Lin[3] Haixun Wang[4]

[1] Simon Fraser University, Canada, jpei@cs.sfu.ca
[2] Chinese University of Hong Kong, adafu@cse.cuhk.edu.hk
[3] The University of New South Wales & NICTA, Australia, lxue@cse.unsw.edu.au
[4] IBM T. J. Watson Research Center, USA, haixun@us.ibm.com

## Abstract

*Recently, the skyline computation and analysis have been extended from one single full space to multidimensional subspaces, which can lead to valuable insights in some applications. Particularly, compressed skyline cubes in the form of skyline groups and their decisive subspaces provide a succinct summarization and compression of multidimensional subspace skylines. However, computing skyline cubes remains a challenging task since the existing methods have to search an exponential number of non-empty subspaces for subspace skylines. In this paper, we propose a novel and efficient method,* Stellar*, which exploits an interesting skyline group lattice on a small subset of objects which are in the skyline of the full space. We show that this skyline group lattice is easy to compute and can be extended to the skyline group lattice on all objects. After computing the skyline in the full space,* Stellar *only needs to enumerate skyline groups and their decisive subspaces using the full space skyline objects. Avoiding searching for skylines in an exponential number of subspaces improves the efficiency and the scalability of subspace skyline computation substantially in practice. An extensive performance study verifies the merits of our new method.*

## 1 Introduction

The skyline operator is important for many multi-criteria decision making applications. For example, to search for flight tickets from Vancouver, Canada to Istanbul, Turkey, a customer prefers low price and short travel time. Route $A$ *dominates* route $B$ (or, $A$ is a better choice than $B$ in the context of this example) if $A.price \leq B.price$, $A.traveltime \leq B.traveltime$, and at least one inequality holds. Those routes not dominated by any others form the skyline. Given a set of multidimensional objects, a *skyline query* returns the complete set of skyline objects.

In many applications, a user may be interested in not only the skyline in the full space (i.e., with respect to all available attributes), but also the skylines in various subspaces. In our flight ticket selection example, a user may be also interested in the number of stops, and thus may want to compare the skylines in subspaces $(price, traveltime)$, $(price, stops)$ and $(price, traveltime, stops)$. Generally, a skyline route with respect to a set of attributes may not be a skyline route any more if some attributes are added or removed.

In [10, 15], the full space skyline computation is extended to subspace skyline computation. Particularly, in [10], we extended the skyline analysis in full space to multidimensional subspace skyline analysis, and proposed a general framework. An object is in the skyline of a subspace if it is not dominated by any other objects in the subspace. Roughly speaking, a group of objects form a *skyline group* if they share common values in a subspace and the shared values are in the skyline of that subspace. A *decisive subspace* of a skyline group is a minimal combination of attributes that qualifies the group of objects exclusively in the skyline of some subspaces. The model using skyline groups and the corresponding decisive subspaces can capture subspace skyline objects and the factors contributing to their skyline membership.

Given a data set, by computing the compressed skyline cube consisting of all skyline groups and the corresponding decisive subspaces, three types of queries can be answered. First, given any subspace, we can tell the set of skyline objects in the subspace. Second, given any object or a group of objects, we can identify the subspaces where the objects are

*The authors are grateful to the anonymous reviewers for their constructive comments. Jian Pei's research is supported in part by NSERC Discovery Grants, NSERC Collaborative Research and Development Grants, NSF Grant IIS-0308001, and IBM Eclipse Innovation Awards. Ada Fu's research is supported in part by the RGC Earmarked Research Grant of HKSAR CUHK 4120/05E. Xuemin Lin's research is supported in part by Australian Research Council Discovery Grant (DP0666428) and UNSW Faculty Research Grant Program (FRGP, PS08709). 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.

in the skylines. Last, multidimensional analysis on skylines in various subspaces can be conducted. Therefore, skyline groups and their decisive subspaces are a succinct summarization and compression of skylines in all subspaces.

While [10] provides a fundamental framework for multidimensional subspace skyline analysis, how to compute skyline groups and decisive subspaces efficiently has not been addressed sufficiently. In [10], a simple algorithm, *Skyey* is proposed by assembling a data cube algorithm and a sorting-based skyline algorithm. The major idea is that, starting from the full space, all non-empty subspaces are searched systematically in a depth-first manner. While skyline objects in a subspace are identified by sorting objects properly, the sorted lists of objects are shared as much as possible by the skyline computation in multiple subspaces.

The *Skyey* algorithm may not be efficient or scalable if the dimensionality is high, since *it has to search skyline in every non-empty subspace*! When the dimensionality is high, the number of subspaces can be huge.

*Can we compute skyline groups and their decisive subspaces without searching all subspaces?*

In this paper, we study the problem of efficient compressed skyline cube computation by developing a fast algorithm *Stellar* which computes skyline groups and decisive subspaces *without searching all subspaces for skylines*. Particularly, we observe a nice relation between the skyline groups formed by full space skyline objects only and the skyline groups formed by all objects: the former lattice is a quotient lattice of the latter one. Then, we develop an efficient method to compute the full space skyline only and use the skyline to shape multidimensional skyline groups and their decisive subspaces. Our method avoids searching for subspace skylines in all proper subspaces!

The remainder of the paper is organized as follows. In Section 2, we present the problem definition. We review related work in Section 3. In Section 4, we draw the relation between two interesting lattices. In Section 5, we develop a new algorithm *Stellar*. Experimental results are presented in Section 6. The paper is concluded in Section 7.

## 2 Multidimensional Subspace Skylines

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. Moreover, we often write a set as a string and omit the set brackets. For example, object set $\{P_2, P_5\}$ and subspace $(A, C)$ are written as $P_2P_5$ and $AC$, respectively.

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 $i_1 < \cdots < i_{|\mathcal{B}|}$.

For objects $u, v \in S$, $u$ is said to *dominate* $v$ in subspace $\mathcal{B}$ if $u.D \leq v.D$ for any $D \in \mathcal{B}$, and there exists at least one dimension $D_{i_0} \in \mathcal{B}$ such that $u.D_{i_0} < v.D_{i_0}$. Object $u$ is a *skyline object* in $\mathcal{B}$ if $u$ is not dominated by any other objects from $S$ in $\mathcal{B}$. The *skyline in subspace* $\mathcal{B}$ is the complete set of skyline objects in $\mathcal{B}$.

