Computing Skyline Incrementally in Response to Online Preference Modification

Tassadit Bouadi[1], Marie-Odile Cordier[1], and René Quiniou[2]

[1] IRISA - University of Rennes 1
[2] IRISA - INRIA Rennes
Campus de Beaulieu, 35042 RENNES, France
{tassadit.bouadi,marie-odile.cordier}@irisa.fr,
rene.quiniou@inria.fr

Abstract. Skyline queries retrieve the most interesting objects from a database with respect to multi-dimensional preferences. Identifying and extracting the relevant data corresponding to multiple criteria provided by users remains a difficult task, especially when the dataset is large. EC^2Sky, our proposal, focuses on how to answer efficiently skyline queries in the presence of dynamic user preferences and despite large volumes of data. In 2008-2009, Wong et al. showed that the skyline associated with any preference on a particular dimension can be computed, without domination tests, from the skyline points associated with first order preferences on that same dimension. Consequently, they propose to materialize skyline points associated with the most preferred values in a specific data structure called *IPO-tree* (*Implicit Preference Order Tree*). However, the size of the *IPO-tree* is exponential with respect to the number of dimensions. While reusing the *merging property* proposed by Wong et al. to deal with the refinements of preferences on a single dimension, we propose an incremental method for calculating the skyline points related to several dimensions associated with dynamic preferences. For this purpose, a materialization of linear size which allows a great flexibility for dimension preference updates is defined. This contribution improves notably the execution time and storage size of queries. Experiments on synthetic data highlight the relevance of EC^2Sky compared to *IPO-Tree*.

1 Introduction

Skyline queries represent a powerful tool for decision-making. Such queries aim at retrieving the most interesting objects from a database with respect to given criteria. In a multidimensional space where the dimension domains are ordered, skyline queries return the points which are not dominated by any other point. A point p dominates a point q if p is strictly better than q on at least one dimension and p is better or equal than q on the remaining dimensions. Identifying and extracting relevant data is often a difficult task especially when dealing with large volumes of data that can be compared according to many criteria. Several studies [15, 20, 22, 24, 18, 11, 10, 4] were carried out on skyline analysis as

A. Hameurlain et al. (Eds.): TLDKS X, LNCS 8220, pp. 34–59, 2013.

a retrieval tool in a decisional context. Skyline queries can formulate multi-criteria queries [16] and obtain the top answers, for example to find the cheapest hotels close to the beach. In the 60s, the search for skyline points was known as the problem of finding admissible points [3] or maximum vectors [2] or Pareto sets. These algorithms have been proved to be ineffective in the case of large databases with many points and many dimensions. Moreover, most of the work mentioned above assume that there exists a predefined order on the domain of each dimension.

An interesting problem arises when users are allowed to define or to change their own preferences online. Thus, on some dimensions the order may change dynamically. This problem is challenging when there are many data points in the dataset and it has attracted the attention of recent work [20, 22]. In fact, the skyline evolves when the preferences change. A naive solution is to recalculate the skyline from scratch for each dynamic preference that has changed. However, it is too expensive on large databases of high dimensionality. The challenge is thus the following: how to efficiently recalculate the least amount of skyline points while minimizing the required memory space.

The solutions proposed by Wong et al. in [22, 20] develop semi-materialization methods to support online query answering for skyline queries involving dynamic preferences. Precisely, the authors of [20] introduced the concept of *n-th order preference*. They showed that the skyline associated with any preference on a particular dimension can be computed from first order preferences on that same dimension. Relying on this *merging property*, they propose to materialize the skyline associated with first order preferences in a specific data structure, called *IPO-tree*, to speed up online query computations. However, to cope with the multiple dimension case they propose to store every possible combination of first-order preferences in an IPO-tree and, so, the size of an IPO-tree is in $O(c^m)$ where m is the number of dimensions with dynamic preferences and c the cardinality of a dimension. In the context of large volume of multidimensional data, this can be intractable. In [22], Wong et al. propose another structure, called *CST* (*Compressed ordered Skyline Tree*), to materialize all possible preference orders. However, this method turns to be incomplete and very complex and, so, it cannot be used. An erratum will be published in [21] for this issue.

The *merging property* of Wong et al. works on one dimension at a time and, thus, is of limited interest. We investigate the case of an arbitrary number of dimensions. Our proposition, EC^2Sky, focuses on how to answer efficiently skyline queries in the presence of several dynamic user preferences despite of large volume of data. This work improves and extends the contribution presented in [5]. Indeed, we extend the related work, introduce three new algorithms related to the EC^2Sky approach, prove the consistency and completeness of the EC^2Sky theorem, and conduct new extensive experiments on large-scale datasets with higher cardinality dimensions and higher dimensional space than the latest experiments presented in [5].

The main idea relies on the incremental addition of dynamic dimensions when computing the skyline. As a side effect, EC^2Sky can return the most relevant

knowledge by emphasizing the compromises associated with the specified preferences. The benefits of this proposition are twofold. On the one hand, complexity in space of the materialization of precomputed skyline reduces to $O(c*m)$ where m is the number of dimensions and c is the size of dimension domains. On the other hand, the number of dominance tests decreases significantly. Some extra memory space and additional runtime is needed to compute skyline related to first-order preferences with respect to Wong et al.'s method. But we proved experimentally that the total computation cost is much lower than in Wong et al.'s method. This contribution enables an incremental computation of skyline points associated with a set of preferences as well as make the interactive modification of preferences easier.

The rest of the paper is organized as follows. In Section 2, we discuss related work on the problem of searching skyline points in the presence of dynamic preferences. In Section 3, we introduce the basic concepts related to skyline queries and dynamic preferences. We develop the formal aspects of our new approach EC^2Sky in Section 4 and present its implementation in Section 5. In Section 6, we present the results of the experimental evaluation performed on synthetic datasets and highlight the relevance of the proposed solution by comparing it to references in the field. We conclude the paper in Section 7.

2 Related Work

Much work has been done on skyline computation. Börzsönyi et al. [4] first investigate and introduce the skyline computation problem in the context of databases, and also propose a block-nested loops (BNL) method and a divide-and-conquer method. Since this pioneering work, many algorithms have been developed for efficient skyline computation. Especially, index-based techniques were proposed in several works to further accelerate skyline queries. The first skyline algorithm incorporated with B-tree or R-tree indexes is proposed in [4]. Then, two progressive processing methods, Bitmap and Index, were proposed in [17]. Skyline queries have been studied in different computational environments, such as full-space skyline retrieval [8, 9, 13, 23] which computes the skyline in a space of a fixed dimensionality, or subspace skyline retrieval [15, 14, 10, 11, 24] where different users may issue skyline queries regarding different subspaces of different dimensions. Skyline queries have also been investigated with various constraints as ranking skyline [18, 6] that enables the user to retrieve the top-K skyline points instead the whole skyline, or metric skyline [7] which retrieves skyline points with dynamic dimensions in the metric space. There have been a considerable number of methods on skyline computation in the context of databases with totally ordered dimensions. However, in real applications, data may include dimensions that are partially ordered in nature, such as categorical (i.e. nominal) dimensions, etc. Some recent studies consider partially ordered dimensions on skyline computation [1], and provide a way to verify dominance among incomparable points over the partial order.

However, most of the work mentioned above assume that there exists only one predefined order on the domain of each dimension. In particular, users cannot express online preferences between dimensions nor customize the preferences between the elements of a given dimension and search the skyline points associated with these preferences. Therefore, other types of skyline queries have been proposed to handle:

- inter-dimension preferences allowing the user to specify the importance of various dimensions and thus to rank the skyline by order of preferences [12]. For example, the price can be considered more important than the distance to the beach. Instead of returning the whole set of skyline points resulting from a query, only the best K points, the top K [6], with respect to the defined inter-dimensions preferences are returned,
- intra-dimension preferences used in [20, 22], allowing each user to express preferences on the different values of a dimension. This kind of preferences is especially attractive for nominal dimensions where there is no evidence of consensual ordering. For example, a user may prefer an hotel of the group *Tulips* to those of the group *Horizon* while another user prefers hotels of the group *Mozilla* to all others.

In this paper, we are particularly interested in skyline queries with dynamic intra-dimension preferences. A naive solution would be to enumerate all possible combinations of preferences and to store the associated skyline. However, the preprocessing burden and the storage induced by a complete materialization of these preferences are prohibitively expensive on large databases. Therefore, other materialization methods have been proposed to support online query answering with dynamic intra-dimensions preferences.

Wong et al. [20] proposed a semi-materialization method based on a specific data structure called IPO-tree (*Implicit Preference Order Tree*). An IPO-tree stores partial useful results corresponding to every combination of first order preferences. A first order preference states that one value is most preferred in some dimension and that the other values are left unordered. An *n-th* order preference specifies an order over n values from some dimension, whereas the other values are less preferred and left unordered. Wong et al. also introduced a property called the *merging property* which makes possible to derive skyline of any n order preference by simple operations on the first order preferences on the same dimension. However, this approach has a main drawback. The *merging property* is applicable to only one dimension at a time. The size of an IPO tree is thus in $O(c^m)$ (where m is the number of dimensions associated with dynamic preferences and c is the cardinality of a dimension). In the context of large databases of high dimensionality this structure becomes very complex.

