# Computing Closed Skycubes

Chedy Raïssi, Jian Pei, Thomas Kister

# Computing Closed Skycubes*

Chedy Raïssi
INRIA
Nancy Grand-Est
France
chedy.raissi@inria.fr

Jian Pei
School of Computing
Simon Fraser University
Burnaby, BC, Canada
jpei@cs.sfu.ca

Thomas Kister
LIRMM UMR CNRS 5506
University of Montpellier 2
Montpellier, France
thomas.kister@lirmm.fr

## ABSTRACT

In this paper, we tackle the problem of efficient skycube computation. We introduce a novel approach significantly reducing domination tests for a given subspace and the number of subspaces searched. Technically, we identify two types of skyline points that can be directly derived without using any domination tests. Moreover, based on formal concept analysis, we introduce two closure operators that enable a concise representation of skyline cubes. We show that this concise representation is easy to compute and develop an efficient algorithm, which only needs to search a small portion of the huge search space. We show with empirical results the merits of our approach.

## 1. INTRODUCTION

Recently, skyline analysis has attracted a lot of interest due to its importance in multi-criteria decision making applications. In a multidimensional space where a preference is defined for each dimension, a point $a$ dominates another point $b$ if $a$ is better (i.e., more preferred) than $b$ on at least one dimension, and $a$ is not worse than $b$ on every dimension. For example, a customer buying flight tickets from Singapore to Paris, France prefers a route of a low price, short travel time and few transits. We say that a route $a$ dominates another route $b$ if $a.price \leq b.price$, $a.traveltime \leq b.traveltime$, $a.n\_transits \leq b.n\_transits$, and at least one strict inequality holds. Given a set of points, the skyline set contains the points that are not dominated by any other points.

*The authors are grateful to the anonymous reviewers for their constructive and insightful comments on the paper. Chedy Raïssi contributed to this work when he was a research fellow at National University of Singapore. Jian Pei's research was supported in part by an NSERC Discovery Grant and an NSERC Discovery Accelerator Supplement Grant. 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 VLDB Endowment,* Vol. 3, No. 1


Traditional skyline computation has always been restricted to the data embedding space of a fixed dimensionality [2, 5, 11, 1]. Recently, the *subspace skyline* problem has attracted a fast growing amount of interest. Given a set of points in an $n$-dimensional space, users may be interested in different skyline queries in different subspaces of different dimensions. In our air-ticket example, an executive user may not care about the ticket price and thus will only focus on the subspace containing the attributes *travel time* and *number of transits*. On the other hand, a traveler for leisure may not have any time constraints and her/his only preference is on *ticket price*. As elaborated in this simple example, to handle various user queries in an efficient manner, a skyline analysis system needs to store and retrieve efficiently the skylines in the $2^n - 1$ possible non-empty subspaces, that is, taking into account all possible combination of attributes. The set of all skylines in all non-empty subspaces is called the *skycube*. This notion is similar to the idea of data cubes in the domain of data warehousing and OLAP analysis.

Although many existing skyline computation methods, such as pruning unnecessary dominance tests using spatial relationships among points, may be extended to tackle the skycube computation problem in one way or another, skycube computation has some unique challenges. Particularly, skycube needs to compute skyline sets in different subspaces. Can we leverage some skyline points in various subspaces to speed up skycube computation? This important direction has only been explored preliminarily. The two major existing studies [15, 9], which explore various strategies to share skyline computation in different subspaces, have a severe common drawback: in order to build a skycube, they have to enumerate and search skyline points over all possible subspaces. This naturally leads to poor performance on high-dimensional data sets. For example, for a 30-dimensional data set, there are $2^{30} - 1$ non-empty subspaces.

Pei *et al.* [7] proposed *Stellar*, a skycube computation method which avoids computing skylines in every subspace. It first computes seed skyline groups in the full space and then extends them to shape the final set of skyline groups and their associated decisive subspaces. *Stellar* is, however, still suffering from some drawbacks.

First, the performance of *Stellar* highly depends on the number of seed skyline groups, since in order to shape the final set of skyline groups, the seed skyline groups are tested against each point which is not a full space skyline. Poor performance may easily be caused by the *fragmentation* of the seed skyline groups – each skyline group may contain only very few points or even only one point in the full space.

As highlighted by the authors in their experiment results, this kind of artefact is very frequent for data sets with independent or anti-correlated distributions.

Second, while *Stellar* avoids enumerating all skyline points in every non-empty subspace using the skyline quotient lattice, it does not tackle the skyline computation in the second step, since *Stellar* does not take advantage of any properties or relationships between skyline points projected on different subspaces. These relationships may help to directly derive the skylines without any domination tests.

In this paper, motivated by the above observations, we study the problem of computing a *skycube concise representation*. We make two major contributions.

First, *we significantly reduce domination tests in a given subspace by identifying skylines that can be derived from skylines in other subspaces.* Skycube computation heavily relies on result sharing techniques in order to reduce the domination tests. We study different properties and identify two types of skylines: Type I skylines which can be *entirely derived with a single inference rule*, and Type II skylines that need advanced derivation rules or possibly domination tests.

Second, *we significantly reduce the number of subspaces needed to be searched.* Based on Galois connections, we define two dual closure operators for a skycube. One of those is then used to build a concise representation of the skycube. We develop an efficient depth-first search algorithm to efficiently compute the closed subspaces and effectively prune the search space.

The rest of the paper is organized as follows. We review the preliminaries and the problem of skycube computation in Section 2. We discuss related work in Section 3. Section 4 studies the different relations and inference properties between different subspaces. In Section 5, we introduce the skycube closure operators and discuss the properties. In Section 6, we develop our new algorithms. The experimental results are presented in Section 7. Section 8 concludes the paper. All mathematical proofs and pseudo-codes are provided in Appendices A and B.

## 2. PRELIMINARIES

Let $\mathcal{D} = \{d_1, \ldots, d_n\}$ be an $n$-dimensional space. Consider a set $\mathcal{S}$ of points in space $\mathcal{D}$. For a point $p$ in $\mathcal{S}$, we denote by $p(d_i)$ the value of $p$ on dimension $d_i$.

A subset $\mathcal{U} \subseteq \mathcal{D}$ is a **$|\mathcal{U}|$-dimensional subspace** of $\mathcal{D}$. Let $M$ be a set of subspaces, i.e., $M \subseteq 2^{\mathcal{D}}$, $\mathcal{U} \in M$ is said to be **maximal** in $M$ if $\nexists \mathcal{V} \in M$ such that $\mathcal{U} \subset \mathcal{V}$.

In a subspace $\mathcal{U} \subseteq \mathcal{D}$, a point $p$ is said to **dominate** another point $q$, denoted by $p \prec_{\mathcal{U}} q$, if $\forall d_i \in \mathcal{U}$, $p(d_i) \leq q(d_i)$ and $\exists d_j \in \mathcal{U}$, $p(d_j) < q(d_j)$. We denote by $p \prec\succ_{\mathcal{U}} q$ that $p$ and $q$ are **incomparable**, that is, neither $p$ dominates $q$ nor $q$ dominates $p$. Two points $p, q$ are **indistinct** in subspace $\mathcal{U}$ if $\forall d_i \in \mathcal{U}$, $p(d_i) = q(d_i)$, denoted by $p =_{\mathcal{U}} q$.