Generally, two objects $u$ and $v$ may have some common values on some dimensions. The concepts of coincident groups and skyline groups capture the object groups sharing common values and skyline memberships in subspaces.

**Definition 1 (c-group and skyline group [10])** In subspace $\mathcal{B}$, a set of objects $G$ forms a *coincident group* (or *c-group* for short) $(G, \mathcal{B})$ if all objects in $G$ have the same projection in $\mathcal{B}$, i.e., $\forall u, v \in G$, $u_\mathcal{B} = v_\mathcal{B}$. We denote the common projection by $G_\mathcal{B}$.

A c-group $(G, \mathcal{B})$ is *maximal* if there exists no other object $v \in (S-G)$ such that $v_\mathcal{B} = G_\mathcal{B}$, and objects in $G$ do not share the same value on any other dimension $D \in (\mathcal{D} - \mathcal{B})$. $\mathcal{B}$ is called the *maximal subspace* for group $G$.

A maximal c-group $(G, \mathcal{B})$ is a *skyline group* if $G_\mathcal{B}$ is in the skyline of subspace $\mathcal{B}$. Clearly, if $(G, \mathcal{B})$ is a skyline group, every object $u \in G$ is also in the skyline of $\mathcal{B}$. ■

Roughly speaking, a decisive subspace for a skyline group is a minimal combination of attributes that enables all objects in the group as subspace skyline objects exclusively.

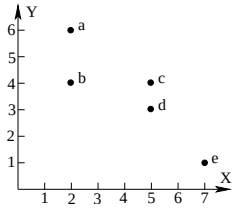
**Definition 2 (Decisive subspace [10])** For skyline group $(G, \mathcal{B})$, a subspace $\mathcal{C} \subseteq \mathcal{B}$ is called *decisive* if (1) $G_\mathcal{C}$ is in the subspace skyline of $\mathcal{C}$; (2) for any object $u \in (S - G)$, $u_\mathcal{C} \neq G_\mathcal{C}$; and (3) there exists no proper subspace $\mathcal{C}' \subset \mathcal{C}$ such that conditions (1) and (2) also hold for $\mathcal{C}'$.

A skyline group can be summarized by the *signature* $Sig(G, \mathcal{B}) = \langle G_\mathcal{B}, \mathcal{C}_1, \ldots, \mathcal{C}_k \rangle$, where $\mathcal{C}_1, \ldots, \mathcal{C}_k$ are all decisive subspaces of the skyline group. ■

As shown in [10], if $\mathcal{C}$ is a decisive subspace for skyline group $(G, \mathcal{B})$, then, all objects in $G$ are also in the skyline of any subspace $\mathcal{A}$ between $\mathcal{B}$ and $\mathcal{C}$, i.e., $\mathcal{C} \subseteq \mathcal{A} \subseteq B$. In other words, skyline groups summarize the skyline membership of groups of objects in various subspaces.

**Example 1 (Concepts)** Consider the set of 5 objects in Figure 1(a). The subspace skylines are shown in Figure 1(b). Although object $d$ is a skyline object in the full space, it is not in any subspace skyline. On the other hand, object $a$ is not in the full space skyline, but it is in the subspace skyline of $X$.

Object $e$ has value 1 in dimension $Y$, which enables $e$ as a skyline object in spaces $Y$ and $XY$. No other object




(a) A set of 5 objects

| Subspace | Skyline |
|---|---|
| $XY$ | $b, d, e$ |
| $X$ | $a, b$ |
| $Y$ | $e$ |

(b) Subspace skylines

**Figure 1. A set of objects in Example 1**

shares the same value on $Y$ with $e$. Thus, $(e, XY)$ is a skyline group with decisive subspace $Y$, which summarizes the skyline membership of $e$ in subspaces $Y$ and $XY$. Object $d$ is in the skyline of space $XY$ but not in any subspaces. Thus, $(d, XY)$ is a skyline group and the decisive subspace is $XY$. Object $b$ is in the skylines of subspaces $XY$ and $X$, and $b.X = 2$ enables its skyline membership. However, in subspace $X$, $b$ has the same projection as $a$. Thus, $(ab, X)$ is a skyline group with decisive subspace $X$, and $(b, XY)$ is another skyline group with decisive subspace $XY$.

Generally, a skyline group may have more than one decisive subspace. We will see an example soon. ■

**Problem Definition**. Given a set of objects $S$ in $n$-dimensional space $\mathcal{D}$, the problem of *compressed skyline cube computation* (or *subspace skyline computation* in [10]) is to compute the complete set of skyline groups and their decisive subspaces. ■

The following result can help in compressed skyline cube computation.

**Theorem 1 (Full space skyline objects [10])** *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 $\mathcal{D}$.* ■

## 3 Related Work

Simultaneous to [10], Yuan et al. [15] studied the problem of computing the skylines in all subspaces and developed efficient algorithms. Different from [10], the algorithms in [15] cannot compute skyline groups and their signatures.

Skyline groups and their decisive subspaces can be regarded as the summarization and compression of subspace skylines. As shown in [10], all subspace skylines can be derived from the complete set of skyline groups and their decisive subspaces. In this paper, we focus on the computation of skyline groups and their signatures instead of only the skylines in subspaces. Directly adopting the algorithms from [15] cannot help since those algorithms also have to compute skylines in all subspaces.

Recently, an alternative approach to subspace skyline analysis was proposed in [13]. Instead of materializing all subspace skylines as in [10, 15], an effective index on the set of objects in question is devised in [13] so that any subspace skyline can be extracted on the fly efficiently. The index can be implemented efficiently using a B+-tree. However, the skyline group and decisive subspace issue is not addressed in [13].

In previous studies, many algorithms have been developed for skyline query answering, such as the divide-and-conquer (DC) and block nested loops (BNL) approaches Borzsonyi et al. [1], the sort-first-skyline (*SFS*) method by Chomicki et al. [2], the method using bit-operations by Tan et al. [12], the nearest neighbor search method by Kossmann et al. [6] (an improvement in [7]), and the integrated method LESS by Godfrey et al. [5]. However, all those methods only consider the skyline in one single space, and do not address the issue of skyline groups and decisive subspaces.

As multidimensional skyline analysis is interesting and useful in a few applications, [10, 15] have been extended in a few interesting ways. For example, Xia and Zhang [14] studied how to incrementally maintain a skyline cube against frequent updates.