In [22], Wong et al. proposed another structure, called CST (*Compressed Ordered skyline Tree*), to materialize all the preference orders. They store the skyline with respect to various refinement orders in a compact data structure. However, a CST tree is very complex. This makes updating preferences a difficult task that requires extensive maintenance and changes in the CST tree.

Also, the method is not complete. In fact, the CST tree is constructed gradually by adding dimensions associated with dynamic preferences one by one. During this construction, some points are disqualified from the skyline when adding a new dimension, while they should be in the skyline. An erratum will be published in [21] for this issue.

While reusing the *merging property* proposed by Wong et al. to deal with the refinements of preferences on a single dimension, we propose an incremental method, named EC^{2}Sky, for calculating the skyline points related to several dimensions associated with dynamic preferences. Indeed, the skyline may change due to dimension preference updates, and hence should be incrementally maintained to avoid re-evaluation from scratch. Unlike the IPO-tree method, the EC^{2}Sky structure facilitates and allows a great flexibility for updating dimensions and dynamic preferences.

3 Basic Concepts

In this section, we present the necessary concepts and definitions related to skyline queries. Many are borrowed from [20, 22]. The notations are summarized in Table 1.

The various definitions are illustrated using the example of Table 2 which describes proposals for travels according to the dimensions Price, Distance from the beach, Hotel group (Gr) and Airline (Air).

Table 1. Summary of notations

Notation	Description
E	Dataset
$\|E\|$	Cardinality of E
$D = S \bigcup Z$	Data space of E
S	Subspace with static preferences
Z	Subspace with dynamic preferences
$\|D\|$	Cardinality of D
d_i	One dimension of D ($1 \leq i \leq \|D\|$)
D'	Subspace of D: $D' \subseteq D$
D^i	Subspace of D: $D^i = D^{i-1} \bigcup \{d_i\}$
$\mathcal{P}(E)$	Power set of E
p, q	Data points
$p(d_i)$	Value of p on dimension d_i
$dom(d_i)$	Dimensional domain of d_i
$\wp$	Preferences on Z
$\wp_i$	Preferences on d_i

Table 2. A set of hotels

Hotel ID	Price	Distance	Hotel group	Airline
a	1600	4	T (Tulips)	G(Gonna)
b	2400	1	T(Tulips)	G(Gonna)
c	3000	5	H(Horizon)	G(Gonna)
d	3600	4	H(Horizon)	R(Redish)
e	2300	2	T(Tulips)	R(Redish)
f	3000	3	M(Mozilla)	W(Wings)
g	3600	4	M(Mozilla)	R(Redish)
h	3000	3	M(Mozilla)	R(Redish)

Example 1. *$E = (a, b, c, d, e, f, g, h)$ is a dataset defined in a 4-dimensional space $D =$ (Price, Distance, Hotel group, Airline), $|E| = 8$, $|D| = 4$. The value of point p on dimension Price is denoted by $p(Price) = 1600$.*

Definition 1. ***(Preference order)*** *A preference order on the domain of a dimension d_i is defined by a partial order $\leq_{d_i}$. For two values $p(d_i)$ and $q(d_i)$ in the domain of d_i, we write $p(d_i) \leq_{d_i} q(d_i)$ if the value $p(d_i)$ is preferred to the value $q(d_i)$. We denote $p(d_i) <_{d_i} q(d_i)$ if $p(d_i) \leq_{d_i} q(d_i)$ and $q(d_i) \not\leq_{d_i} p(d_i)$,*

Example 2. *In Table 2, both dimensions, **Price** and **Distance**, are totally ordered by the relation $\leq_{d_i}$, which corresponds to the order relation $\leq$ indicating the smallest of two real numbers. The preference means that the lower the price and the distance, the more preferable a hotel. No order is given a priori on the dimensions Hotel group and Airline. It is up to users to express their own preferences among values belonging to these dimensions.*

Definition 2. ***(Preference type)*** *We distinguish two types of preferences:*

- *static preferences: they correspond to a predefined order relation,*
- *dynamic preferences: they correspond to an order relation that can vary from one user to another or from one user session to another.*

By abuse of language we use *dynamic (resp. static) dimension* instead of *dimension associated with dynamic (resp. static) preferences*. In the rest of this paper, we denote by S the subspace associated with static preferences and by Z the subspace associated with dynamic preferences, with $D = S \bigcup Z$ and $S \bigcap Z = \emptyset$.

Example 3. *$S = \{Price, Distance\}$: the values of these two dimensions follow the order relation $\leq$ specifying that the lower the price (resp. the distance), the more preferable the hotel (ex: $a(Price) <_{Price} d(Price)$). This order is accepted by any user, so it is static. For $Z = \{Gr, Air\}$ no order relation is defined a priori (i.e., in a static way) on these dimensions. The definition of an order is left to users and may vary from one user to another.*

Definition 3. ***(Dominance relation)*** *p dominates q on $D' \subseteq D$, denoted by $p \prec_{D'} q$, if p is preferred or equal to q on any dimension of D' and p is preferred to q on at least one dimension:*
$\forall d_i \in D'$, $p(d_i) \leq_{d_i} q(d_i) \wedge \exists d_i \in D'$, $p(d_i) <_{d_i} q(d_i)$.
$p =_{D'} q$ denotes the fact that p is equally preferred to q on any dimension of D'. When $D' = D$, $p \prec_{D'} q$ is simply noted $p \prec q$.

Example 4. *Let a customer looking for a hotel that is both close to the beach and affordable. In this case hotel a dominates hotel d ($a \prec_{(Price,Distance)} d$) since*
$a(Price) <_{Price} d(Price)$ *and*
$a(Distance) =_{Distance} d(Distance)$.
Hotel a does not dominate hotel b ($a \not\prec_{(Price,Distance)} b$) since
$b(Distance) <_{Distance} a(Distance)$.

The following definitions concern a subspace $D' \subseteq D$. Obviously, these definitions can be generalized to the full dimension space D. The skyline set, or simply the skyline, of a dataset on a subspace contains the points in the dataset that are not dominated by any other point in that dataset.

Definition 4. ***(Skyline)*** *The skyline set of the dataset E on the subspace D' with Z being the subspace associated with the dynamic preferences $\wp$ is the set of points that are not dominated by any point in E:*
$Sky(D', E)_{(Z,\wp)} = \{p \in E |\ \forall q \in E,\ q \not\prec_{D'} p\}$.

If $\wp = \emptyset$, $Sky(D', E)_{(Z,\wp)}$ is simply written $Sky(D', E)$.

Example 5. $Sky(\{Price, Distance\}, E) = \{a, b, e\}$.
$Sky(\{Price, Distance, Gr, Air\}, E) = \{a, b, e, c, d, f, h\}$ *when no preferences are given on dimensions Gr and Air. The points c, d, f and h are no longer dominated by the skyline points $\{a, b, e\}$ since no point can dominate on the unordered dimensions Gr and Air. The point g is dominated by the point h on the static dimensions and have the same values on the dynamic dimensions and so is not in the skyline.*

The set $Sky(D', E)$ contains points, denoted by $MaxSky(D', E)$, that are the best along at least one dimension. It also contains, and this is a major interest of this approach, points denoted $CompSky(D', E)$ that are not dominant on any dimension of D' while being better than any point of E on at least one dimension. These points represent interesting *compromise* solutions for the user from a decision making point of view.

Definition 5. ***(Partition of skyline sets)***
The skyline set of the subspace $D' \subseteq D$ can be decomposed into two sets of points, $MaxSky$ and $CompSky$.
$Sky(D', E) = MaxSky(D', E) \bigcup CompSky(D', E)$ *with :*

- $MaxSky(D', E) = \{p \in Sky(D', E) |\ \exists D'' \subseteq D',\ \forall q \in E,\ p \preceq_{D''} q\}$,
- $CompSky(D', E) = \{p \in Sky(D', E) |\ \forall q \in E,\ q \neq p,\ \exists D'' \subseteq D',\ p \prec_{D''} q\}$.

When D' is restricted to only one dimension ($D' = \{d_i\}$),
$Sky(D', E) = MaxSky(D', E)$.