Definition 1 (Skyline and skycube). *Given a set $\mathcal{S}$ of points, a point $p \in \mathcal{S}$ is a **skyline point** (or a **skyline** for short) in a subspace $\mathcal{U}$ if $\nexists q \in \mathcal{S}$, $q \prec_{\mathcal{U}} p$. The **skyline set** $SKY(\mathcal{U})$ in subspace $\mathcal{U}$ contains all skyline points in $\mathcal{U}$.*

*The **skycube** is the set of skyline sets in all non-empty subspaces of $\mathcal{D}$, that is, $\{SKY(\mathcal{U}) | \mathcal{U} \in 2^{\mathcal{D}} - \emptyset\}$.*

It is easy to see that the indistinctness relation is an equivalence relation (i.e., the relation is reflexive, symmetric and transitive). Therefore, we can collect the skyline points indistinct from each other into a group.

| Objects | A | B | C | D |
|---|---|---|---|---|
| $o_1$ | 1 | 3 | 6 | 8 |
| $o_2$ | 1 | 3 | 5 | 8 |
| $o_3$ | 2 | 4 | 5 | 7 |
| $o_4$ | 4 | 4 | 4 | 6 |
| $o_5$ | 3 | 9 | 9 | 7 |
| $o_6$ | 5 | 8 | 7 | 7 |

**Table 1: The running example data set $\mathcal{S}$.**

Definition 2 (Indistinct/incomparable skyline). *A point $p \in SKY(\mathcal{U})$ is an **indistinct skyline** in subspace $\mathcal{U}$ if there exists another point $q \in SKY(\mathcal{U})$ such that $p \neq q$ and $p =_{\mathcal{U}} q$. A subset $X$ of $SKY(\mathcal{U})$ is an **indistinct skyline group** in $\mathcal{U}$, denoted by $X = \{p_i^{=}\}$, if (1) $|X| \geq 2$; (2) for any $p, q \in X$, $p =_{\mathcal{U}} q$; and (3) for any point $p \in X$, $q' \in SKY(\mathcal{U}) - X$, $p \prec\succ_{\mathcal{U}} q'$.*

*A point $p \in SKY(\mathcal{U})$ is called an **incomparable skyline** in space $\mathcal{U}$ if $p$ does not belong to any indistinct skyline group. In other words, $p$ is incomparable to any other points in $SKY(\mathcal{U})$. We denote $p$ by $p^{\prec\succ}$.*

The notions of indistinct skyline groups and incomparable skylines are critical in this work. Using indistinct skyline groups, we remove the redundant skyline information in a subspace. Indistinct skyline groups and incomparable skylines represent the essential and non-redundant skyline information in a subspace.

Definition 3 (Skyline (equivalence) class). *For subspaces $\mathcal{U}$ and $\mathcal{V}$, $SKY(\mathcal{U})$ and $SKY(\mathcal{V})$ are **equivalent**, denoted by $SKY(\mathcal{U}) =_{SKY} SKY(\mathcal{V})$, if (1) the two sets contain the same set of points; (2) $p$ is an incomparable skyline in $SKY(\mathcal{U})$ if and only if $p$ is an incomparable skyline in $SKY(\mathcal{V})$; and (3) a subset $X \subseteq \mathcal{S}$ is an indistinct skyline group in $SKY(\mathcal{U})$ if and only if $X$ is an indistinct skyline group in $SKY(\mathcal{V})$.*

*A **skyline (equivalence) class** $\mathcal{X} \subseteq 2^{\mathcal{D}} - \emptyset$ is a set of subspaces such that $\forall \mathcal{U}, \mathcal{V} \in \mathcal{X}$, $SKY(\mathcal{U}) =_{SKY} SKY(\mathcal{V})$ and $\forall \mathcal{V}' \in 2^{\mathcal{D}} - \emptyset - \mathcal{X}$, $SKY(\mathcal{U}) \neq_{SKY} SKY(\mathcal{V}')$. We denote by $[\mathcal{U}]$ the skyline class that $\mathcal{U}$ belongs to.*

The notion of skyline class captures another type of redundant skyline information in multidimensional skyline analysis. We can group subspaces into concise skyline classes.

Example 1 (Running example). *In this paper, we use the data set $\mathcal{S}$ in Table 1 containing 4 dimensions and 6 objects as the running example. The skycube contains the skyline sets in the 15 non-empty subspaces listed in Table 2.*

*In our example, subspace $\{A\}$ contains two indistinct skyline points as $o_1(A) = 1 = o_2(A)$ and 1 is the minimal value on subspace $\{A\}$. To highlight the different nature of the skyline points, either indistinct or incomparable, the skylines in subspace $\{A, D\}$ can be written as a multiset: $\{\{o_3^{\prec\succ}\}, \{o_4^{\prec\succ}\}, \{o_1^{=}, o_2^{=}\}\}$. Moreover, $SKY(\{A, D\}) \neq_{SKY} SKY(\{B, D\})$, since the skyline points for $\{B, D\}$ can be represented by the multiset $\{\{o_1^{=}, o_2^{=}\}, \{o_4^{\prec\succ}\}\}$.*

## 3. RELATED WORK

There is a rich body of literature on skyline computation. Kung *et al.* [5] studied the *Pareto frontier* for a finite set of

| Subspace | Skyline | Subspace | Skyline |
|---|---|---|---|
| A,B,C,D | $o_2^{\nearrow}, o_3^{\nearrow}, o_4^{\nearrow}$ | B,C | $o_2^{\nearrow}, o_4^{\nearrow}$ |
| A,B,C | $o_2^{\nearrow}, o_4^{\nearrow}$ | B,D | $o_1^{=}, o_2^{=}, o_4^{\nearrow}$ |
| A,B,D | $o_1^{=}, o_2^{=}, o_3^{\nearrow}, o_4^{\nearrow}$ | C,D | $o_4^{\nearrow}$ |
| A,C,D | $o_2^{\nearrow}, o_3^{\nearrow}, o_4^{\nearrow}$ | A | $o_1^{=}, o_2^{=}$ |
| B,C,D | $o_2^{\nearrow}, o_4^{\nearrow}$ | B | $o_1^{=}, o_2^{=}$ |
| A,B | $o_1^{=}, o_2^{=}$ | C | $o_4^{\nearrow}$ |
| A,C | $o_2^{\nearrow}, o_4^{\nearrow}$ | D | $o_4^{\nearrow}$ |
| A,D | $o_1^{=}, o_2^{=}, o_3^{\nearrow}, o_4^{\nearrow}$ | | |

**Table 2: The skycube of the data set $\mathcal{S}$ in Table 1.**

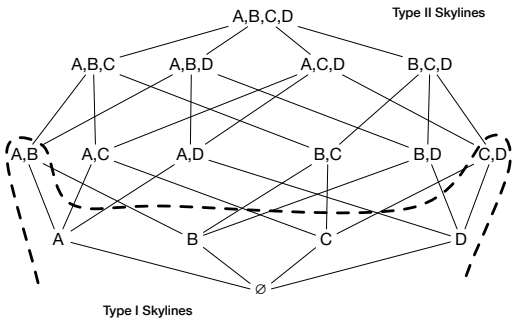



**Figure 1: The border between type I and type II skylines for the toy data set.**

vector. Börzsönyi *et al.* [2] introduced skyline queries into the database community. Since then, many methods have been developed for answering skyline queries, possibly with various constraints or in different computational environment. An extensive survey on skyline computation methods is interesting, but far beyond the capacity of this paper.

Until recently, most of the existing studies compute skylines in a given space. Moreover, as domination tests are the major cost in skyline computation, those methods explore various ways to avoid or speed up domination tests.