There have been also some other recent studies on high dimensional subspace skylines. For example, Chan et al. [4] studied finding top-$k$ frequent skyline points in multidimensional subspaces. Chan et al. [3] considered the $k$-dominance relation and explored $k$-dominant skylines.

## 4 Seed Skyline Groups and Lattices

Theorem 1 is an important hint in computing skyline groups: each skyline group must have a full space skyline object as a seed. However, it does not answer the question *how the subspace skylines are shaped by the full space skyline objects*.

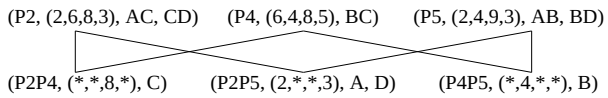
**Definition 3 (Seed skyline groups)** For a data set $S$ in space $\mathcal{D}$, an object in the full space skyline is called a *seed object*. The set of seed objects is denoted by $F(S)$.

The skyline groups on $F(S)$ are called the *seed skyline groups* with respect to data set $S$. The lattice of seed skyline groups is called the *seed lattice*, denoted by $\mathcal{SSG}(S)$. ■

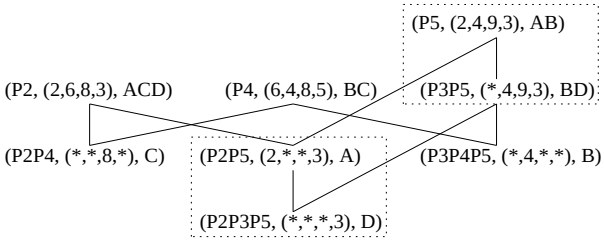
Then, *what is the relation between the skyline groups on $S$ and the seed skyline groups?*

**Example 2 (Seed lattice)** Consider the set $S$ of five objects in 4-d space $ABCD$ shown in Figure 2 as our running example.

Objects $P_2$, $P_4$ and $P_5$ are in the full space skyline. That is, $F(S) = \{P_2, P_4, P_5\}$ is the set of *seed objects*. Moreover, it can be verified that object $P_3$ is in the skylines of




(a) The seed lattice, i.e., the lattice of skyline groups and decisive subspaces on $F(S)$.




(b) The skyline groups and decisive subspaces on $S$.

**Figure 3. The skyline groups and decisive subspaces on $S$ and $F(S)$.**

| oid | $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|---|
| $P_1$ | 5 | 6 | 10 | 7 |
| $P_2$ | 2 | 6 | 8 | 3 |
| $P_3$ | 5 | 4 | 9 | 3 |
| $P_4$ | 6 | 4 | 8 | 5 |
| $P_5$ | 2 | 4 | 9 | 3 |

**Figure 2. An object set $S$ as our running example.**

subspaces $B$, $D$ and $BD$. Object $P_1$ is not in any subspace skylines.

Figures 3(a) and 3(b), respectively, show the *seed lattice*, i.e., the skyline group lattice on $F(S)$, and the lattice of all skyline groups on $S$. In the figures, we omit the unit elements and the zero elements in the lattices.

Interestingly, the two lattices of skyline groups and their decisive subspaces are quite similar to each other, except for the following differences.

Object $P_3$, which is not a seed object, shares the same value with seed objects $P_2$ and $P_5$ on dimension $D$, and $D$ is one of the decisive subspaces of group $P_2P_5$ in Figure 3(a). Thus, the skyline group $(P_2P_5, (2, *, *, 3), A, D)$ in Figure 3(a) is expanded to include a child group $(P_2P_3P_5, (*, *, *, 3), D)$ in Figure 3(b). Moreover, the decisive subspace of group $P_2P_5$ in Figure 3(b) is adjusted to $AD$ since dimensions $A$ and $D$ are needed together to distinguish these two seed objects from $P_3$ and establish their skyline membership as an exclusive group.

In addition, $P_3$ shares the same values with $P_5$ on dimensions $B$ and $D$, and $BD$ is a decisive subspace of $P_5$ in Figure 3(a). A new group $P_3P_5$ is added in Figure 3(b) to capture this sharing. The decisive subspace of group $P_5$ in Figure 3(b) remains $AB$.

$P_3$ also shares the same value with seed object $P_4$ on dimension $B$ which is a decisive subspace of group $P_4P_5$ in Figure 3(a). Thus, the group $P_4P_5$ in Figure 3(a) is extended to the group $P_3P_4P_5$ in Figure 3(b).

On the other hand, although object $P_1$ also shares with seed object $P_2$ the same value on dimension $B$, however, it does not cause any change in the skyline groups and the decisive subspaces since $B$ is not in the decisive subspace of $P_2$.

In summary, we obtain the following two observations.

- *The only differences between the two lattices are some splitting groups caused by some objects that are not in the full space skyline but share some common values with some seed objects in part of their decisive subspaces.*
- *The seed lattice is a quotient lattice of the skyline group lattice on $S$.* ■

Motivated by the observations in Example 2, we extend Theorem 1 [10] to the following more thorough result. Limited by space, we omit the proof here.

**Theorem 2 (Seed lattices)** *For a data set $S$, let $\mathcal{SG}_S$ be the skyline group lattice. Then, the seed lattice $\mathcal{SSG}(S)$ is a quotient lattice of $\mathcal{SG}_S$.* ■

Theorem 2 provides important insights into the structure of skyline groups. *The skyline groups on $S$ can be composed in two steps. First, we can find the seed skyline groups and the seed lattice formed by seed objects. Then, we can extend the seed skyline groups properly to include those objects that are not seeds but share some common values in the decisive subspaces.*

## 5 Algorithm Stellar

In this section, we develop algorithm *Stellar*, which computes skyline groups and decisive subspaces by searching only the full space skyline. First, we introduce the dominance matrix and the coincidence matrix, which will be used to compute the seed skyline groups and their decisive

| | $P_2$ | $P_4$ | $P_5$ |
|---|---|---|---|
| $P_2$ | $\emptyset$ | $AD$ | $C$ |
| $P_4$ | $B$ | $\emptyset$ | $C$ |
| $P_5$ | $B$ | $AD$ | $\emptyset$ |

(a) Dominance matrix $M_{dom}$

| | $P_2$ | $P_4$ | $P_5$ |
|---|---|---|---|
| $P_2$ | $ABCD$ | $C$ | $AD$ |
| $P_4$ | $C$ | $ABCD$ | $B$ |
| $P_5$ | $AD$ | $B$ | $ABCD$ |

(b) Coincidence matrix $M_{co}$

**Figure 4. Two matrices in Example 5.**

subspaces. Then, we explain how to find the seed skyline groups and the seed lattice. We also discuss how to extend the seed lattice to include those non-seed objects. Last, we present the whole algorithm.