Example 6. *Let $D' = \{Price, Distance\}$.*
$Sky(\{Price, Distance\}, E) = \{a, b, e\}$.
$MaxSky(\{Price, Distance\}, E) = \{a, b\}$ *since $a(Price)$ (resp. $b(Price)$) is the most preferred value on dimension Price (resp. Distance).*
$CompSky(\{Price, Distance\}, E) = \{e\}$ *since the value of e is not the most preferred either on Price or on Distance. However, $e(Price)$ is preferred to $b(Price)$ on dimension Price and $e(Distance)$ is preferred to $a(Distance)$ on dimension Distance. So, e is better than any $MaxSky$ point on at least one dimension.*

Various kinds of skyline queries can be formulated. Conventional skyline queries retrieve the most interesting objects of a multidimensional dataset. Our goal is to aid a user explore his dataset by letting him express various preferences on dynamic dimensions and assess the consequences of such choices by retrieving the most preferred points i.e. the skyline points. For example, let Hotel group be a dimension with dynamic preferences. Different users may have different preferences on that dimension. If a customer prefers the Hotel group $Horizon$ to all the other hotels, c, d, f and h are added in $Sky(\{Price, Distance, Gr\}, E)$ since they become the best along the Hotel group dimension. However, for another customer preferring $Tulips$ to all the others, c, d, f and h do not belong to the skyline since they are dominated by a, b and e. An interesting observation is that hotels associated with a, b and e are always in the skyline no matter which preference order on the Hotel group is chosen (because a is the only one with the best price, b is the only one with the best distance from the beach and e represent a compromise for dimensions Price and Distance).

When a user formulates a query involving a dimension d_i with dynamic preferences, she/he can specify the preference order on the $|d_i|$ values of this dimension. The order is total if all these values are ordered. But this is not always possible and the user may order only n of the $|d_i|$ values. Implicitly, she/he considers that they are more preferred than the $(|d_i| - n)$ remaining values which are left unordered. This corresponds to the notion of *n-th order implicit* preference introduced in [20].

Definition 6. ***(n-th order preference)*** *Let $d_i \in Z$ and $|d_i| = m$. $\wp_i$ is an n-th order preference on d_i iff :*

- $\wp_i = v_1 <_{d_i} \ldots <_{d_i} v_n <_{d_i} *$, *with* $v_1 \in dom(d_i), \ldots, v_n \in dom(d_i)$ *and* $n \leq m$,
- $\forall k \in \{n+1, .., m\}, v_n <_{d_i} v_k$.

When $n = 1$, $\wp_i = v_1 <_{d_i} *$ is called a first order preference.

Thus $<_{d_i}$ is a total order on the values $\{v_1 \ldots v_n\}$ of d_i, and a partial order on the whole dimensional domain of d_i.
Note the importance of first order preferences: they are sufficient to determinate the dominant points of a dimension.

Example 7. *For the dimension Hotel group in Table 2, a user prefers* $T(Tulips)$ *to* $M(Mozilla)$, T *to* $H(Horizon)$ *and* M *to* H *(i.e.,* $T <_{Gr} M <_{Gr} H$*). This preference is a third order preference and defines a total order. Some other user could prefer Hotel group* T *to any other group (i.e.,* $T <_{Gr} *$*). In this case, the preference is a first order preference which defines the partial order* $\{T <_{Gr} M, T <_{Gr} H\}$.

$\wp_i = v_1 <_{d_i} \ldots <_{d_i} v_n <_{d_i} *$ denotes the set of binary preferences $\wp_i = \{v_1 <_{d_i} v_2, v_2 <_{d_i} v_3, \ldots, v_n <_{d_i} *\}$. The absence of preference on dimension d_i is denoted by $\wp_i = \emptyset$. In the sequel, we use both notations for $\wp_i$.

Example 8. *Let* $Z = \{Gr\}$, $\wp = \{H <_{Gr} *\}$ *(equivalent to* $\{H <_{Gr} M, H <_{Gr} T\}$*).*
Then $Sky(D,E)_{(Z,H<_{Gr}*)} = \{a,b,e,c,d,f,h\}$*. The points c and d are no longer dominated by the skyline points* $\{a,b,e\}$ *since they have the best values on the dimension Gr. The values M and H of the dimension Gr are left unordered, so the points f and h become compromise skyline points because they are no longer dominated by the skyline points* $\{a,b,e\}$ *on the dimension Gr.*

We give below some useful properties of the preference relationship. These properties will be used later to reduce the number of domination tests during skyline computation. In the following, $\wp = \bigcup_{i=1}^{|Z|} \wp_i$ denotes the set of dynamic preferences associated with $Z \subseteq D$ combined implicitly with the set of static preferences associated with $S \subseteq D$.

Definition 7. ***(Preference inclusion)*** *Let* $|Z| = m$, $\wp = \{\wp_1, .., \wp_m\}$ *and* $\wp' = \{\wp'_1, .., \wp'_m\}$*, where every* $\wp_i$ *and* $\wp'_i$ *are sets of binary preferences associated with the dimension* $d_i \in Z$.
Then, $\wp \subseteq \wp'$ *if and only if* $\wp_i \subseteq \wp'_i$ *for* $1 \leq i \leq m$.

Definition 8. ***(Preference refinement)*** *Let* $\wp'$ *and* $\wp''$ *two preferences sets on the subspace* Z. $\wp''$ *is a refinement of* $\wp'$ *if* $\wp' \subseteq \wp''$.

Property 1. ***(Monotonicity of preference refinement)*** *Let* $\wp'$ *and* $\wp''$ *two preferences sets on* Z*. If* $\wp''$ *is a refinement of* $\wp'$ *then* $Sky(D,E)_{(Z,\wp'')} \subseteq Sky(D,E)_{(Z,\wp')}$.

The following example illustrates property 1.

Example 9. *Let* $Z = \{Gr\}$,
$\wp' = \{H <_{Gr} *\}$ *and* $\wp'' = \{H <_{Gr} T <_{Gr} *\}$.
$\wp''$ *is a refinement of* $\wp'$ *since* $\wp' \subset \wp''$.
$Sky(D,E)_{(Z,\wp')} = \{a,b,c,d,e,f,h\}$ *and* $Sky(D,E)_{(Z,\wp'')} = \{a,b,c,d,e\}$.
We have $Sky(D,E)_{(Z,\wp'')} \subset Sky(D,E)_{(Z,\wp')}$.

Property 1 indicates that when preferences are refined, the skyline may become smaller and so, some skyline points may be disqualified. Also, if a point is not in the skyline related to some preference, it won't belong to the skyline

related to a refined preference. We will use property 1 later to reduce the number of domination tests in our approach.

The following theorem formulates an important property called the *merging property* that was introduced by Wong et al. [20]. This property provides a means to derive the skyline related to any possible n-th order preference by operations on first order preferences on the same dimension.

Theorem 1. ***(Merging property)*** *Let $\wp^{'}$ and $\wp^{''}$ be two preferences differing only on dimension d_i, i.e. $\wp^{'}_j = \wp^{''}_j$ for all $j \neq i$. Let $\wp^{'}_i = v_1 <_{d_i} \ldots < v_{k-1} <_{d_i} *$ and $\wp^{''}_i = v_k <_{d_i} *$. Let $PSky(D,E)_{(Z,\wp')}$ be the set of points in $Sky(D,E)_{(Z,\wp')}$ with d_i values in $\{v_1 \ldots v_{k-1}\}$. Let $\wp^{'''}$ be a preference differing from $\wp^{'}$ and $\wp^{''}$ only on dimension d_i and $\wp^{'''}_i = v_1 < \ldots < v_{k-1} < v_k < *$. Then the skyline associated with $\wp^{'''}$ is:*

$$Sky(D,E)_{(Z,\wp''')} = (Sky(D,E)_{(Z,\wp')} \bigcap Sky(D,E)_{(Z,\wp'')}) \bigcup PSky(D,E)_{(Z,\wp')}.$$

Example 10. *Let* $\wp^{'} = \{M <_{Gr} *\}$, $\wp^{''} = \{H <_{Gr} *\}$,
$\wp^{'''} = \{M <_{Gr} H <_{Gr} *\}$ *and* $Z = \{Group\}$.
$Sky(D,E)_{(Z,\wp''')} = (Sky(D,E)_{(Z,\wp')} \bigcap Sky(D,E)_{(Z,\wp'')}) \bigcup PSky(D,E)_{(Z,\wp')}$
$= (\{a,b,e,f,h\} \bigcap \{a,b,e,c,d,f,h\}) \bigcup \{f,h\}$
$= \{a,b,e,f,h\}$

Wong et al. have proposed successively two interesting methods, IPO-tree [20] and CST [22], for skyline computation based on the properties and theorem 1 presented above. However, the implementation based on these two proposals raises several problems:

- The size of the IPO-Tree structure is in $O(c^m)$ where m is the number of dimensions with dynamic preferences and c the cardinality of a dimension. So it is intractable in the context of large databases with high dimensionality and does not allow scaling. It is worth-noting that the *merging property* is applicable to only one dimension at a time,
- The CST method does not solve the *IPO-Tree* problems because its algorithm is incomplete (it disqualifies points which should be in the skyline).

The second proposal (CST) is incomplete [21], thus we focus on the first proposal (IPO-Tree). The IPO-Tree method supports the refinement of preferences. However, it addresses the treatment of one dynamic dimension at a time (merging property). To cope with several dynamic dimensions, Wong et al. propose to store every combination of the first order preferences related to these dimensions. So, the size of the proposed materialization structure is exponential, which is prohibitive when dealing with several dimensions. We propose in Section 4 an incremental method which makes possible to introduce dynamic dimensions one by one. It relies on a structure that enables an effective materialization of dynamic preferences.

4 EC²Sky: An Incremental Skyline Computation

In this section we introduce the proposed incremental method and the theorem that grounds the method.