Recently, Yuan *et al.* [15] computed skylines in all non-empty subspaces in a batch and introduced the notion of skyline cube. Simultaneously, Pei *et al.* [9] introduced the notion of skycube semantics based on skyline groups and their associated decisive subspaces. The two groups collaboratively integrated the two methods [10]. Moreover, Xia and Zhang [14] studied the incremental maintenance of skyline cubes.

As discussed in Section 1, two new challenges emerge in skyline cube computation. First, we need to compute skylines in many subspaces. Second, it is non-trivial to develop pruning techniques to reduce domination tests using skylines crossing subspaces. Only very limited effort has been pursued to tackle those challenges. Specifically, the first approaches for skyline cube computation [15, 9, 10] need to examine the skylines in every possible subspace, which is costly when the dimensionality is high. Recently, Pei *et al.* [7] proposed a new algorithm to compute skyline groups and decisive subspaces without enumerating all the possible subspaces. Still, as analyzed in Section 1, the method in [7] does not explore domination test reduction crossing subspaces. Tao *et al.* [12] developed a filtering technique to reduce candidates in subspace skyline queries. However, the method is not set for skyline cube computation, and has to be applied to subspaces one by one.

Our study is also related to formal concept analysis [3]. Although formal concept analysis has been used in many data analysis and data mining aspects, such as mining frequent closed itemsets [6], its application in skyline cube computation has only been explored to a limited extent [9, 7]. Critically different from [9, 7], this study pursues a more thorough investigation on using equivalent classes of objects as well as of subspaces to compute and compress skylines.

# 4. SKYLINES DERIVATION

Domination tests, which assess whether a point is dominated by some other points in the data set, contribute to the major cost in skyline computation. This cost may become even more severe in skycube computation since we have to compute skylines in many subspaces. Orthogonal to many techniques speeding up domination tests, a fundamental theoretical question is whether a skyline in a subspace can be derived without domination tests.

Pei *et al.* [9] pointed out that the skyline membership monotonicity does not hold in general. Even if an object $p$ belongs to $SKY(\mathcal{U})$ and $SKY(\mathcal{W})$ such that $\mathcal{U} \subset \mathcal{W}$, $p$ may not belong to $SKY(\mathcal{V})$ where $\mathcal{U} \subset \mathcal{V} \subset \mathcal{W}$. In this section, we observe some interesting monotonic properties of skyline membership which can be applied to efficiently compute a skycube. The key idea is to carefully categorize skyline points into indistinct skyline points and incomparable skyline points, and use them in inferences.

## 4.1 Derivation Rules

THEOREM 1 (SKYLINE UNION DERIVATION). *For subspaces $\mathcal{U}$ and $\mathcal{V}$, if $p \in SKY(\mathcal{U})$ and $p \in SKY(\mathcal{V})$, then $p \in SKY(\mathcal{U} \cup \mathcal{V})$.*

COROLLARY 1. *For subspaces $\mathcal{U}$ and $\mathcal{V}$,*

$$SKY(\mathcal{U}) \cap SKY(\mathcal{V}) \subseteq SKY(\mathcal{U} \cup \mathcal{V}).$$

EXAMPLE 2. *In our running example (Table 1), $o_1$ is a skyline in subspaces $\{A\}$ and $\{B\}$. Thus, by theorem 1, $o_1$ is a skyline in subspace $\{A, B\}$. Furthermore, object $o_2$ is a skyline point in subspaces $\{A, C\}$ and $\{B\}$. Using Corollary 1, we have $\{o_2\} \subset SKY(\{A, B, C\})$.*

Theorem 1 and Corollary 1 show that it is feasible to efficiently derive some skyline points in a subspace without any domination tests. When Theorem 1 and Corollary 1 are applicable, only simple set intersection operations are needed to derive those skyline points.

Under what conditions, can we derive the complete skyline set in a subspace without domination tests?

DEFINITION 4 (TYPES OF SUBSPACES). *A subspace $\mathcal{U}$ is called a **type I subspace** if all points in $SKY(\mathcal{U})$ are indistinct from each other, that is, $\forall p, q \in SKY(\mathcal{U})$, $p =_{\mathcal{U}} q$.*

*A subspace is of **type II** if it is not of type I.*

THEOREM 2. *For subspaces $\mathcal{U}$ and $\mathcal{V}$ such that $SKY(\mathcal{U}) \cap SKY(\mathcal{V}) \neq \emptyset$, if $\mathcal{U}$ and $\mathcal{V}$ are type I subspaces, then $\mathcal{U} \cup \mathcal{V}$ is a type I subspace, and $SKY(\mathcal{U}) \cap SKY(\mathcal{V}) = SKY(\mathcal{U} \cup \mathcal{V})$.*

Theorem 2 states that the skyline set in a type I subspace may be efficiently computed using set intersection operations. Obviously, every one dimensional subspace is a type I

subspace. The skyline set in a one dimensional subspace can be obtained by a single scan of the data set even without any index. Thus, we can extract the skyline sets in type I subspaces in a bottom-up manner without any domination tests. Figure 1 shows the border between the two different types of subspaces in our running example data set.

We can also use incomparable skylines to derive some skylines in a type II subspace.

THEOREM 3 (INCOMPARABILITY RULE). *If $p$ is an incomparable skyline in subspace $\mathcal{U}$, then for any subspace $\mathcal{V}$ such that $\mathcal{U} \subseteq \mathcal{V}$, $p \in SKY(\mathcal{V})$.*

Theorem 3 can be regarded as a corollary of Theorem 1 in [9]. Our new contribution here is that Theorem 3 can be used to derive skylines in type II subspaces directly.

EXAMPLE 3. *In our running example, $SKY(\{A, C\}) = \{o_2^{\prec\succ}, o_4^{\prec\succ}\}$, that is, points $o_2$ and $o_4$ are incomparable skyline points in subspace $\{A, C\}$. Using Theorem 3, we know that $o_2$ and $o_4$ also belong to $SKY(\{A, B, C\})$, $SKY(\{A, C, D\})$ and $SKY(\{A, B, C, D\})$.*

In Table 3, we list the types and skylines of the subspaces in our running example. Theorems 2 and 3 are effective and efficient in deriving skylines in different subspaces in a bottom-up manner. Only a small number of skylines need to be computed using techniques other than the two theorems (the rightmost column in the table).

## 4.2 Computing Skylines by Domination Tests

Let us discuss the two cases where domination tests are needed.

### *Indistinct skylines*

When we try to infer the skylines in a subspace $\mathcal{W}$ from the skylines in a subspace $\mathcal{U}$ ($\mathcal{U} \subset \mathcal{W}$), $SKY(\mathcal{U})$ may contain indistinct skyline points (i.e., $\exists p, q \in SKY(\mathcal{U}), p =_{\mathcal{U}} q$). Those indistinct skyline points make the incomparable derivation rule not applicable. In this case, the following rule apply.

THEOREM 4. *For subspaces $\mathcal{U}$ and $\mathcal{W}$ such that $\mathcal{U} \subset \mathcal{W}$, a point $p \in SKY(\mathcal{U})$ is a skyline point in $\mathcal{W}$ if and only if $\nexists q$ such that $q =_{\mathcal{U}} p$, $q \prec_{\mathcal{W}-\mathcal{U}} p$.*

In order to conduct the domination tests in Theorem 4, a block-nested-loop approach similar to BNL [2] can be used. Critically, in the block-nested-loop, we only need to compare in subspace $\mathcal{W} - \mathcal{U}$ the indistinct skylines in subspace $\mathcal{U}$. This rule reduces significantly the size of the input and thus the number of domination tests comparing to a thorough search in subspace $\mathcal{W}$.