Without loss of generality, we assume that there exist no objects $u, v \in S$ such that $u.D = v.D$ for every dimension $D$. If such a situation happens, the two objects can be bound together since they always appear together if they are involved in any skyline groups.

### 5.1 Dominance and Coincidence Matrices

**Example 3 (Dominance and coincidence matrices)** Consider our running example in Figure 2. We compute the skyline objects in the full space as the seed objects. As a byproduct, we can populate a *dominance matrix* $M_{dom}$ and a *coincidence matrix* $M_{co}$ as shown in Figure 4.

In the dominance matrix $M_{dom}$, the cell at row $P_i$ and column $P_j$, denoted by $dom_{P_i,P_j}$, records the dimensions on which $P_i$ has a smaller value than $P_j$. For example, $dom_{P_2,P_4} = AD$ since $P_2.A < P_4.A$ and $P_2.D < P_4.D$.

In the coincidence matrix $M_{co}$, the cell at row $P_i$ and column $P_j$, denoted by $co_{P_i,p_j}$, records the dimensions on which $P_i$ and $P_j$ share the same values. For example, $co_{P_2,P_4} = C$ since $P_2.C = P_4.C$. Clearly the coincidence matrix is symmetric, i.e., $co_{P_i,P_j} = co_{P_j,P_i}$. Moreover, $co_{P_i,P_i} = ABCD$. ∎

**Definition 4 (Dominance and coincidence matrices)** The *dominance matrix* $M_{dom} = \{dom_{o,o'}\}$, where $o, o' \in F(S)$, $dom_{o,o'} = \{D | D \in \mathcal{D}, o.D < o'.D\}$. The *coincidence matrix* $M_{co} = \{co_{o,o'}\}$, where $o, o' \in F(S)$, cell $co_{o,o'} = \{D | D \in \mathcal{D}, o.D = o'.D\}$. ∎

Apparently, we have the following property.

**Property 1 (Dominance and coincidence matrices)** *In the dominance matrix $M_{dom}$ and the coincidence matrix $M_{co}$, for any $o, o' \in F(S)$, (1) $dom_{o,o} = \emptyset$; (2) $co_{o,o} = \mathcal{D}$; and (3) $co_{o,o'} = co_{o',o} = (\mathcal{D} - dom_{o,o'} - dom_{o',o})$.* ∎

Property 1 indicates that a coincidence matrix is redundant. In implementation, we can either derive the coincidence matrix on the fly, or only store a triangle matrix since $co_{o,o'} = co_{o',o}$. To keep our description simple, we conceptually use the coincidence matrix. A dominance matrix and a coincidence matrix can be derived as the byproducts of finding skyline objects in the full space.

### 5.2 Identifying Seed Skyline Groups

Recall that for skyline group $(G, \mathcal{B})$, $\mathcal{B}$ is called the maximal subspace of $G$. Thus, to compute a skyline group, we need to determine its maximal subspace and its decisive subspaces.

#### 5.2.1 Maximal Subspaces of Seed Skyline Groups

Each seed object is unique and thus forms a skyline group in the full space. They are called the *singleton seed skyline groups*. In addition, if some seed objects share values on some dimensions, they may also form seed skyline groups.

**Example 4 (Other seed skyline groups)** Since $co_{P_2,P_5} = AD$, objects $P_2$ and $P_5$ share common values on dimensions $A$ and $D$. Interestingly, $P_2$ and $P_5$ have a smaller value than $P_4$, the only other seed object on both $A$ and $D$. Thus, $P_2P_5$ form a skyline group. Similarly, the other seed skyline groups $P_2P_4$ and $P_4P_5$ can be identified. ∎

Generally, we have the following result.

**Theorem 3 (Skyline groups by sharing)** *$(G, \mathcal{B})$ ($|G| > 1$) is a seed skyline group if and only if for any $w \notin G$, $\mathcal{B} = \bigcap_{u,v \in G, u \neq v} co_{u,v}$ and $\mathcal{B} \cap dom_{u,w} \neq \emptyset$.* ∎

Theorem 3 says, in order to find skyline groups, we do not need to search every subspaces. Instead, we only need to examine how objects share common values in subspaces. Moreover, we do not need to sort objects in various subspaces. The dominance matrix and the coincidence matrix are sufficient.

#### 5.2.2 Decisive Subspaces of Seed Skyline Groups

*How can we determine the decisive subspaces for a seed skyline group?*

**Example 5 (Decisive subspaces of seed objects)** $(P_2, (2, 6, 8, 3))$ is a skyline group. What are its decisive subspaces?

Since $dom_{P_2,P_4} = AD$, in all subspaces not containing dimensions $A$ and $D$, object $P_2$ is either dominated by $P_4$ or has the same values as $P_4$. Thus, to form a skyline group by itself, $P_2$ needs either dimension $A$ or dimension $D$. Similarly, since $dom_{P_2,P_5} = C$, $P_2$ needs dimension $C$ to be not dominated by $P_5$ and to be also different from $P_5$. Therefore, in any super-space of either $AC$ or $CD$, $P_2$ is a skyline object. In other words, $AC$ and $CD$ are the decisive subspaces for skyline group $(P_2, (2, 6, 8, 3))$.

Interestingly, if we treat each dimension as a binary variable, then the conjunctive normal form $(A \vee D) \wedge C$ represents the requirement of subspaces that $P_2$ itself is qualified as a skyline group. Each conjunction in the minimum disjunctive normal form, $(A \wedge C) \vee (C \wedge D)$, gives a decisive subspace.

Similarly, group $(P4, (6, 4, 8, 5))$ has a decisive subspace $BC$, and group $(P_5, (2, 4, 9, 3))$ has two decisive subspaces $AB$ and $BD$. This observation can also be extended to seed skyline groups of multiple objects. ■

Let us generalize the observation in Example 5.