Let us examine how the addition of a dynamic dimension d_i impacts the skyline that was previously computed for the dimension subspace D^{i-1}. Intuitively, the computation of the new skyline for $D^i = D^{i-1} \bigcup d_i$ is a two-step process. First, compute the skyline associated to the new dynamic dimension union the static dimensions $d_i \bigcup S$, as if it were independent of the other dynamic dimensions. Second, take into account the correlations between the new dimension d_i and the previous dimensions D^{i-1} to update the new computed skyline. This second task consists in, i) removing from the skyline independently computed for $d_i \bigcup S$ the points that are disqualified i.e. are dominated on the dynamic dimensions of D^{i-1}, ii) removing the set of old skyline points that are disqualified i.e. are dominated on the new dimension d_i, iii) completing the resulting skyline with points that are new compromises for $D^{i-1} \bigcup d_i$.

In the following, we assume that the subset of dimensions D^i is such that $D^i = D^{i-1} \bigcup d_i$, with $d_i \in Z$, $i \in \{1, .., |Z|\}$, $D^i \subseteq D$ and $D^0 = S$. This notation represents the incremental addition of dimensions in skyline computation.

Consider the addition of a dynamic dimension d_i to a set D^{i-1} of $i-1$ dynamic dimensions. As sketched above, the first task is to compute $Sky(d_i \bigcup S, E)_{(Z,\wp)}$, the skyline related to dimension d_i as if it were independent of the other dynamic dimensions. Wong et al.'s method can be used to achieve this task. However, this set may contain skyline points that are disqualified i.e. they are dominated on the dynamic dimensions of the subspace D^{i-1}. Precisely, let $p, q \in Sky(d_i \bigcup S, E)_{(Z,\wp)}$ be two skyline points with the same values on every dimension of $d_i \bigcup S$. If q is preferred on D^{i-1} it will dominate p and disqualify it from the skyline $Sky(D^i, E)_{(Z,\wp)}$. This set of points is denoted $CutSky(d_i \bigcup S, E)$.

Definition 9. ***(Disqualified skyline points from*** $d_i \bigcup S$***)*** *The set of skyline points related to the subspace* $d_i \bigcup S$ *that are disqualified by the introduction of the subspace* D^{i-1} *is defined by* $CutSky(d_i \bigcup S, E) = \{p \in Sky(d_i \bigcup S, E)_{(Z,\wp)} \mid \exists q \in Sky(d_i \bigcup S, E)_{(Z,\wp)},\ p =_{d_i \bigcup S} q \wedge q \prec_{D^{i-1}} p\}$.

Example 11. *Let* $D^{i-1} = D^1 = \{Price, Distance, Air\}$, $d_i = Gr$, *the new dimension, and the preferences:*
$\{M <_{Gr} H <_{Gr} T\}$ *and* $\{W <_{Air} *\}$.
$Sky(D^1, E)_{(Z,\wp)} = \{a, b, e, f\}$ *and* $Sky(\{Gr\} \bigcup S, E)_{(Z,\wp)} = \{a, b, e, f, h\}$.
$CutSky(\{Gr\} \bigcup S, E) = \{h\}$ *as the point h should be removed from the skyline* $Sky(D^2, E)_{(Z,\wp)}$ *since it is dominated by the point f on the subspace* D^1.

On the other hand, the old skyline $Sky(D^{i-1}, E)_{(Z,\wp)}$may contain points that are disqualified by dominant points brought by the new dimension d_i. Precisely, let $p, q \in Sky(D^{i-1}, E)_{(Z,\wp)}$ be two skyline points with the same values on every

dimension of D^{i-1}. If q is preferred on the new dimension d_i, it will dominate p and disqualify it from the skyline $Sky(D^i, E)_{(Z,\wp)}$. This set of points is denoted $CutSky(D^{i-1}, E)$.

Definition 10. ***(Disqualified skyline points from*** D^{i-1}***)*** *The set of skyline points related to the subspace* D^{i-1} *that are disqualified by the introduction of dimension* d_i *is defined by*
$CutSky(D^{i-1}, E) =$
$\{p \in Sky(D^{i-1}, E)_{(Z,\wp)} \mid \exists q \in Sky(D^{i-1}, E)_{(Z,\wp)}, p =_{D^{i-1}} q \wedge q \prec_{d_i \bigcup S} p\}$.

Example 12. *Let* $D^{i-1} = D^1 = \{Price, Distance, Gr\}$, $d_i = Air$, *the new dimension, and the preferences* $\{M <_{Gr} H <_{Gr} T\}$ *and* $\{G <_{Air} R <_{Air} W\}$.
$Sky(D^1, E)_{(Z,\wp)} = \{a, b, e, f, h\}$, $CutSky(\{Air\} \bigcup S, E) = \{\}$ *and*
$CutSky(D^1, E) = \{f\}$ *as the point* f *should be removed from the skyline*
$Sky(D^2, E)_{(Z,\wp)}$ *since it is dominated by the point* h *on the new dimension* $d_i = Air$.

Finally, some new points should appear in the new skyline. Precisely, before taking into account the new dimension d_i, some points may be dominated on every dimension of $D^{i-1} \bigcup S$ and, so, are not in the skyline. But, when dimension d_i is introduced, being better on d_i than some skyline points they were dominated by, they may well be no longer dominated by any skyline point from $Sky(D^{i-1}, E)_{(Z,\wp)}$ on some dimensions from D^{i-1}: they are new compromise skyline points. This set of points is denoted $NewCompSky(D^i, E)$.

Definition 11. ***(New compromise skyline)***
Let $C = Sky(D^{i-1}, E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S, E)_{(Z,\wp)}$.
The set of new compromise skyline points is defined by
$NewCompSky(D^i, E) = \{p \in E - C \mid \forall q \in E, \exists d_k \in D^i, p \prec_{d_k} q\}$.

Example 13. *Let* $D^{i-1} = D^1 = \{Price, Distance, Gr\}$, $d_i = Air$, *the new dimension, and the preferences:*
$\{M <_{Gr} H <_{Gr} T\}$ *and* $\{G <_{Air} R <_{Air} W\}$.
$Sky(D^1, E)_{(Z,\wp)} = \{a, b, e, f, h\}$, $Sky(\{Air\} \bigcup S, E)_{(Z,\wp)} = \{a, b, e\}$,
$CutSky(\{Air\} \bigcup S, E) = \{\}$ *and* $CutSky(D^1, E) = \{f\}$. *But, if we consider simultaneously the two dimensions* Gr *and* Air *then* c *is no longer dominated by* f. *As* f *was the only point* c *was dominated by,* c *becomes a new skyline point. Since* c *is the only such "promoted" point,* $NewCompSky(D^2, E) = \{c\}$.

Suppose we want to extend a dimensional subspace D^{i-1} with a new dimension d_i. The following theorem states that the skyline of the extended subspace can be computed by removing disqualified skyline points from the old skyline and by adding the new skyline points brought by the preference on the new dimension. The new skyline points are either dominant points on the new dimension or new compromise skyline points introduced by the new preference.

Theorem 2. ***(Incremental skyline)***
Let E *be a* $|D|$*-dimensional dataset,* $Z \subseteq D$ *the subspace of size* $|Z| = m$ *with*

dynamic preferences $\wp = \{\wp_j\}_{j=1,..,m}$ *on* D, $Sky(d_i \bigcup S, E)$ *the skyline of the subspace* $S \bigcup d_i$ *and* $D^i = D^{i-1} \bigcup d_i$, *with* $i = \{1, .., m\}$.

$Sky(D^i, E)_{(Z,\wp)} = (Sky(D^{i-1}, E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S, E)_{(Z,\wp)}) - (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S)) \bigcup NewCompSky(D^i, E)$.

Example 14. ***(Illustration of Theorem 2)*** *Let* $D^1 = \{Price, Distance, Gr\}$, $D^2 = \{Price, Distance, Gr, Air\}$ *and we consider the preferences of the previous example.*
$Sky(D^2, E)_{(Z,\wp)} =$
$(Sky(D^1, E)_{(Gr, H<_{Gr}M<_{Gr}T)} \bigcup Sky(\{Air\} \bigcup S, E)_{(Air, G<_{Air}W<_{Air}R)})$
$-((CutSky(D^1, E) \bigcup CutSky(\{Air\} \bigcup S, E)) \bigcup NewCompSky(D^2, E) =$
$(\{a, b, e, f, h\} \bigcup \{a, b, e\})$ - $(\{f\} \bigcup \{\}) \bigcup \{c\} =$
$\{a, b, c, e, h\}$.

Proof. **(Theorem 2)** Here follows a sketch of the proof of theorem 2. We consider successively the points that are disqualified from the skyline and the points that are added to the skyline. Let $D^i \subseteq D$ and $D^i = D^{i-1} \bigcup \{d_i\}$. Let Z be the subspace of D with dynamic preferences and $Sky(D^i, E)$ be the skyline of the subspace D^i.