### *New Incomparable Skylines*

So far, we only try to derive skylines in a space from skylines in subspaces. However, not all skylines in a space are skylines in some subspaces.

EXAMPLE 4. *In our running example, $SKY(\{A, D\}) = \{o_1^{=}, o_2^{=}, o_3^{\prec\succ}, o_4^{\prec\succ}\}$ but $SKY(\{A\}) = \{o_1^{=}, o_2^{=}\}$ and $SKY(\{D\}) = \{o_4^{\prec\succ}\}$. Point $o_3$ is a skyline point in subspace $\{A, D\}$, though it is not a skyline in either $\mathcal{A}$ or $\mathcal{D}$. This type of skyline points cannot be inferred by the previous derivation rules.*

We call a point $p$ a **new incomparable point** in subspace $\mathcal{W}$ if $p$ is incomparable with any skyline points in any subspace $\mathcal{U} \subset \mathcal{W}$. Some new incomparable points may be skyline points. We identify the subset of new incomparable points which may be skyline points as follows.

DEFINITION 5 (ENTAILED CANDIDATE). *A point $p$ is an **entailed candidate** in a subspace $\mathcal{U}$ if $\forall d_i \in \mathcal{U}$, $min_{SKY(\mathcal{U})}(d_i) \leq p(d_i) \leq max_{SKY(\mathcal{U})}(d_i)$, where $min_{SKY(\mathcal{U})}(d_i)$ (respectively $max_{SKY(\mathcal{U})}(d_i)$) is the minimum (respectively the maximum) value on dimension $d_i$ of all the skyline points in subspace $\mathcal{U}$ obtained by applying the derivation rules and the indistinct skylines.*

We have the following rule.

THEOREM 5. *In a subspace $\mathcal{U}$, if a point $p$ is not a skyline point obtained by applying the derivation rules and the indistinct skylines, and is not an entailed candidate, $p$ is not a skyline.*

Theorem 5 enables the construction of a reduced set of skyline candidates $\mathcal{L}$ whose dimension values are entailed by the previously computed skyline points. This entailment property, which can be seen as a pruning technique, enables the reduction of the input size (which otherwise would be the set of *all* data points) and can be efficiently implemented using an index such as $B^+$-Trees.

EXAMPLE 5. *In our running example, for subspace $\mathcal{W} = \{A, D\}$, using the incomparability rule (Theorem 3), we derive from $SKY(\{D\}) = \{o_4\}$ that $o_4 \in SKY(\mathcal{W})$. The indistinct skylines $o_1, o_2$ from subspace $\{A\}$ are also skyline points in $\mathcal{W}$. In order to find incomparable skylines candidates, we must extract all points $p$ such that: $1 \leq p(A) \leq 4$ and $6 \leq p(D) \leq 8$. The candidates set is $\mathcal{L} = \{o_3, o_5\}$. Furthermore, $o_3 \prec_{\mathcal{U}} o_5$, thus, $SKY(\mathcal{W}) = \{o_1^{=}, o_2^{=}, o_3^{\prec\succ}, o_4^{\prec\succ}\}$.*

# 5. CONCISE REPRESENTATION FOR SKYCUBES

All the previous approaches on skycube computation except for [7] have to process all possible subspaces, and thus are unscalable when coping with high-dimensional data sets. For example, to process a 30-dimensional data set, classical algorithms have to process $2^{30} - 1$ subspaces.

Here we propose to reduce the number of subspaces to be processed using notions from formal concept analysis [4], which provides an elegant mathematical framework for a concise skycube representation based on closure operators.

Formally, let $\mathcal{O} = \mathcal{S} \times \{\prec\succ, =\}$ be the space of indistinctness/incomparableness annotations of all objects in space $\mathcal{D}$. A *skyline context* is a triplet $(\mathcal{O}, 2^{\mathcal{D}}, R)$, where $R \subseteq \mathcal{O} \times 2^{\mathcal{D}}$ represents the skylines in subspaces. A tuple $(p,\mathcal{U}) \in R$ means that the object $p$ is a skyline in the subspace $\mathcal{U}$. Apparently, a skyline context can be represented in a 2-dimensional table where each row represents a unique object and each column represents a unique subspace. Table 4 illustrates the skyline context in our running example.

To apply formal concept analysis, we build a Galois connection on the skyline context (i.e., a particular correspondence between the two partially ordered sets $2^{\mathcal{O}}$ and $2^{\mathcal{D}}$).

| | Type I subspace | Type II subspace | |
|---|---|---|---|
| subspaces | Derived from Theorem 2 | Derived from Theorem 3 | Others |
| $\{A, B\}$ | $o_1, o_2$ | – | – |
| $\{A, C\}, \{B, C\}$ | – | $o_4$ | $o_2$ |
| $\{A, D\}$ | – | $o_4$ | $o_1, o_2, o_3$ |
| $\{B, D\}$ | – | $o_4$ | $o_1, o_2$ |
| $\{C, D\}$ | $o_4$ | – | – |
| $\{A, B, C\}, \{B, C, D\}$ | – | $o_2, o_4$ | – |
| $\{A, B, D\}$ | – | $o_3, o_4$ | $o_1, o_2$ |
| $\{A, C, D\}, \{A, B, C, D\}$ | – | $o_2, o_3, o_4$ | – |

**Table 3: The skyline points in the running example data set based on their types.**

| **Objects** | $A$ | $B$ | $C$ | $D$ | $A,B$ | $A,C$ | $A,D$ | $B,C$ | $B,D$ | $C,D$ | $A,B,C$ | $A,B,D$ | $A,C,D$ | $B,C,D$ | $A,B,C,D$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $o_1^{=}$ | × | × | | | × | | × | | × | | | × | | | |
| $o_1^{\prec\succ}$ | | | | | | | | | | | | | | | |
| $o_2^{=}$ | × | × | | | × | | × | | × | | | × | | | |
| $o_2^{\prec\succ}$ | | | | | | × | | × | | | × | | × | × | × |
| $o_3^{=}$ | | | | | | | | | | | | | | | |
| $o_3^{\prec\succ}$ | | | | | | | × | | | | | × | × | | × |
| $o_4^{=}$ | | | | | | | | | | | | | | | |
| $o_4^{\prec\succ}$ | | | × | × | | × | × | × | × | × | × | × | × | × | × |

**Table 4: The skyline context in our running example.**

Definition 6. *For a set of objects $O \subseteq \mathcal{O}$, define $\alpha : 2^{\mathcal{O}} \to 2^{\mathcal{D}}$ as follows, which returns the set of maximal subspaces where only the objects in $O$ are skylines:*

$$\alpha(O) = \{\ \mathcal{U} \in 2^{\mathcal{D}} \mid SKY(\mathcal{U}) = O,\ \mathcal{U} \text{ is maximal}\ \} \quad (1)$$

*For a set of subspaces $M \subseteq 2^{\mathcal{D}}$, define $\beta : 2^{\mathcal{D}} \to 2^{\mathcal{O}}$ as follows, which returns the set of objects which are the skylines in each of the subspaces in $M$:*

$$\beta(M) = \{\ o \in \mathcal{O} \mid o \in SKY(\mathcal{U}),\ \forall \mathcal{U} \in M\ \} = \bigcap_{\mathcal{U} \in M} SKY(\mathcal{U}) \quad (2)$$