**Theorem 4 (Decisive subspace)** *$\mathcal{C}$ is a decisive subspace of skyline group $(G, \mathcal{B})$ if and only if for each object $o \notin G$, there exists at least one dimension $D \in \mathcal{C}$ such that $G_D < o.D$.* ■

Theorem 4 discloses the inherent meaning of decisive subspaces. To facilitate finding the decisive subspaces, we collect the dominance information in the dominance matrix and the coincidence matrix as the byproducts of finding seed objects. Again, we do not need to search all subspaces. In fact, the dominance matrix is sufficient.

A disjunctive normal form $C_1 \vee \ldots \vee C_n$ is *minimum* if there exist no $C_i, C_j$ $(1 \leq i, j \leq n)$ such that $C_i \rightarrow C_j$. The following rule gives the decisive subspaces of seed skyline groups.

**Corollary 1 (Decisive subspaces)** *$\mathcal{C} = D_{i_1} \cdots D_{i_k}$ is a decisive subspace of a seed skyline group $(G, \mathcal{B})$ if and only if $D_{i_1} \wedge \ldots \wedge D_{i_k}$ is a conjunction in the minimum disjunctive normal form of formula $\bigwedge_{u \notin G}(\bigvee_{D \in (\mathcal{B} \cap dom_{o,u})} D)$, where $\{dom\}$ is the dominance matrix and $o$ is any object in $G$.* ■

Clearly, in our problem, only positive instances appear. The minimum disjunctive normal form is unique and can be computed in polynomial time. In fact, for a seed skyline group $G$, its decisive subspaces can be computed efficiently by scanning a row of $o \in G$ in the dominance matrix and using bitmap vectors to record the subspaces.

**Example 6 (Computing decisive subspaces)** Let us consider computing decisive subspaces for seed group $(P_5, (2, 4, 9, 3))$. We scan the row of $P_5$ in the dominance matrix. Since $dom_{P_5,P_2} = B$, we initiate a candidate subspace $B$. The next entry in the row is $dom_{P_5,P_4} = AD$. Thus, we make another copy of the existing candidate subspace and add to them $A$ and $D$, respectively. Then, we have two candidate subspaces: $AB$ and $BD$. If there are more entries in the row, we can extend the subspaces. At any time, we only maintain the minimal subspaces. That is, we never maintain a candidate subspace $\mathcal{C}'$ if there is another candidate subspace $\mathcal{C} \subset \mathcal{C}'$. ■

### 5.3 Accommodating Non-Seed Objects

**Example 7 (Non-seed objects)** Object $P_3$ in our running example shares the same values with object $P_5$ in subspace $BCD$, which is a superset of decisive subspace $BD$ of group $(P_5, (2, 4, 9, 3))$. Thus, we split the group into two groups. The first group contains $P_5$ and the second group contains $P_3P_5$.

Group $P_5$ has two decisive subspaces in the seed lattice: $AB$ and $BD$. $AB$ is not affected by $P_3$ since $P_3$ and $P_5$ does not share on $A$. Thus, $AB$ is still a decisive subspace for $P_5$ in the skyline lattice on the whole data set. On the other hand, $BD \subset BCD$. Therefore, $BD$ becomes the decisive subspace of the new group $(P_3P_5, (*, 4, 9, 3))$.

The expansion of group $(P_4P_5, (*, 4, *, *))$ is a little bit different. $P_4P_5$ share with $P_3$ on dimension $B$, which is the maximal subspace where $P_4P_5$ form a skyline group. Thus, we do not need to split the group. Instead, we only add in $P_3$. The decisive subspace of the group, $B$, remains the same. ■

We generalize the observation in Example 7 as follows.

**Theorem 5 (Non-seed objects)** *Given a set of objects $S$ in space $\mathcal{D}$. $(G, \mathcal{B})$ is a skyline group on $S$ if and only if there exists a seed skyline group $(G, \mathcal{B}')$ such that*

- *$G = G'$, $\mathcal{B} = \mathcal{B}'$, and there exists no other object $o \in (S - F(S))$ such that $G_{\mathcal{B}} = o_{\mathcal{B}}$; or*
- *$G' \subset G$, $\mathcal{B}' \supset \mathcal{B}$, $(G - G')$ is the set of all objects in $(S - F(S))$ that share common values with $G'$ in $\mathcal{B}$ but not in any proper super-space of $\mathcal{B}$, and there exists a decisive subspace $\mathcal{C}'$ of seed skyline group $(G', \mathcal{B}')$ such that $\mathcal{C}' \subseteq \mathcal{B}$.*

*A space $\mathcal{C} \subseteq \mathcal{B}$ is a decisive subspace of skyline group $(G, \mathcal{B})$ on $S$ if and only if*

- *$\mathcal{C}$ is a decisive subspace of seed group $(G', \mathcal{B}')$ on $F(S)$ and there exists no object $o \in (S - G)$ such that $o_{\mathcal{C}} = G_{\mathcal{C}}$; or*
- *there exists a decisive subspace $\mathcal{C}'$ of seed group $(G', \mathcal{B}')$ on $F(S)$ such that (1) $\mathcal{C}' \subset \mathcal{C}$; (2) there are some object $o \in (S - G)$, $o_{\mathcal{C}'} = G_{\mathcal{C}'}$ but $o_{\mathcal{C}} \neq G_{\mathcal{C}}$; and (3) $\mathcal{C}$ is the minimal subspace that have the above two properties.* ■

Theorem 5 gives the rules on how to accommodate non-seed objects in to the skyline lattice. Again, we do not need to search any subspaces for skylines. Instead, we only need to scan all those non-seed objects once against the seed lattice and make adjustments as indicated by the theorem.

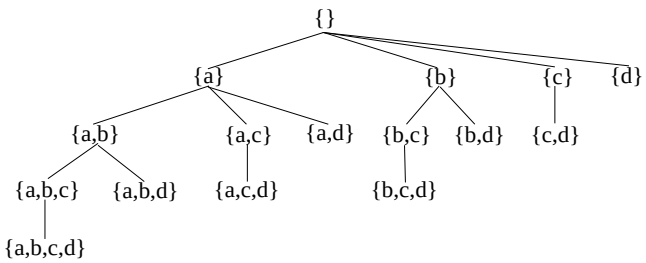



**Figure 5. A set enumeration tree.**

### 5.4 The Algorithm

Algorithm *Stellar* exploits our findings in the previous subsections to compute the complete set of skyline groups. Basically, the algorithm works in two steps. First, it computes the seed skyline groups. Then, it adds in the non-seed objects based on their sharing of common values with some seed skyline groups.