- Any element of $CutSky$ must be disqualified from the resulting skyline.
  If $p \in (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S))$ then there exists some $q \in Sky(D^i, E)$ such that
  $(q =_{D^{i-1}} p$ and $q \prec_{d_i \bigcup S} p)$ or $(q =_{d_i \bigcup S} p$ and $q \prec_{D^{i-1}} p)$.
  This means that $q \prec_{D^i} p$. Thus, p should not be in $Sky(D^i, E)_{(Z,\wp)}$.
- Any element of $Sky(D^i, E)_{(Z,\wp)}$ must belong to
  $\{ (Sky(D^{i-1}, E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S, E)_{(Z,\wp)}) - (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S)) \bigcup NewCompSky(D^i, E)\}$.
  Any $p \in Sky(D^i, E)_{(Z,\wp)}$ is such that:
  - either there exists a dimension $d_j \in D^i$ such that $\forall q \in E$, $p \preceq_{d_j} q$. In this case, $p \in (Sky(D^{i-1}, E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S, E)_{(Z,\wp)})$
  - or for every $q \in E$, there exists a $d_i \in Z$ such that $p \prec_{d_i} q$. In this case, $p \in NewCompSky(D^i, E)$ and $p \notin (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S)$

  In both cases p belongs to $Sky(D^i, E)_{(Z,\wp)}$ □

Theorem 2 provides a scheme for an incremental computation of skyline queries associated with several dynamic dimensions. In the following, we describe more precisely our proposal.

5 EC²Sky Implementation

In this section, we present the implementation of incremental skyline computation. We introduce some definitions to characterize the points that are involved

in the incremental computation of skyline points and facilitate the specification of the algorithms and of the materialization structure. To ensure an efficient and online computation of skyline, we provide an effective materialization structure detailed in the sequel. We propose a trade-off between (i) *materialize* all the skyline points for all possible preferences and (ii) *calculate*, for each user query, the skyline points associated with the preferences formulated in the query. Our approach is based on three steps:

1. compute and store the skyline on static dimensions. One can adopt any existing algorithm (e.g. [1]) that computes the skyline for partially ordered domains;
2. for each dimension with dynamic preferences, compute and store the candidate skyline points according to any possible first order preference;
3. rely on the information stored in step 1 and 2 to compute the skyline points related to user preferences on incrementally introduced dynamic dimensions.

5.1 Skyline Associated with Static Dimensions

In step 1 we compute all the skyline points corresponding to the defined static preferences of D. Two concepts introduced by Wong et al. in [22] are helpful. They decompose the set $Sky(D, E)$, corresponding to the defined static preferences of D and denoted by $\wp_\emptyset$, into two subsets: the *global skyline set GSky(D,E)* and the *order-sensitive skyline set OsSky(D,E)*.

The points in the global skyline set $GSky(D, E)$ remain in the skyline whenever any preference on any dimension of Z is added.

Definition 12. ***(Global skyline points)*** *The global skyline set of the space $D = S \bigcup Z$ on the dataset E, is defined by*
$GSky(D, E) =$
$\{p \in Sky(D, E) \mid \forall\ q \in Sky(D, E), \nexists\ d_i \in Z,\ p =_S q \wedge p(d_i) \neq_{d_i} q(d_i)\}$

Some skyline points are qualified order-sensitive because, depending on the preferences associated with dynamic dimensions, these points may be skyline or not. Note first that no global skyline points is order sensitive. Second, $CutSky$ points have to be searched among order sensitive skyline points.

Definition 13. ***(Order-sensitive skyline points)*** *The order-sensitive skyline set of the space D on the dataset E, is defined by*
$OsSky(D, E) = \{p \in Sky(D, E) \mid p \notin GSky(D, E)\}$ *or equivalently*
$OsSky(D, E) = Sky(D, E) - GSky(D, E)$.

Example 15. *Let $S = \{Price,\ Distance\}$ and $Z = \{Gr,\ Air\}$.*
Then $GSky(D, E) = \{a, b, e\}$ and $OsSky(D, E) = \{c, d, f, h\}$, because all the skyline points are distinct.

5.2 Skyline Associated with Dynamic Dimensions

This section details step 2 of our approach. In this step, we pre-compute the useful information that does not depend on the dynamic preferences provided

by users. For each dimension d_i with dynamic preferences, we introduce *the candidate skyline point* (CP_{d_i}), *the new skyline point set* ($NewSky_{(d_i,\wp_i^j)}$) and *the compromise candidate point set* ($CandComp_{(d_i,\wp_i^j)}$).
The set CP_{d_i} represents the points that may become skyline points over the dimension d_i. It is the set of points from $OsSky(D,E)$

(1) having on $d_i \in Z$ a value different from any point of $GSky(D,E)$ that dominates them,
(2) having the same value on the static dimensions but different values on $d_i \in Z$.

In the sequel, $p \prec^j_{d_i \bigcup S} q$ indicates that p dominates q on the subspace $d_i \bigcup S$ according to the first order preference $\wp_i^j$ of the dimension d_i.

Definition 14. ***(Candidate skyline points)*** *The candidate skyline point set of the dynamic dimension d_i, is defined by*
$CP_{d_i} =$
$\{p \in OsSky(D,E) \mid \exists\ q \in GSky(D,E),\ q \prec_S p,\ p(d_i) \neq_{d_i} q(d_i)\} \bigcup$
$\{p \in OsSky(D,E) \mid \exists\ q \in OsSky(D,E),\ q =_S p,\ p(d_i) \neq_{d_i} q(d_i)\}$

Example 16. *Let $S = \{Price, Distance\}$ and $d_i = \{Gr\}$. Then $CP_{Gr} = \{c,d,f,h\}$.*

To find the new skyline after the introduction of the new dimension d_i, it is sufficient to test the points in CP_{d_i} instead of all non-skyline points. This can significantly reduce the number of domination tests.

$NewSky_{(d_i,\wp_i^j)}$ (Algorithm 1) represents the set of points in CP_{d_i} that are preferred to the points in $GSky(D,E)$ according to the first order preference $\wp_i^j = v_j <_{d_i} *$ such that $v_j \in dom(d_i)$. Intuitively, NewSky points are equivalent to MaxSky points on d_i according to the preference $\wp_i^j$

Definition 15. ***(New skyline points)***
The new skyline point set of the dynamic dimension d_i, is defined by
$NewSky_{(d_i,\wp_i^j)} = \{p \in CP_{d_i} \mid \forall\ q \in GSky(D,E) \bigcup \{\ CP_{d_i} - p\},\ q \not\prec^j_{d_i \bigcup S} p\}$

Example 17. $NewSky_{(Gr,H<_{Gr}*)} = \{c,d,f,h\}$.

Finally, $CandComp_{(d_i,\wp_i^j)}$ (Algorithm 2) represents the set of points that may become skyline compromises (i.e. compromise candidates) when considering a new dimension. They are computed for each first order preference $\wp_i^j$ on d_i.

Definition 16. ***(Compromise candidate points)***
Let $E' = (CP_{d_i}$ - $NewSky_{(d_i,\wp_i^j)})$ and $E'' = (GSky(D,E) \bigcup NewSky_{(d_i,\wp_i^j)})$. The compromise candidate points associated with the preference $\wp_i^j$ is a set of pairs (p, Set_p) defined by
$CandComp_{(d_i,\wp_i^j)} =$
$\{(p, Set_p) \in E' \times \mathcal{P}(E'') \mid \forall q \in \mathcal{P}(E''),\ \exists d_k \in \{d_i\} \bigcup S,\ p \prec^j_{d_k} q\}$.

Algorithm 1. Calculate $NewSky_{(d_i,\wp_i^j)}$

input : d_i: a dimension, $\wp_i^j$: a first order preference on d_i, $GSky(D,E)$: global skyline set, CP_{d_i}: candidate skyline set over d_i
output: $NewSky_{(d_i,\wp_i^j)}$

```
1 NewSky_(d_i,℘_i^j) ← ∅
2 foreach p ∈ CP_{d_i} do
3     bool ← true
4     foreach q ∈ GSky(D,E) ⋃ { CP_{d_i} − p} do
5         if q ≺^j_{d_i ⋃ S} p then
6             bool← false
7             exit
8     if bool then
9         NewSky_(d_i,℘_i^j) ← NewSky_(d_i,℘_i^j) ⋃{p}
```

Example 18
$CandComp_{(Air,R<_{Air}*)} = \{(f,\{a,b\})\}$ where the notation $\{(f,\{a,b\})\}$ means that f belongs to $CandComp_{(Air,R<_{Air}*)}$ because f dominates a (resp. b) on at least one dimension from $\{Air\} \bigcup S$ (here Distance (resp. Airline))).

5.3 The EC²Sky Structure

Now, let us consider how to construct an EC^2Sky data structure to store efficiently all the precomputed information. Our aim is to avoid building a data structure containing all the combinations of the dynamic preferences on all dimensions as proposed in [20] . In section 5.1 and 5.2 and thanks to theorem 2, we have shown that the skyline of an extended dimensional subspace can be computed by taking into account first order preferences only. We propose to store in EC^2Sky structure all the sets $NewSky$ and $CandComp$ associated to any first order preference in each dimension.