Example 6. *Consider the skyline context in Table 4. Let $O = \{o_2^{\prec\succ}, o_4^{\prec\succ}\}$, $\alpha(O) = \{\ \{A, B, C\}, \{B, C, D\}\ \}$. There are some other subspaces where $o_2$ and $o_4$ are unique incomparable skylines, such as $\{A, C\}$ and $\{B, C\}$. Those subspaces, however, are not maximal.*

*Let $M = \{\ \{A\},\ \{A, D\}\ \}$, $\beta(M) = \{o_1^{=}, o_2^{=}\}$, which is the intersection of the sets of skylines in subspaces $\{A\}$ and $\{A, D\}$.*

Property 1. *The pair $(\alpha, \beta)$ is a Galois connection between the two posets $(2^{\mathcal{O}}, \subseteq)$ and $(2^{\mathcal{D}}, \subseteq)$.*

The two operators $\alpha$ and $\beta$ are said to be dually adjoint. Apparently, the following holds.

$$O \subseteq \beta(M) \Longleftrightarrow M \subseteq \alpha(O) \quad (3)$$

Based on the previous Galois connection properties we can build two closure operators $h_* = \alpha \cdot \beta$ and $h^* = \beta \cdot \alpha$. In fact, in this work we are interested in only one of them: $h_*$.

Theorem 6. *$h_*$ and $h^*$ are closure operators.*

The closure $h_*(M)$ of a set of subspaces $M \subseteq 2^{\mathcal{D}}$ returns the maximal subspaces $\mathcal{U}$ having the same skylines as $M$.

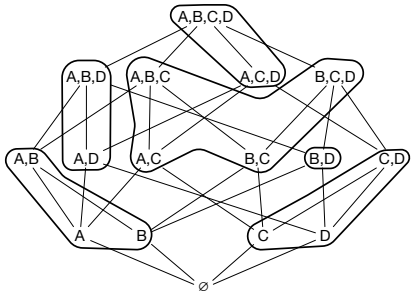



**Figure 2: The different skyline equivalence classes based on the closure operator $h_*$ in the running example.**

Obviously, this closure operator induces a partitioning of the skycube based on the skylines in subspaces.

Definition 7. ***Closed subspaces and generators.*** *For a set of subspaces $M \subseteq 2^{\mathcal{D}}$, if $h_*(M) = M$, every subspace in $M$ is called closed. A subspace $\mathcal{U}$ is a generator if no proper subspace of $\mathcal{U}$ is in the same equivalence class $[\mathcal{U}]$.*

Example 7. *The equivalence classes in the running example are illustrated in Figure 2. An equivalence class may have multiple closed subspaces and generators. For example, $[\{A, B, C\}]$ has closed subspaces $\{A, B, C\}$ and $\{B, C, D\}$, and generators $\{A, C\}$ and $\{B, C\}$.*

Lemma 1. *For a subspace $\mathcal{U}$ and any subspace $M \in h_*(\{\mathcal{U}\})$, $SKY(\mathcal{U}) = SKY(M)$.*

The collection of closed subspaces and the generators of all equivalence classes are sufficient to represent a skycube – the skylines in any subspaces can be inferred from the closed subspaces and its associated generators.

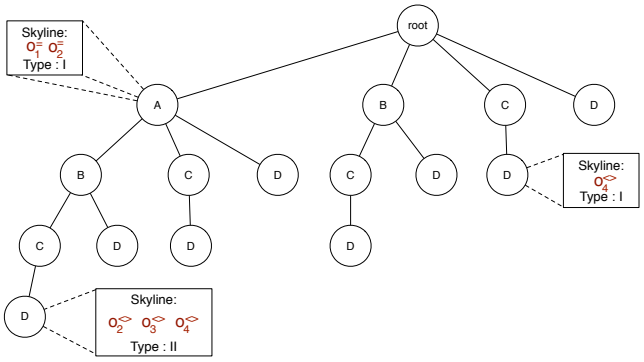



**Figure 3: The prefix-tree data structure.**

## 6. ALGORITHMS

In this section, we introduce three different algorithms based on the theoretical results in Sections 4 and 5. *Orion* is a breadth-first search algorithm that enumerates all subspaces and uses all derivation rules to minimize domination tests. Its variant, the *Orion-tail* algorithm, uses the entailment property in order to reduce the number of candidates for the non-trivial skylines that cannot be derived using the skyline inference rules. The last algorithm, *Orion-clos*, makes use of the collection of closed subspaces and the generators, and follows the popular *divide-and-conquer* paradigm in a depth-first search to compute and store closed subspaces and their associated skylines. Our algorithms share some similarities with the frequent closed itemset mining algorithms, like CLOSET and CLOSET+ [8, 13]. To tackle the presence of multiple closed subspaces in a skyline equivalence class, we introduce a new strategy for pruning the search space.

To enumerate subspaces, all our algorithms use a prefix-tree which stores additional information for each subspace. Each node stores the identity of the subspace it represents, the type of the subspace (i.e., I or II), and a set of skyline points. In this skyline set, each point is marked as either incomparable or indistinct. In addition to the prefix-tree, the *Orion-tail* variant also uses $B^+$-Trees, one for each dimension, in order to speed up the interval queries needed for the selection of entailment candidates. The *Orion-clos* algorithm also maintains a list (implemented as a hash list) of closed subspaces and the associated skyline points. The prefix-tree structure for our running example is shown in Figure 3, where each node is labeled by the new dimension added at the node, and the space represented by the node is the collection of dimensions from the root to the node.

Algorithms *Orion* and *Orion-tail* are simple level-wise search algorithms. They enumerate iteratively the skycube lattice, starting from the simple 1-dimensional subspaces and stopping after processing the maximum space $\mathcal{D}$. At each iteration, the algorithms check if the current subspace is of Type I or II, apply the corresponding inference rules, and conduct, if necessary, classical domination tests. In the *Orion-tail* algorithm, the last step of the classical domination tests comes after generating and pruning an entailed candidate in order to reduce the number of domination tests. The pseudo-codes are given in Algorithm 1 in Appendix B.

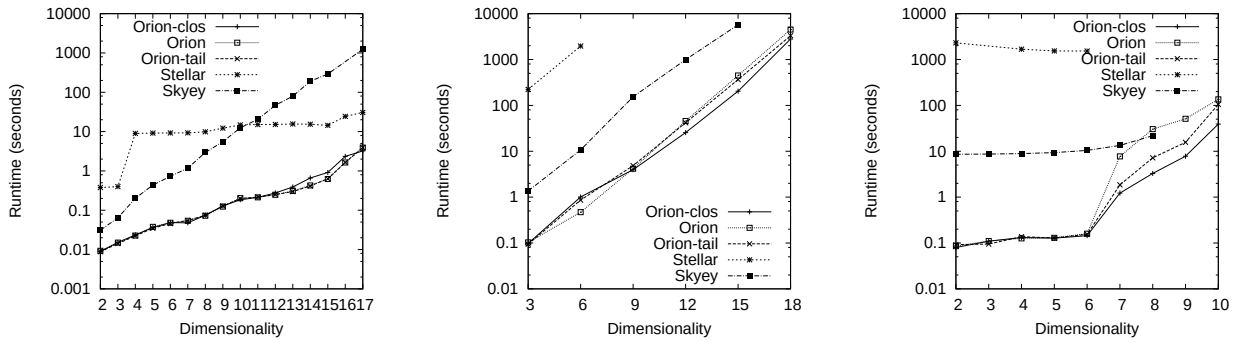
*Orion-clos* uses a divide-and-conquer strategy and is based on a recursive function $depthFirstSkyComputation()$ which explores the skycube lattice in a depth-first manner. Each call to the function only considers a portion of the search space corresponding to the superspaces of current subspace at the search frontier.