In the first step of computing the seed skyline groups, instead of searching subspace skylines in all subspaces like *Skyey* does, *Stellar* only considers how the seed objects coincide in subspaces.

To enumerate all possible subsets of seed objects that share common values in some subspaces, we can use a set enumeration tree [11]. A set enumeration tree uses a total order on the set of all objects. Each subset of objects is treated as a string such that all objects in the subset are sorted in the total order. Then, a subset $X$ is an ancestor of another subset $Y$ in the set enumeration tree if $X$ is a prefix of $Y$. For example, the set enumeration in Figure 5 enumerates all possible subsets of object set $\{a, b, c, d\}$, where the alphabetical order is used.

When there are many seed objects, the complete set enumeration tree can be huge. However, we do not need to search all subsets. Instead, we are only interested in those maximal c-groups – a set of objects $G$ share common values in the maximum subspace $\mathcal{B}$, and no proper superset of $G$ shares in $\mathcal{B}$.

**Example 8 (Search for maximal c-groups)** A segment of a coincidence matrix on objects $\{o_1, o_2, o_3, o_4, o_5\}$ is shown as follows.

| | $o_1$ | $o_2$ | $o_3$ | $o_4$ | $o_5$ |
|---|---|---|---|---|---|
| $o_1$ | $ABCD$ | $ACD$ | $B$ | $ABCD$ | $CD$ |
| $o_2$ | $ACD$ | $ABCD$ | $\emptyset$ | $CD$ | $BCD$ |

...

According to the set enumeration tree, all those maximal c-groups can be divided into 4 subsets: the ones having object $o_1$, the ones having object $o_2$ but no $o_1$, the ones having object $o_3$ but no $o_1$ and $o_2$, and the ones having object $o_4$ but no $o_1$, $o_2$ and $o_3$.

Let us find those maximal c-groups having $o_1$. $o_1$ and $o_2$ share common values in space $ACD$. Moreover, $o_4$ shares common values with $o_1$ in $ABCD$. Therefore, every subset having $o_1o_2$ cannot be maximal if it does not contain $o_4$. $o_1o_2o_4$ form a maximal c-group in space $ACD$. We can add in $o_5$ to form another maximal c-group in space $CD$. After searching the branch of $o_1o_2$, we turn to branch $o_1o_3$ and find maximal c-group $o_1o_3o_4$ in subspace $B$. Similarly, we can find the other maximal c-group having $o_1$: $o_1o_4$ in space $ABCD$. Note that $o_1o_5$ share common values in $CD$. However, $o_2$ and $o_4$ also share common values with $o_1$ on these two dimensions. Thus, $o_1o_5$ is not maximal.

Now, let us find those object subsets having $o_2$ but not $o_1$. First we consider $o_2o_4$, which share common values in subspace $CD$. However, $o_1$ also share common values with $o_2$ in $CD$. Thus, any subset containing $o_2o_4$ but not $o_1$ cannot be a maximal c-group. All the recursive search of $o_2o_4$ such as $o_2o_4o_5$ can be pruned. ■

The algorithms for finding the maximal c-groups is shown in Figure 6. Please note that the idea of computing the closure subsets using a set enumeration tree is not new at all. Instead, it has been exploited extensively in mining frequent closed itemsets [8], such as [9, 16]. However, the detailed algorithms in [9, 16] are different from the al-

**Input:** a set of seed objects $F(S)$ in space $\mathcal{D}$, and coincidence matrix $M_{co}$;
**Output:** the set of all maximal c-groups of seed objects;
**Method:**
1: let $F(S) = \{o_1, \ldots, o_l\}$;
2: for $i = 1$ to $(l-1)$ do
3: call *recursive-search*$(o_i, \{o_i\}, \{o_{i+1}, \ldots, o_l\}, \mathcal{D})$;

**Function** *recursive-search*$(u, G, H, \mathcal{B})$
31: let $G' = \{o | o \in (F(S) - G), co_{u,o} \supseteq \mathcal{B}\}$;
32: if $G' \not\subseteq H$ then return;
// any group containing $G$ but not $(G' - H)$
// cannot be maximal
33: let $G = G \cup G'$ and $H = H - G'$;
34: output $(G, \mathcal{B})$ as a maximal c-group;
35: let $H = \{o_{i_1}, \ldots, o_{i_k}\}$ such that $i_1 < \cdots < i_k$;
36: for $j = 1$ to $k$ do
37: let $\mathcal{B}' = co_{u,o_{i_j}} \cap \mathcal{B}$;
38: if $\mathcal{B}' \neq \emptyset$ then
39: let $H' = \{o | o \in H, co_{u,o} \supseteq \mathcal{B}'\}$;
40: call *recursive-search*$(u, G \cup \{o_{i_j}\}, H', \mathcal{B}')$;
return;

**Figure 6. Computing maximal c-groups.**

**Input:** a set of objects $S$ in space $\mathcal{D}$;
**Output:** the set of skyline groups;
**Method:**

1: compute full space skyline objects $F(S)$, as a byproduct, populate the dominance matrix and the coincidence matrix on $F(S)$;
2: compute maximal c-groups on $F(S)$ (Figure 6);
3: identify the decisive subspaces of the maximal c-groups using Corollary 1;
4: if a maximal c-group does not have a non-empty decisive subspace, then it is not a seed skyline group and thus is drop;
5: add in non-seed objects according to Theorem 5;

**Figure 7. Algorithm *Stellar*.**

gorithm here since the problem settings are different.

Based on the above discussion, we have algorithm *Stellar* as shown in Figure 7.

## 6 Experimental Results

In this section, we report an extensive performance study using both real data sets and synthetic data sets. We us: the Great NBA Players' technical statistics data set which is meaningful in practice. We also use synthetic data sets with uniform, correlated and anti-correlated distributions. We evaluate the efficiency and the scalability of both *Skyey* and *Stellar* with respect to dimensionality, database size and data distribution.

Both algorithms were implemented using Microsoft Visual C++ V6.0. Experiments were conducted on a PC with an Intel Pentium 4 3.0 GHz CPU and 1.0 GB main memory, running Microsoft Windows XP operating systems.