For each dimension d_i, we compute and store CP_{d_i} and for each first order preference on d_i, we compute and store the two sets: $NewSky_{(d_i,\wp_i^j)}$ and $CandComp_{(d_i,\wp_i^j)}$ related to the first order preference j on dimension d_i. The sets $NewSky_{(d_i,\wp_i^j)}$ and $CandComp_{(d_i,\wp_i^j)}$ associated with any possible first order preference on dimension *Hotel group* or dimension *Airline* are presented in Table 3.

Now, we evaluate the space complexity of the EC^2Sky structure. Let m be the number of dimensions associated with dynamic preferences and c be the maximal cardinality of a dimension associated with dynamic preferences. The space complexity of the EC^{2}Sky structure is given by:

Algorithm 2. Calculate $CandComp_{(d_i,\wp_i^j)}$

input : d_i: a dimension, $\wp_i^j$: a first order preference on d_i, $NewSky_{(d_i,\wp_i^j)}$: skyline points added by d_i for $\wp_i^j$, $GSky(D,E)$: global skyline set, CP_{d_i}: candidate skyline set
output: $CandComp_{(d_i,\wp_i^j)}$

```
1 CandComp_(d_i,℘_i^j) ← ∅
2 foreach p ∈ {CP_di − NewSky_(d_i,℘_i^j)} do
3     Set_p ← ∅
4     foreach q ∈ {GSky(D,E) ⋃ NewSky_(d_i,℘_i^j)} do
5         foreach d_k ∈ {d_i} ⋃ S do
6             if p ≺^j_dk q then
7                 Set_p ← Set_p ⋃ q
8         CandComp_(d_i,℘_i^j) ← CandComp_(d_i,℘_i^j) ⋃ {(p, Set_p)}
```

Table 3. Illustration of an EC^2Sky structure with two dynamic dimensions and three first order preferences for each dimension

$GSky = \{a, b, e\}$					
$CP_{Gr} = \{c, d, f, h\}$			$CP_{Air} = \{f\}$		
$\wp = M <_{Gr} *$	$\wp = T <_{Gr} *$	$\wp = H <_{Gr} *$	$\wp = R <_{Air} *$	$\wp = G <_{Air} *$	$\wp = W <_{Air} *$
$NewSky_{\{Gr,\wp\}}$			$NewSky_{\{Air,\wp\}}$		
$\{f, h\}$	$\{\}$	$\{c, d, f, h\}$	$\{\}$	$\{\}$	$\{f\}$
$CandComp_{\{Gr,\wp\}}$			$CandComp_{\{Air,\wp\}}$		
$\{(c, \{a, b, e\}), (d, \{a, b, e\})\}$	$\{(f, \{a\}), (h, \{a\})\}$	$\{\}$	$\{(f, \{a, b\})\}$	$\{(f, \{a, e\})\}$	$\{\}$

$$\sum_{i=0}^{m}(c) = O(c.m)$$

We can note that the size of the EC^{2}Sky structure is significantly smaller than the number of possible *n-th* order preferences given by:

$$(\sum_{i=0}^{c-1}(P_i(c)))^m = O((c.c!)^m)$$

Where $P_i(c)$ is the number of permutations of ordering i elements from c elements. The space complexity of the EC^{2}Sky structure is also significantly smaller than the space complexity of IPO-tree structure given by:

$$\sum_{i=0}^{m}(c+1)^i = O(c^m)$$

Algorithm 3. Calculate $CutSky(D^{i-1})$

input : $Sky(D^{i-1}, E)$, $Sky(D^i, E)$ and $Sky(d_i \bigcup S, E)$
output: $CutSky(D^{i-1})$

```
1 CutSky(D^{i-1}) ← ∅
2 foreach p ∈ Sky(D^i, E) do
3     foreach q ∈ Sky(D^{i-1}, E) ⋃ Sky(d_i ⋃ S, E) do
4         if p =_{D^{i-1}} q and q ≺_{d_i ⋃ S} p then
5             CutSky(D^{i-1}) ← CutSky(D^{i-1}) ⋃ {p}
6             break
```

For example, when $m = 3$ and $c = 40$, the number of stored preferences in EC^{2}Sky structure is 123 only, while in IPO-tree structure is $70,644$, and the number of all possible *n-th* order preferences ($n \in 1, .., c$) is $4.1 * 10^9$. This is 574.35 times smaller than the IPO-tree and $714,502,572$ times smaller than the number of all possible *n-th* order preferences. The difference is more obvious when the number of dimensions m is high.

5.4 Query Evaluation

In this section, we describe step 3 of our proposal (cf. beginning of section 5). The information precomputed and stored in step 1 and 2 is used in step 3 to calculate, interactively, the skyline set according to the specified preferences in the user query.

One dimension with dynamic preferences First, we consider only one dimension with dynamic preferences in the dimensional space D. According to the user query, we are faced with two cases:
(i) *Query with first order preferences*: to compute the skyline associated with a first-order preference $\wp_i^j$, we use the two sets $GSky(D, E)$ and $NewSky_{(d_i, \wp_i^j)}$ stored in step 2, as follows : $Sky(d_i \bigcup S, E)_{(Z, \wp_i^j)} = GSky(D, E) \bigcup NewSky_{(d_i, \wp_i^j)}$. Recall that, when dealing with one dimension only, there is no compromise points ($CompSky = \emptyset$).

Example 19. *We use the EC^{2}Sky structure in Table 3 to illustrate the different steps of a query evaluation. The skyline associated with the preference $\wp = \{M <_{Gr} *\}$ (stored in the EC^{2}Sky structure shown in Table 3) is computed as follow:*
$Sky(\{Gr\} \bigcup S, E)_{(Z, M <_{Gr} *)} = GSky(D, E) \bigcup NewSky_{\{Gr, M <_{Gr} *\}}$.
Thus, $Sky(\{Gr\} \bigcup S, E)_{(Z, M <_{Gr} *)} = \{a, b, e, f, h\}$, which is the skyline for $\wp$.

Algorithm 4. Calculate $NewCompSky(D^i, E)$

input : EC^2Sky structure, $GSky(D, E)$: global skyline points
output: $NewCompSky(D^i, E)$

1 $CandCompSet = \bigcup^{i} \bigcup^{j} CandComp_{(d_i, \wp_i^j)}$
2 $NewCompSky(D^i, E) \leftarrow \emptyset$
3 **foreach** $p \in CandCompSet$ **do**
  `// Compute the set of points dominated by p on at least one dimension of` D^i
4 $Dominated(p) \leftarrow \{q | \exists\ d_i \in D^i, p \prec_{d_i} q\}$
  `// Select the set of points that dominate the skyline points on at least one dimension`
5 **if** $Dominated(p) = Sky(D^{i-1}, E) \bigcup Sky(d_i \bigcup S, E) - (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S)$ **then**
6 $NewCompSky(D^i, E) \leftarrow NewCompSky(D^i, E) \bigcup \{p\}$
7 Dominance test over all the elements of the set $NewCompSky(D^i, E)$

(ii) *Query with n-th order preferences*: in this case we use the *merging property* of Wong et al. [20] (see Theorem 1). This is illustrated by the following example.

Example 20. *The skyline associated with the preference* $\wp = \{M <_{Gr} H <_{Gr} *\}$ *can be computed from the skyline related to the preferences* $\wp^1 = \{M <_{Gr} *\}$ *and* $\wp^2 = \{H <_{Gr} *\}$ *(stored in the* EC^2Sky *structure shown in Table 3), as follow:*
$Sky(\{Gr\} \bigcup S, E)_{(Z, M <_{Gr} *)} = \{a, b, e, f, h\}$ *(cf. example 19), which is the skyline for* $\wp^1$.
In the same way, $Sky(\{Gr\} \bigcup S, E)_{(Z, H <_{Gr} *)} = \{a, b, e, c, d, f, h\}$, *which is the skyline for* $\wp^2$.

Finally, to compute $\wp = \{M <_{Gr} H <_{Gr} *\}$, *we use the merging property (Theorem 1) :*
$Sky(D, E)_{(Z, M <_{Gr} H <_{Gr} *)} =$
$(Sky(\{Gr\} \bigcup S, E)_{(Z, M <_{Gr} *)} \bigcap Sky(\{Gr\} \bigcup S, E)_{(Z, H <_{Gr} *)}) \bigcup$
$PSky(D, E)_{(Z, M <_{Gr} *)} =$
$(\{a, b, e, f, h\} \bigcap \{a, b, e, c, d, f, h\}) \bigcup \{f, h\} = \{a, b, e, f, h\}$

Several dimensions with dynamic preferences Second, we consider the case of several dimensions with dynamic preferences which is more complex. According to definition 10 and 11, some skyline points ($CutSky$ points) may be disqualified

Algorithm 5. EC^2Sky structure construction

input : E: Dataset, D: Data space of E, S: Subspace with static preferences, Z: Subspace with dynamic preferences
output: EC^2Sky structure

```
// Step 1: computation of static and order sensitive skyline
   points
```
1 Compute $GSky(D, E)$
2 Store $GSky(D, E)$ in the EC^2Sky structure
3 Compute $OsSky(D, E)$
4 Store $OsSky(D, E)$ in the EC^2Sky structure

```
// Step 2: computation of the EC2Sky structure
```
5 **foreach** $d_i \in Z$ **do**
6 Compute and Store CP_{d_i} in the EC^2Sky structure
7 **foreach** $\wp_i^j \in \wp_i$ **do**
8 Compute $NewSky_{(d_i,\wp_i^j)}$; `// Algorithm 1`
9 Store $NewSky_{(d_i,\wp_i^j)}$ in the EC^2Sky structure
10 Compute $CandComp_{(d_i,\wp_i^j)}$; `// Algorithm 2`
11 Store $CandComp_{(d_i,\wp_i^j)}$ in the EC^2Sky structure

when a new dimension is introduced, while new skyline points (*CompSky* points) may appear.