At the first step, the skylines in the 1-dimensional subspaces are computed and inserted into the prefix-tree as the children of the *root* node. At the same time, the skyline points in the full space are computed, since the full space is always a closed subspace. Then, each of the new child nodes are processed. The *Orion-clos* algorithm follows the same reasoning as in algorithm *Orion* and relies heavily on the derivation rules to reduce domination tests. Thus, for each node processed in a depth-first manner, the algorithm checks if the subspace is of Type I, and thus the skylines are computed only using Theorem 2. If the subspace is of Type II, then some of the skyline points are derived using the incomparability rule (directly copying the incomparable points from the direct parent node, and copying the incomparable points from the level 1 node having the same label) and with an indistinct skylines comparison on the newly added dimension. After this step, a classical domination test (the BNL function) is applied to discover the remaining non-derivable skyline points. In the depth-first search space traversal, the selection of entailment candidates cannot be applied since the pruning technique requires that all subspaces must be already processed.

Thanks to the properties of the closed subspaces and their generators, some parts or even entire branches (i.e., those subspaces with the same closed subspaces) do not need to be processed as they are already included in a skyline equivalence class. In order to decide whether we can prune a subspace on a branch, the current node representing the current processed subspace is compared with the closed subspaces in the list $\mathcal{LC}$. This step is similar in spirit to the closed frequent itemset mining. However, for any itemset, there exists only a unique closed itemset. In our case this is not true since there may be more than one closed subspace in an equivalence class. Consequently, the *Orion-clos* algorithm determines, for each node $\mathcal{W}$, which sub-branches may contain potential subspaces not covered by any of the previously discovered closed subspaces, and prunes the other sub-branches by simply applying the set difference $\mathcal{D} \backslash \bigcup \{$ all subspaces $\mathcal{U}$ in closed subspaces sets s.t $\mathcal{W} \subset \mathcal{C}\}$. This operation is implemented using bitmap vectors for efficiency reasons. Finally, if a subspace $\mathcal{W}$ is not covered by any of the closed subspaces and is maximal, it is added to the list of closed subspaces. The pseudocode of the algorithm is given in Algorithm 2 in Appendix B.

EXAMPLE 8 (*Orion-clos*). *In our running example, we start from node A's branch. The node B representing subspace* $\{A, B\}$ *is generated and processed. The subspace is in type I because* $SKY(\{A\}) \cap SKY(\{B\}) = \{o_1^=, o_2^=\} \neq \emptyset$. *No more operations are needed for this subspace. Subspace* $\{A, B\}$ *is not covered by any of the closed subspaces in the list* $\mathcal{LC} = \{\{A, B, C, D\}\}$, *since* $SKY(\{A, B, C, D\}) \neq SKY(\{A, B\})$ *(recall that the full space skyline is computed and introduced as a closed subspace at the beginning of the algorithm). Hence, the algorithm continues to the branch in a depth-first fashion.*

*The subspace* $\{A, B, C\}$ *is in Type II, since* $SKY(\{A, B\}) \cap SKY(\{C\}) = \emptyset$. *The algorithm thus uses the incomparability rule and copies the incomparable*




(a) The NBA data set (b) The MBL data set (c) The IPUMS data set

**Figure 5: Scalability w.r.t. dimensionality.**

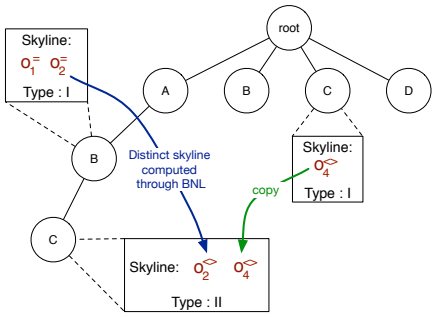



**Figure 4: The computation of subspace $\{A, B, C\}$.**

| Data set | Objects | Dimensions |
|---|---|---|
| NBA | 20493 | 17 |
| MBL | 92797 | 18 |
| IPUMS | 75836 | 10 |

**Table 5: Summary of real-world data sets.**

*skyline points from the parent node $\{A, B\}$ (no incomparable point) and from node $\{C\}$ (point $o_4$). The other possible skyline points, in this case, $o_2$, are then computed using a classical domination test, as shown in Figure 4. Because $SKY(\{A, B, C\}) \neq SKY(\{A, B, C, D\})$, the algorithm detects that $\{A, B, C\}$ is a closed subspace of a new equivalence class. The subspace is added to the list $\mathcal{LC}$.*

*In our running example, only* 7 *nodes are needed to represent the entire skycube and only* 9 *skyline points out of* 28 *need to be computed using domination tests, the rest being inferred. This demonstrates the saving in* Orion-clos.

## 7. EXPERIMENTS

In this section, we report an extensive experimental evaluation of our algorithms, using real data sets. All the algorithms were implemented in C/C++ and the experiments were performed on an 8-core Intel Xeon CPU $3.00GHz$ machine running Linux 2.6.31 operating system.

We compared the performance of our three algorithms with the state-of-the-art skycube algorithms: *Stellar* [7] and *Skyey* [9]. The real data sets used include the NBA table, which represents NBA's players statistics, the MLB's baseball players statistics table, and a sample from the IPUMS USA 2008 census data representing different households attributes. These data sets are summarized in Table 5.

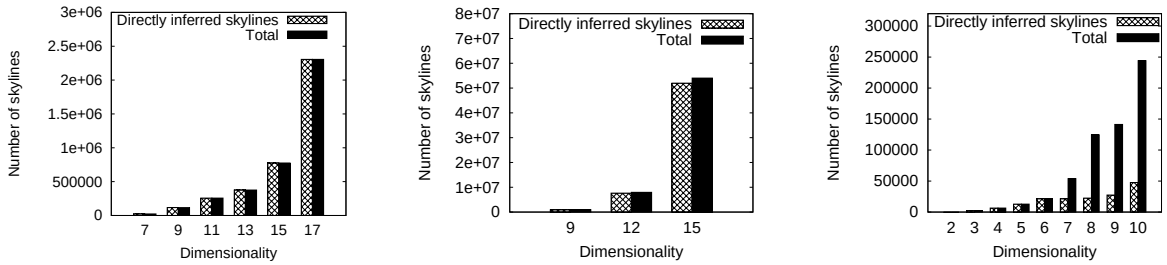
**Scalability**. Figure 6 shows the scalability with respect to dimensionality. *Orion*, *Orion-tail* and *Orion-clos* are always faster than *Skyey* and *Stellar* on the three data sets. *Skyey* and *Stellar* are unable to complete the skycube computation for the MBL and IPUMS data sets when the dimensionality are greater than six. We terminated the programs after 48 hours.

**Skyline derivations.** One major contribution in this paper is our new techniques to infer skylines without domination tests. Figure 7 shows the number of skylines derived using our rules against the total number of skylines. Our derivation rules are very efficient. For instance, all the skylines in the NBA data set can be derived by the *type I* rule and the *incomparable skylines* rule. This explains in part the very short runtime of our methods on the 17-dimensional table. The similar observation is obtained from the MBL data set. In such cases, very few dominance tests are required during the skycube construction. However, in the IPUMS data set, only a small percentage of the skylines can be inferred directly, which explains the longer runtime by the *Orion* algorithms.