### 6.1 Results on Real Data Set NBA

The real data set used in our empirical study is the Great NBA Players' technical statistics from 1960 to 2001. The data set is available at the NBA official website (basketball-reference.com). In our experiments, we use the regular season player statistics, which contains the statistics of $17,265$ players, and is the largest table in the data set. There are 17 dimensions in this table. Hereafter, we refer to this table as the *NBA* data set. According to the semantics of this data set, the larger the dimension values (e.g., total points, total minutes), the better the player. Thus, a player with larger dimension values dominates those with smaller values.

In [10], the same table was used to evaluate and illustrate the meaningfulness of multidimensional skyline analysis.

To evaluate the scalability of *Skyey* and *Stellar* with respect to dimensionality, we run the two algorithms on the data set using the first $d$ dimensions, where $d$ varies from 1 to 17. The results are shown in Figure 8. The runtime is shown in logarithmic scale.

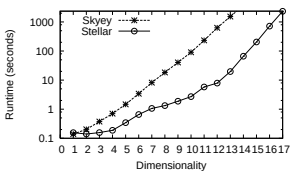



**Figure 8. Scalability w.r.t. dimensionality on real data set *NBA*.**

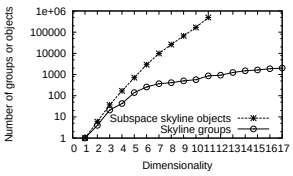



**Figure 9. Numbers of skyline groups and subspace skyline objects in real data set *NBA*.**

The results show that *Stellar* is much faster than *Skyey* on this data set. To fully understand the difference in efficiency, we plot the number of skyline groups and the number of subspace skyline objects in Figure 9, where the numbers of groups and objects are in logarithmic scale. If a player appears in the skylines of multiple subspaces, it is counted multiple times in the number of subspace skyline objects. This number is also the size of SkyCube in [15]. As mentioned before, *Skyey* computes SkyCube as a byproduct, and *Stellar* computes skyline groups directly.

The number of subspace skyline objects grows exponentially as the dimensionality increases. That is due to the exponential increase of the number of possible subspaces. *Skyey* has to compute the subspace skyline in every non-empty subspace. The cost grows exponentially as the dimensionality increases.

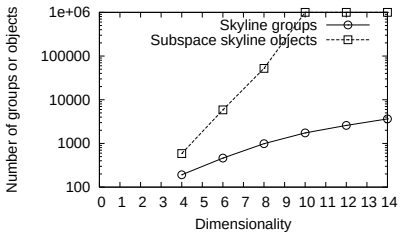
On the other hand, the number of skyline groups increases moderately with respect to dimensionality. In this data set, we observe that there are no other players sharing the same values with those skyline players on their decisive subspaces. In such a case, the number of skyline groups is bounded by the number of players in the full space skyline, and thus is not exponential with respect to dimensionality. Thus, *Stellar* saves substantially by computing skyline groups directly without searching subspaces for skylines.

As a typical application where skyline analysis is meaningful, the NBA data set does not have a large number of skyline objects.

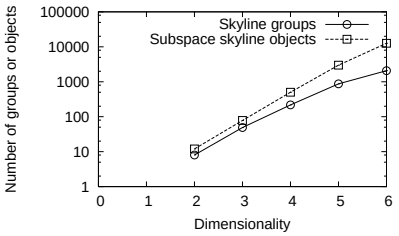
We also test the algorithms on some other real data sets. The results are consistent. Limited by space, we omit the details here. From the experimental results on the real data sets, we observe the following.

The number of skyline groups and the number of full space skyline objects are relatively small on real data sets, and often do not grow exponentially. That makes the skyline analysis on those data sets meaningful.

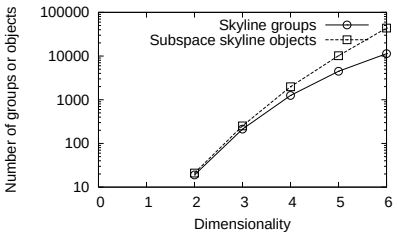
There are dimension value sharing in real data sets,




(a) Correlated distributed data set

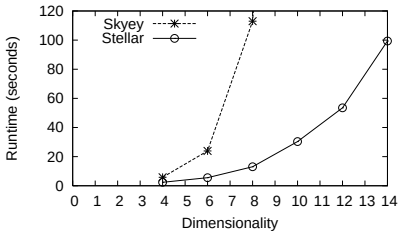



(b) Equally distributed data set

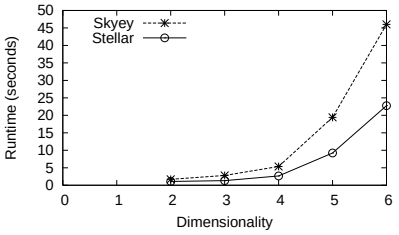



(c) Anti-correlated distributed data set

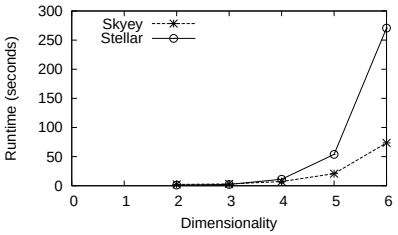
**Figure 10. Skyline distribution in three types of synthetic data sets, each data set has $100,000$ tuples.**




(a) Correlated distributed data set

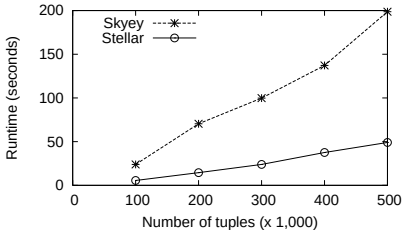



(b) Equally distributed data set

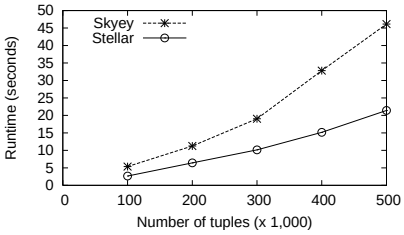



(c) Anti-correlated distributed data set

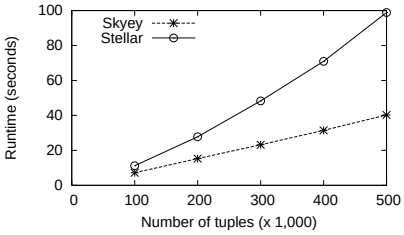
**Figure 11. Scalability w.r.t. dimensionality in three types of synthetic data sets, each data set has $100,000$ tuples.**




(a) Correlated distributed data set $6$ dimensions.