The computation of the *CutSky* set is described by Algorithm 3. We just show how to compute $CutSky(D^{i-1})$ because the calculation of $CutSky(d_i \bigcup S)$ is performed in the same way.

The compromise skyline points are the set of compromise candidates that become skyline compromises. The computation of this set is described by Algorithm 4.

We are now in position to detail the EC^2Sky method. Algorithm 5 and 6 describe the general process of EC^2Sky. Algorithm 5 outlines the required steps to construct the EC^2Sky structure. At the end of step 2 (Algorithm 5, lines 8 and 11), we calculate the skyline sets, stored in the EC^2Sky structure. Each of these sets corresponds to one dimension with dynamic preferences defined in the user query.

Algorithm 6 is dedicated to step 3, the computation of changing elements of the skyline. The sets *CompSky* (Algorithm 6, line 11) and the union of *CutSky* (Algorithm 6, line 10) are computed. As stated by the *incremental skyline theorem* (Theorem 2), the final skyline is obtained by eliminating all the *CutSky* points and by adding all the *CompSky* points to the union of skylines related to queries involving one dynamic preference (Algorithm 6, line 12).

Algorithm 6. $EC^2Sky(Sky(D,E)_{(Z,\wp)})$

input : $Sky(D,E)_{(Z,\wp)}$:skyline query
output: $Sky(D,E)_{(Z,\wp)}$

// Step 3: computation of changing points
$Sky(D^0,E)_{(Z,\wp)} \leftarrow GSky(D,E)$
for $i \leftarrow 1$ **to** $m = |Z|$ **do**
  if $\wp_i = \emptyset$ **then**
    $Sky(d_i \bigcup S,E)_{(Z,\wp)} \leftarrow GSky(D,E) \bigcup CP_{d_i}$
  else
    if *isFirstOrderPref*($\wp_i$) **then**
      $Sky(d_i \bigcup S,E)_{(Z,\wp)} \leftarrow GSky(D,E) \bigcup NewSky_{(d_i,\wp_i^j)}$
    else
      Use the merging property
  Compute $CutSky(D^{i-1})$ (resp. $CutSky(d_i \bigcup S)$)
  Compute $NewCompSky(D^i,E)$; // Algorithm 4
  $Sky(D^i,E)_{(Z,\wp)} \leftarrow Sky(D^{i-1},E)_{(Z,\wp)} \bigcup Sky(d_i \bigcup S,E)_{(Z,\wp)} \bigcup NewCompSky(D^i,E) - (CutSky(D^{i-1}) \bigcup CutSky(d_i \bigcup S))$
$Sky(D,E)_{(Z,\wp)} \leftarrow Sky(D^m,E)_{(Z,\wp)}$

Example 21. *The skyline associated with the preferences* $\wp = \{M <_{Gr} H <_{Gr} *, G <_{Air} *\}$*, can be computed from the skyline associated with the preferences* $\wp_1 = \{M <_{Gr} H <_{Gr} *\}$ *and* $\wp_2 = \{G <_{Air} *\}$.
Let $D^1 = \{Price, Distance, Gr\}$ *and* $D^2 = \{Price, Distance, Gr, Air\}$*. The skyline associated to* $\wp_1$ *and* $\wp_2$ *is computed in the same way as in example 20.*
$Sky(D^1,E)_{(Z,M<_{Gr}H<_{Gr}*)} = \{a,b,e,f,h\}$ *and* $Sky(\{Air\} \bigcup S,E)_{(Z,G<_{Air}*)} = \{a,b,e\}$.

Since, we have two dimensions with dynamic preferences we compute the sets $CutSky(D^1)$*,* $CutSky(\{Air\} \bigcup S)$ *and* $NewCompSky(D^2,E)$*. For this example,* $CutSky(D^1) = \emptyset$*,*
$CutSky(\{Air\} \bigcup S) = \emptyset$ *and* $NewCompSky(D^2,E) = \emptyset$*. Finally,*
$Sky(D^2,E)_{(Z,\wp)} = Sky(D^1,E)_{(Z,M<_{Gr}H<_{Gr}*)} \bigcup Sky(\{Air\} \bigcup S,E)_{(Z,G<_{Air}*)}$
$= \{a,b,e,f,h\}$.

Our proposal provides the user with a way to express preferences and with the ability to change them without being penalized by long response times. A good performance is achieved by storing only the minimal amount of information required to enable quick and easy updates.

The experimental evaluation presented in the following highlight the relevance of the proposed solution.

6 Experiments

In this section, we report an experimental evaluation of our algorithm EC^2Sky on synthetic data sets. EC^2Sky is implemented in $C/C++$ and the experiments were performed on an Intel Xeon CPU at 3GHz and 16 GB of RAM on a Linux platform. For static dimensions, the data were produced by the generator released by the authors of [4]. Three kinds of data sets were generated: independent data, correlated data and uncorrelated data. The description of these data sets can be found in [4]. Like in [20], we only show the experimental results for the uncorrelated data sets. The results for independent data sets and correlated data sets are similar, but the execution times are much shorter for correlated data sets. The dynamic dimensions were generated according to a Zipfian distribution [19]. By default, we set the Zipfian θ parameter to 1. We obtained $1,000,000$ tuples for 6 dimensions with static order. The number of dynamic dimensions varied from 1 to 40 and the cardinality of these dimensions varied from 2 to 50. We chose a query template such that the most frequent value of some dynamic dimension has the highest priority over all other values. This represents a parameter that becomes more difficult to manage as the skyline tends to be larger.

Table 4. Default values

Parameter	Value
No. of tuples	100K
No. of dimensions with static preferences (prefs)	6
No. of dimensions with dynamic prefs	4
No. of values in dimension with dynamic prefs	4
Zipfian parameter θ	1

In the following experiments, we compare the performance of our algorithm EC^{2}Sky with the algorithm IPO-tree implemented by [20], in terms of the execution time and the storage size.

Scalability with Respect to Dimensionality. In the first experiments, the number of static dimensions was set to 6 and the number of dynamic dimensions varied from 1 to 40. Figure 1 shows that the execution time and the storage size of both EC^2Sky and IPO-tree increase with the number of dynamic dimensions. However, the increase rate of IPO-tree is greater than the increase rate of EC^2Sky. This comes from the complexity of the preferences tree built by IPO-tree. The IPO-tree structure contains more nodes, yielding a larger storage size. Beyond 6 dynamic dimensions, IPO-tree overflows the available memory. This is due to its tree size in $O(c^m)$ (m is the number of dynamic dimensions and c the cardinality of a dimension), which induces an exponential increase of the storage size. The table built by EC^2Sky has a size in $O(c * m)$, which induces a