**Skycube concise representation.** Figure 7 compares the number of equivalent classes ("closed nodes" in the figure) in our methods against the number of skylines. The number of equivalent classes is dramatically less than the number of skylines. This clearly shows the effectiveness of the skyline concise representation. We also plot the number of subspace skylines processed in our methods. For instance, instead of computing the $2^{17}$ possible subspaces in the NBA data set, our methods need to process only $26,924$ subspace skylines, store only $5,304$ closed subspaces as the concise representation of the whole skycube. In other words, only 4% of the subspace skylines are needed to present a lossless compression of the skycube. This explains the efficiency of our methods in both space and time.

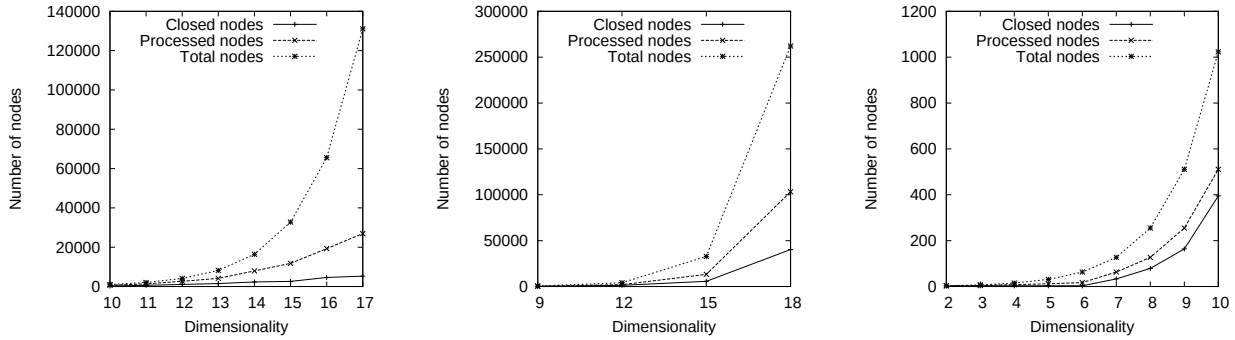
## 8. CONCLUSIONS

In this paper, we proposed new efficient methods to compute skyline cubes. Our approaches significantly reduce domination tests for subspaces and the number of subspaces searched. Theoretically, we studied two types of skyline




(a) The NBA data set (b) The MBL data set (c) The IPUMS data set

**Figure 6: Skylines inferences w.r.t. dimensionality.**




(a) The NBA data set (b) The MBL data set (c) The IPUMS data set

**Figure 7: The number of equivalent classes w.r.t. dimensionality.**

points that can be easily and directly derived without any domination tests. Moreover, based on formal concept analysis, we introduced closure operators that enable a concise representation for skyline cubes. Our empirical study results clearly demonstrated the merits of our approach in efficiency, scalability and conciseness.

# 9. REFERENCES


[1] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the average number of maxima in a set of vectors and applications. *J. ACM*, 25(4):536–543, 1978.

[2] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE'01*, pages 421–430. IEEE Computer Society.

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

[4] B. Ganter and R. Wille. *Applied Lattice Theory: Formal Concept Analysis*. 1997.

[5] 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.

[6] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. *ICDT*, 1540:398–416, 1999.

[7] J. Pei, A. W.-C. Fu, X. Lin, and H. Wang. Computing compressed multidimensional skyline cubes efficiently. In *ICDE*, pages 96–105. IEEE, 2007.

[8] J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.

[9] J. Pei, W. Jin, M. Ester, and Y. Tao. Catching the best views of skyline: a semantic approach based on decisive subspaces. In *VLDB*, pages 253–264. VLDB Endowment, 2005.

[10] J. Pei, Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, and Q. Zhang. Towards multidimensional subspace skyline analysis. *ACM Trans. Database Syst.*, 31(4):1335–1381, 2006.

[11] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *VLDB*, pages 301–310, 2001.

[12] Y. Tao, X. Xiao, and J. Pei. Subsky: Efficient computation of skylines in subspaces. In *ICDE*, 2006. IEEE Computer Society.

[13] J. Wang, J. Han, and J. Pei. Closet+: searching for the best strategies for mining frequent closed itemsets. In *KDD*, pages 236–245. ACM, 2003.

[14] T. Xia and D. Zhang. Refreshing the sky: the compressed skycube with efficient support for frequent updates. In *SIGMOD*, pages 491–502, 2006. ACM.

[15] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang. Efficient computation of the skyline cube. In *VLDB*, pages 241–252. VLDB Endowment, 2005.

# APPENDIX

## A. MATHEMATICAL PROOFS

PROOF OF THEOREM 1. (By contradiction). Assume $p \in SKY(\mathcal{U})$ and $p \in SKY(\mathcal{V})$, but $p \notin SKY(\mathcal{U} \cup \mathcal{V})$. According to the definition of domination, there exists a point $p'$ such that $\forall d_i \in \mathcal{U} \bigcup \mathcal{V}, p'(d_i) \leq p(d_i)$ and there exists at least a dimension $d_j \in \mathcal{U} \bigcup \mathcal{V}$ such that $p'(d_j) < p(d_j)$. Thus, $\forall d_k \in \mathcal{U}, p'(d_k) \leq p(d_k)$. This leads to a contradiction to the assumption $p \in SKY(\mathcal{U})$. □

PROOF OF THEOREM 2. (By contradiction). Let $\mathcal{W} = \mathcal{U} \cup \mathcal{V}$. Suppose that $SKY(\mathcal{U}) \cap SKY(\mathcal{V}) \neq \emptyset$ and $\mathcal{W}$ is a type II subspace. By Corollary 1, there exist at least two points $p, q$ in the skyline set of subspace $\mathcal{W}$ such that $p \prec\succ_{\mathcal{W}} q$ and they are also elements of the skyline set of subspaces $\mathcal{U}$ and $\mathcal{V}$. This leads to a contradiction to the assumption that $\mathcal{U}$ and $\mathcal{V}$ are type I subspaces such that $p =_{\mathcal{U}} q$ and $p =_{\mathcal{V}} q$ hold. □

PROOF OF THEOREM 3. (By contradiction) Assume that point $p$ is an incomparable skyline in subspace $\mathcal{U}$, but in a subspace $\mathcal{V}$ such that $\mathcal{U} \subset \mathcal{V}$, $p \notin SKY(\mathcal{V})$.

By definition of skyline domination, there exists a point $p'$ such that $\forall d_i \in \mathcal{V}, p'(d_i) \leq p(d_i)$ and there exists at least a dimension $d_j \in \mathcal{V}$ such that $p'(d_j) < p(d_j)$. Two cases arise:

1. $\exists d_k \in \mathcal{U}, p'(d_k) < p(d_k)$. This leads to a contradiction to the assumption that $p \in SKY(\mathcal{U})$.
2. $\forall d_k \in \mathcal{U}, p'(d_k) = p(d_k)$, and $\exists d_l \in \mathcal{V} \setminus \mathcal{U}, p'(d_l) < p(d_l)$. This leads to a contradiction to the assumption that $p$ is an incomparable skyline in $\mathcal{U}$.

□

PROOF OF THEOREM 4. The only-if direction is obvious. We prove by contradiction the if direction.