(b) Equally distributed data set $4$ dimensions.




(c) Anti-correlated distributed data set $4$ dimensions.

**Figure 12. Scalability w.r.t. database size in three types of synthetic data sets.**

though not heavy. Skyline groups provide a substantial summarization and compression of subspace skylines. On real data sets with a relatively high dimensionality, the number of subspace skyline objects still can be huge. Searching all subspaces is very costly. *Stellar* is clearly more efficient and more scalable than *Skyey* on real data sets since it exploits the seed skyline groups effectively and never searches all subspaces.

### 6.2 Results on Synthetic Data Sets

In order to fully understand the performance of *Skyey* and *Stellar* with respect to different data distributions, we use synthetic data sets. Particularly, we want to explore in what situations *Skyey* and *Stellar* have advantages, respectively.

We generate three types of data sets using the data generator provided by the authors of [1]: (1) the *correlated distributed data sets* where if a record is good in one dimension, likely it is also good in other dimensions; (2) the *equally distributed data sets* where the attribute values of the generated records are uniformly distributed; and (3) the *anti-correlated data sets* where if a record is good in one dimension, it is unlikely to be good in other dimensions. To introduce a moderate coincidence in dimensions, we truncate the values so that each number has $4$ digits in the decimal part.

Figure 10 shows the skyline distribution in the three types of data sets, where each data set has $100,000$ tuples. In the correlated data sets, the number of skyline groups is orders of magnitudes smaller than the number of subspace skyline objects, and does not grow exponentially. This distribution is similar to the one in the real data sets exam-

ined. In the equally or anti-correlated distributed data sets, the number of skyline groups and the number of subspace skyline objects increase almost exponentially, and the difference between the two numbers is not substantial. In such data sets, many objects in a subspace skyline also form groups individually in that subspace.

The scalability of the two methods with respect to dimensionality on the synthetic data sets is shown in Figure 11. In a correlated distributed data set, *Stellar* performs substantially better than *Skyey*. Not searching all the subspaces brings a tremendous benefit to *Stellar*. In an equally distributed data set, *Stellar* is still faster than *Skyey*, but the gap is much smaller than that in correlated distributed data sets. Interestingly, in anti-correlated distributed data sets, *Skyey* is even faster than *Stellar*. The reason is that most of the subspace skyline objects form skyline groups. On the other hand, there are much more objects ($100,000$ objects in this experiment) than the number of dimensions to search.

The experiments on the three types of data sets clearly illustrate the difference between the two algorithms. *Stellar* exploits skyline groups as a concise summarization of subspace skylines. When the summarization and compression by skyline group is effective, *Stellar* has good performance. However, if the compression ratio is low, then *Skyey* may have good performance since the dimensionality is typically smaller than the number of objects.

Furthermore, computing multidimensional skylines on anti-correlated distributed data sets is much more costly than on correlated distributed data sets. This is consistent with the results in previous studies (e.g., [1, 10, 15]). Although *Skyey* may performs better than *Stellar*, the runtime of both of them increases fast as the dimensionality grows.

Using the synthetic data sets, we also test the scalability of the two algorithms with respect to database size. The results are shown in Figure 12. Both algorithms are scalable with respect to database size. *Stellar* is faster than *Skyey* on correlated or equally distributed data sets, but slower than *Skyey* on anti-correlated data sets.

### 6.3 Summary

The extensive performance study clearly demonstrate the advantages of *Stellar* and *Skyey*. As analyzed before, *Skyey* searches subspaces for subspace skylines and merge them into skyline groups. *Stellar* exploits the skyline groups and avoids searching all subspaces. Thus, if the skyline groups summarize and compress subspace skylines well, then *Stellar* performs substantially better than *Skyey*. On the other hand, if most of the subspace skyline objects form a unique skyline group in their subspaces, then *Skyey* can be faster.

On the real data sets where skyline analysis is meaningful, it seems that the data distribution favors *Stellar*. On the other hand, in a real application with an exponential number of skyline objects, the advantage of skyline objects over the others is limited, and the meaningfulness of skyline analysis could be questionable.

## 7 Conclusions

While multidimensional skyline analysis is useful in many applications, computing multidimensional skylines and compressed skyline cubes is challenging since there are an exponential number of subspaces to search. The previous methods searching all subspaces may not achieve good performance when the dimensionality is high.

In this paper, we propose a novel and efficient method, *Stellar*, which exploits the seed skyline group lattice formed by full space skyline objects. We show that this skyline group lattice is easy to compute and can be extended to the skyline group lattice on all objects. After computing the skyline in the full space, *Stellar* only needs to enumerate skyline groups and their decisive subspaces using the full space skyline objects. Avoiding searching for skylines in an exponential number of subspaces improves the efficiency and the scalability of subspace skyline computation substantially in practice. An extensive performance study verifies the efficiency and the scalability of our new method.

## References


[1] S. Borzsonyi et al. The skyline operator. In *ICDE'01*.
[2] J. Chomicki et al. Skyline with pre-sorting. In *ICDE'03*.
[3] C.-Y. C. et al. Finding k-dominant skylines in high dimensional space. In *SIGMOD'06*.
[4] C. Y. C. et al. On high dimensional skylines. In *EDBT'06*.
[5] P. Godfrey et al. Maximal vector computation in large data sets. In *VLDB'05*.
[6] D. Kossmann et al. Shooting stars in the sky: an online algorithm for skyline queries. In *VLDB'02*.
[7] D. Papadias et al. An optimal and progressive algorithm for skyline queries. In *SIGMOD'03*.
[8] N. Pasquier et al. Discovering frequent closed itemsets for association rules. In *ICDT'99*.
[9] J. Pei et al. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *DMKD'00*.
[10] J. Pei et al. Catching the best views in skyline: A semantic approach. In *VLDB'05*.
[11] R. Rymon. Search through systematic set enumeration. In *KR'92*.
[12] K. Tan et al. Efficient progressive skyline computation. In *VLDB'01*.
[13] Y. Tao et al. Subsky: Efficient computation of skylines in subspaces. In *ICDE'06*.
[14] T. Xia and D. Zhang. Refreshing the sky: the compressed skycube with efficient support for frequent updates. In *SIGMOD'06*.
[15] Y. Yuan et al. Efficient computation of the skyline cube. In *VLDB'05*.
[16] M. J. Zaki and C. J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *SIAM DM'02*.