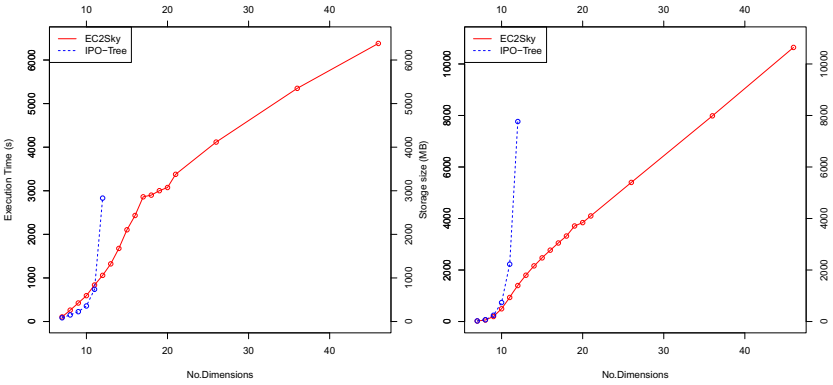


Fig. 1. Scalability with respect to dimensionality

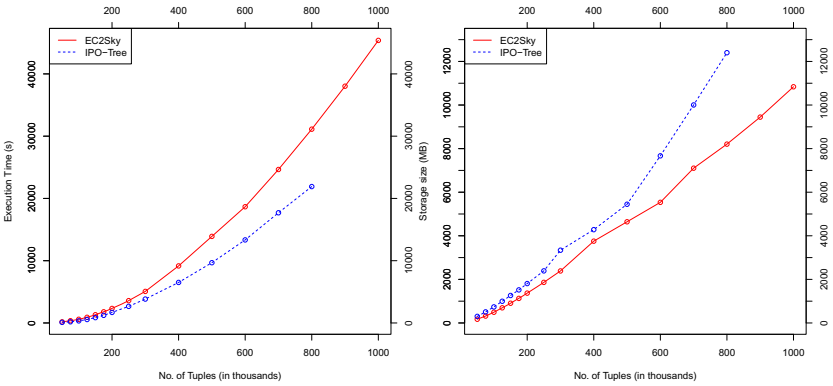
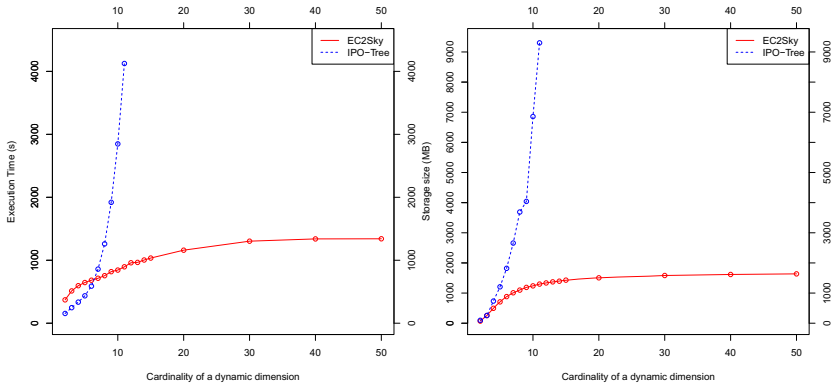


Fig. 2. Scalability with respect to database size

substantial increase of the storage size but which evolves more slowly than the IPO-tree. The results are similar to those in Figure 1 when the number of static dimensions is set to 4.

Scalability with Respect to the Database Size. In this experiment, the number of tuples of the dataset varies from $50,000$ to $1,000,000$. Figure 2 shows that the execution time and the storage size of both EC^2Sky and IPO-tree increase with the size of the dataset. This is because the size of the information stored and analyzed increases with the increase of the dataset. However, our method is more efficient than the IPO-tree method. Beyond 800.000 tuples, IPO-tree overflows the available memory. Indeed, IPO-tree stores all the skylines



(a) (b)

Fig. 3. Scalability with respect to the cardinality of the dynamic dimensions

associated to all possible combinations of the different first order preferences of all the dynamic dimensions whereas EC^2Sky stores only the skyline points corresponding to the first order preferences of each dynamic dimension. The skyline of the various combinations of preferences are derived from simple operations of intersection and union.

Scalability with Respect to the Cardinality of Dynamic Dimensions. We vary the cardinality of the dynamic dimensions from 2 to 50. Figure 3 shows that the execution time and the storage size of both EC^2Sky and IPO-tree increase when the cardinality of the dimensions increases. Once more, EC^2Sky is more efficient than IPO-tree. The size of the treestructure of IPO-tree is exponential in $O(c^m)$. So, it becomes more complex and larger when the cardinality of dimensions (c) increases. IPO-tree overflows the available memory for a dynamic dimension cardinality above 11. We can also observe a significant increase of the related execution time.

7 Conclusion

In this paper, we have proposed a new efficient method to compute skyline queries in the presence of dimensions associated with dynamic user preferences. We have investigated preferences on dimension values that can be expressed by any partial or complete order, with a particular focus on the compromise points which are important in decision making. Our approach, is based on a materialization of the first order preferences, that can respond efficiently to skyline queries related to user preferences even in the context of large volumes of data. The experimentations presented in this paper highlight the performance improvements of EC^2Sky compared to IPO-tree [20].

The consideration of dimensions with dynamic preferences opens several promising future research directions. First, to demonstrate the usefulness of our method, we want to experiment our algorithm on a real data set. We are particularly interested in the analysis of simulation results from a biophysical model to extract the most polluting plots in a watershed with respect to different analysis criteria. Another possible future direction is to investigate how to compute skyline queries in the context of hierarchical and aggregated data. The adopted approach would search the best compromises along the set of axes. However, this approach raises several problems. One is to define a computation adapted to the level of the explored hierarchy. Another is to define the semantics of skyline points at different levels of granularity.

Acknowledgments. This work was funded by the French National Research Agency (ANR) through the ACASSYA project (ANR-08-STRA-01).

References

[1] Balke, W.T., Guntzer, U., Siberski, W.: Exploiting indifference for customization of partial order skylines. In: Proceedings of the 10th International Database Engineering and Applications Symposium, pp. 80–88. IEEE Computer Society (2006)
[2] Bentley, J.L., Kung, H.T., Schkolnick, M., Thompson, C.D.: On the average number of maxima in a set of vectors and applications. J. ACM 25(4), 536–543 (1978)
[3] Bitran, G.R., Magnanti, T.L.: The structure of admissible points with respect to cone dominance. Optimization Theory and Applications 29(4), 573–614 (1979)
[4] Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc. of the 17th International Conference on Data Engineering, pp. 421–430. IEEE Computer Society (2001)
[5] Bouadi, T., Cordier, M.-O., Quiniou, R.: Incremental computation of skyline queries with dynamic preferences. In: Liddle, S.W., Schewe, K.-D., Tjoa, A.M., Zhou, X. (eds.) DEXA 2012, Part I. LNCS, vol. 7446, pp. 219–233. Springer, Heidelberg (2012)
[6] Brando, C., Goncalves, M., González, V.: Evaluating top-k skyline queries over relational databases. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 254–263. Springer, Heidelberg (2007)
[7] Chen, L., Lian, X.: Efficient processing of metric skyline queries. IEEE Trans. on Knowl. and Data Eng. 21(3), 351–365 (2009)
[8] Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting: Theory and optimizations. In: Proc. of Intelligent Information Systems, pp. 595–604. Springer, Heidelberg (2005)
[9] Godfrey, P., Shipley, R., Gryz, J.: Algorithms and analyses for maximal vector computation. The VLDB Journal 16(1), 5–28 (2007)
[10] Huang, Z., Guo, J., Sun, S.L., Wang, W.: Efficient optimization of multiple subspace skyline queries. J. Comput. Sci. Technol. 23(1), 103–111 (2008)
[11] Jin, W., Tung, A.K.H., Ester, M., Han, J.: On efficient processing of subspace skyline queries on high dimensional data. In: Proc. of the 19th International Conference on Scientific and Statistical Database Management. IEEE Computer Society (2007)

[12] Mindolin, D., Chomicki, J.: Preference elicitation in prioritized skyline queries. The VLDB Journal 20(2), 157–182 (2011)
[13] Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. 30(1), 41–82 (2005)
[14] Pei, J., Jin, W., Ester, M., Tao, Y.: Catching the best views of skyline: a semantic approach based on decisive subspaces. In: Proc of the 31st International Conference on Very Large Data Bases, pp. 253–264, VLDB Endowment (2005)
[15] Raïssi, C., Pei, J., Kister, T.: Computing closed skycubes. Proc. VLDB Endow. 3(1), 838–847 (2010)
[16] Sawaragi, Y., Nakayama, H., Tanino, T.: Theory of Multiobjective Optimization. Academic Press, Orlando (1985)
[17] Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient progressive skyline computation. In: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 301–310. Morgan Kaufmann Publishers Inc. (2001)
[18] Tao, Y., Xiao, X., Pei, J.: Efficient skyline and top-k retrieval in subspaces. IEEE Trans. on Knowl. and Data Eng. 19(8), 1072–1088 (2008)
[19] Trenkler, G.: In: Johnson, N.l., Kotz, S., kemp, A.W. (eds.) Univariate Discrete Distributions, 2nd edn. John wiley (1994) ISBN 0-471-54897-9; Computational Statistics & Data Analysis, 17(2), 240–241 (1994)
[20] Wong, R.C.W., Fu, A.W.C., Pei, J., Ho, Y.S., Wong, T., Liu, Y.: Efficient skyline querying with variable user preferences on nominal attributes. Proc. VLDB Endow. 1(1), 1032–1043 (2008)
[21] Wong, R.C.W., Pei, J., Fu, A.W.C., Wang, K.: An erratum on "online skyline analysis with dynamic preferences on nominal attributes". IEEE Trans. on Knowl. and Data Eng. (to be published)
[22] Wong, R.C.W., Pei, J., Fu, A.W.C., Wang, K.: Online skyline analysis with dynamic preferences on nominal attributes. IEEE Trans. on Knowl. and Data Eng. 21(1), 35–49 (2009)
[23] Xia, T., Zhang, D., Tao, Y.: On skylining with flexible dominance relation. In: Proc. of the 2008 IEEE 24th International Conference on Data Engineering, pp. 1397–1399. IEEE Computer Society (2008)
[24] Yuan, Y., Lin, X., Liu, Q., Wang, W., Yu, J.X., Zhang, Q.: Efficient computation of the skyline cube. In: Proc. of the 31st International Conference on Very Large Data Bases, pp. 241–252, VLDB Endowment (2005)