Assume that there is a point $q \prec_{\mathcal{W}} p$ so that $p \notin SKY(\mathcal{W})$. Since $p \in SKY(\mathcal{U})$, $q(d_i) \leq p(d_i)$ for any dimension $d_i \in \mathcal{U}$. Since $q \prec_{\mathcal{W}} p$, $q(d_i) \geq p(d_i)$ for any dimension $d_i \in \mathcal{U}$. Thus, $q(d_i) = p(d_i)$ for any dimension $d_i \in \mathcal{U}$, that is, $q =_{\mathcal{U}} p$. Since $q \prec_{\mathcal{W}} p$, $q(d_j) \geq p(d_j)$ for any dimension $d_j \in \mathcal{W} - \mathcal{U}$, and there must be one dimension $d \in \mathcal{W} - \mathcal{U}$ such that $q(d) > p(d)$. In other words, $p \succ_{\mathcal{W}-\mathcal{U}}$, a contradiction to the condition $p \not\succ_{\mathcal{W}-\mathcal{U}} q$. □

PROOF OF THEOREM 5. Consider a point $p$ to be incomparable over a subspace $\mathcal{U}$, by definition, this means that $\forall o_i \in SKY(\mathcal{U})$ $p \prec\succ_{\mathcal{U}} o_i$, hence the following two cases need not happen:

1. $p$ must not be smaller than the skyline points minimal value over any dimension $d_i$ (because in this case it will dominate at least a skyline point $q$, $p(d_i) < q(d_i)$)
2. $p$ must not be bigger than the skyline points maximal value over any dimension $d_i$ (because in this case it will be dominated by at least a skyline point $q$, $q(d_i) < p(d_i)$).

By combining these two conditions, the result follows. □

PROOF OF PROPERTY 1. We only need to prove that the following properties hold. For a set of objects $O \subseteq \mathcal{O}$ and a set of subspaces $M \subseteq 2^{\mathcal{D}}$, $\alpha$ and $\beta$ are both monotonic. Moreover, for a set of objects $O_1 \subseteq \mathcal{O}$ and a set of subspaces $M_1 \subseteq 2^{\mathcal{D}}$,

$$O_1 \subseteq \beta(\alpha(O_1)), M_1 \subseteq \alpha(\beta(M_1)) \tag{4}$$

Let $O_1, O_2 \subseteq \mathcal{O}$ be two sets of objects such that $O_1 \subseteq O_2$, and $M_1, M_2 \subseteq \mathcal{P}(\mathcal{D})$ be two sets of subspaces such that $M_1 \subseteq M_2$.

1. (Monotonicity). For any $\mathcal{U} \in \alpha(O_2)$ and $o_2 \in O_2$, $o_2 \in SKY(\mathcal{U})$. Since $O_1 \subseteq O_2$, for any $o_1 \in O_1$, $o_1 \in SKY(\mathcal{U})$. Thus, every $\mathcal{U} \in \alpha(O_2)$ is also an element of $\alpha(O_1)$, which means $\alpha(O_2) \subseteq \alpha(O_1)$.
   According to Equation (2), since $M_1 \subseteq M_2$, $\beta(M_2) = \beta(M_1) \bigcap \beta(M_2 \setminus M_1)$. Thus, $\beta(M_2) \subseteq \beta(M_1)$.
2. $\forall o \in O_1$, we have $o \in SKY(\mathcal{U})$ for every subspace $\mathcal{U} \in \alpha(O_1)$. Thus, $O_1 \subseteq \beta(\alpha(O_1))$. The same reasoning applies to the dual in Equation 4.

□

PROOF OF THEOREM 6. The theorem proof follows directly from the properties of the Galois connections.

- Monotonicity: $M \subseteq M' \Rightarrow h_*(M) \subseteq h_*(M')$.
- Idempotency: $h_*(h_*(M)) = h_*(M)$.
- Extensivity: $M \subseteq h_*(M)$.

□

## B. ALGORITHM PSEUDO-CODE

**Algorithm 1**: *Orion* algorithm and its variant *Orion-tail* based on derivation rules

**Data**: Data set $\mathcal{S}$ defined on space $\mathcal{D}$
**Result**: The entire skycube
**begin**
  $\mathcal{O} \leftarrow$ All Objects of data set $\mathcal{S}$;
  **if** Orion-tail *variant* **then**
    Generate B+-Trees for each dimension;
  **while** $\mathcal{T} \leftarrow SubspaceGeneration() \neq \emptyset$ **do**
    **foreach** *node representing subspace* $\mathcal{W} \in \mathcal{T}$ **do**
      **if** $\mathcal{W}$*.father is a type I* **then**
        `/* Theorem 2 derivation rule */`
        Choose $\mathcal{U}, \mathcal{V} \subset \mathcal{W}$ s.t. $|\mathcal{U}| = |\mathcal{V}| = |\mathcal{W}| - 1$;
        $SKY(\mathcal{W}) \leftarrow SKY(\mathcal{U}) \cap SKY(\mathcal{V})$;
        **if** $SKY(\mathcal{W}) \neq \emptyset$ **then**
          Set $\mathcal{W}$ as a type I subspace;
        **else**
          Set $\mathcal{W}$ as a type II subspace;
      **if** $\mathcal{W}$ *is a type II subspace* **then**
        **foreach** $\mathcal{U} \subset \mathcal{W}$ *s.t.* $|\mathcal{U}| = |\mathcal{W}| - 1$ **do**
          `/* Incomparable skylines derivation rule */`
          Copy all incomparable skylines from $\mathcal{U}$;
          `/* Process the indistinct skylines based on dominations tests */`
          BNL(indistinct skylines of subspace $\mathcal{U}$);
        `/* Process the new incomparable skylines */`
        **if** Orion-tail *variant* **then**
          $\mathcal{L} \leftarrow$ Process entailment candidates;
          $SKY(\mathcal{W}) \leftarrow BNL(\mathcal{L})$;
        **else**
          $SKY(\mathcal{W}) \leftarrow BNL(\mathcal{O} - SKY(\mathcal{W}))$;
  return $SKY(\mathcal{W})$;
**end**

**Algorithm 2**: The Orion-clos algorithm

**Data**: A data set $\mathcal{D}$
**Result**: The collection $\mathcal{LC}$ of closed subspaces sets
$\mathcal{LC} \leftarrow \emptyset$;
Generate first level of prefix-tree $P$;
Compute the fullspace skyline;
**foreach** *root's child node p from prefix-tree* $P$ **do**
  `DepthFirstGeneration`$(p, \mathcal{LC})$;
**end**
return $\mathcal{LC}$;

**Procedure** `DepthFirstGeneration`

**Data**: A node $p$, the collection of closed subspaces sets $\mathcal{LC}$

**foreach** *child node* $n$ *of* $p$ **do**
  **if** $n$*.father is of type I* **then**
    Do intersection on parents nodes;
    Adjust type (I or II) for n;
  **end**
  **if** $n$ *is of type II* **then**
    *Copy incomparable skylines from parent nodes*;
    *Find non-distinct skylines*;
    *Complete with BNL over remaining points*;
  **end**
  **if** $n$ *is covered by a closed subspaces set in* $\mathcal{LC}$ **then**
    **if** $\mathcal{D} \setminus \bigcup\{$ *all subspaces* $\mathcal{U}$ *in* $\mathcal{LC}$ *s.t.* $\mathcal{W} \subset \mathcal{C}\} = \mathcal{B} = \emptyset$ **then**
      Delete $n$ from $P$;
  **end**
  **else**
    `/* Develop only on non-covered subbranches` $\mathcal{B}$ `of` $n$ `*/`
    `DepthFirstGeneration`$(n, \mathcal{C})$;
    **if** $n$ *is representing or part of a closed subspaces set* **then**
      Update $\mathcal{LC}$ with this new subspace;
  **end**
**end**