Decisive Skyline Queries for Truly Balancing Multiple Criteria

Akrivi Vlachou[a,*], Christos Doulkeridis[b], João B. Rocha-Junior[c] and Kjetil Nørvåg[d]

[a]Department of Information & Communication Systems Engineering, University of the Aegean, Karlovasi, 83200, Greece
[b]Department of Digital Systems, University of Piraeus, Karaoli & Dimitriou 80, Piraeus, 18534, Greece
[c]Universidade Estadual de Feira de Santana, Av. Transnordestina, Feira de Santana, 44036-900, Brazil
[d]Norwegian University of Science and Technology, Sem Sælandsv. 9, Trondheim, 7491, Norway

ARTICLE INFO

Keywords:
skyline queries
decisive subspaces
query processing

ABSTRACT

Skyline queries have emerged as an increasingly popular tool for identifying a set of interesting objects that balance different user-specified criteria. Although in several applications the user aims to detect data objects that have values as good as possible in *all* specified criteria, skyline queries fail to identify only those objects. Instead, objects whose values are good in a subset of the given criteria are also included in the skyline set, even though they may take arbitrarily bad values in the remaining criteria. To alleviate this shortcoming, we study the decisive subspaces that express the semantics of skyline points and determine skyline membership. We propose a novel alternative query, called *decisive skyline query*, which retrieves a set of points that balance all specified criteria. We study two variants of the proposed query, the *strict* variant, which retrieves only the subset of skyline points that have the full data space as decisive subspace, and the *relaxed* variant, which imposes the decisive semantics in a more flexible way. Furthermore, we present pruning properties that accelerate the process of finding the decisive skyline set. Capitalizing on these pruning properties, we propose a novel efficient algorithm for computing decisive skyline points. Our experimental study, which employs both synthetic and real data sets for various experimental setups, demonstrates the efficiency and effectiveness of our algorithm, and shows that the newly proposed query is more intuitive and informative for the user.

1. Introduction

Skyline queries [1] constitute a powerful tool for data analysis and multi-objective optimization, as they enable balancing of different (and often conflicting) criteria specified by the user. Given a multidimensional data set, the dimensions correspond to the criteria that need to be balanced by the skyline query. Such queries return a set of interesting data points, also called *skyline points*, that are not dominated by any other point in all dimensions. A point p *dominates* another point q, if p is better than or equal to q in all dimensions and strictly better than q in at least one dimension. Nevertheless, the skyline set contains also points that fail to balance among all given criteria, as we demonstrate in the following example.

Example 1. *(Motivating example) Assume that a tourist is interested in booking a hotel with a low price, a good ranking based on the customers' ratings, and nearby the beach. To this end, she performs skyline analysis on an online hotel database in order to discover hotels that fulfill all criteria* [1]*. In Figure 1, the hotels belonging to the skyline set of the hotel database are depicted. However, by inspecting this result set, we observe that several hotels have good values on subsets of the available criteria only, but not all of them. For instance,* City Center Hotel *is included in the result due to its rank (minimum value), despite the fact that it has the second worst value in distance. Similarly,* Paradise Hotel *has the lowest price, but it fails to balance the remaining two criteria. The same holds for* Sunset Hotel*, which has the worst rank among the skyline points and is included in the result set because of its combined values in distance and price. However, had the user been really interested in such a hotel, she would have specified as criteria only a subset of the dimensions, namely distance and price. In this paper, we argue that the user needs a new query type that*

*Corresponding author
avlachou@aegean.gr (A. Vlachou); cdoulk@unipi.gr (C. Doulkeridis); joao@uefs.br (J.B. Rocha-Junior); noervaag@ntnu.no (K. Nørvåg)
ORCID(s): 0000-0002-1961-9167 (A. Vlachou); 0000-0002-3219-0510 (C. Doulkeridis); 0000-0002-6925-9729 (J.B. Rocha-Junior); 0000-0002-4250-9329 (K. Nørvåg)

[1] A skyline query supports both minimization and maximization of different dimensions. Without loss of generality or inconsistency, we assume that minimum values are preferable, unless mentioned otherwise.

	Price	*Distance*	*Rank*
Panorama Hotel	$70	600m	8
Heaven Hotel	$50	100m	16
Beach Hotel	$60	1000m	4
Imperial Resort	$200	200m	6
City Center Hotel	$160	1500m	1
Sunset Hotel	$20	300m	20
Paradise Hotel	$10	1800m	18

Figure 1: Hotels in the skyline of a hotel database.

excludes from its results set hotels that may have arbitrarily bad values. In this example, only Panorama Hotel *would satisfy the constraint imposed by this new query. We call this new query type* decisive skyline query.

As shown in the example, the skyline query always returns the data point with the best value in one dimension, regardless of the values in the other dimensions, as this point cannot be dominated by any other point. Yet, this point will not be seriously considered by a user who is interested all queried dimensions, despite the fact that it belongs to the skyline set. Hence, we observe that skyline queries fail to fulfill all user-specified criteria. Put differently, the skyline definition imposes "OR semantics" between the different criteria. In the hotel database, the skyline set contains the hotels that are the best tradeoffs among (a) rank, price and distance, OR (b) rank, price, OR (c) price, distance, OR have (d) the minimum price, OR (e) the minimum rank, OR (f) the minimum distance. But this is not the objective of the user's search, since she specified a combination of *three* criteria, i.e., the best tradeoffs between rank, price and distance.

A direct consequence (or effect) of the aforementioned "OR semantics" is that the skyline cardinality [2, 3, 4] increases rapidly with the dimensionality of the data space. Therefore, the skyline operator loses its discriminating power and returns a large fraction of the data set as results. The high cardinality of the skyline set originates from the fact that as the number of criteria increases, the combinations of different criteria increase exponentially. In turn, the probability that a point is dominated in all different combinations decreases, thus leading to more skyline points. Intuitively, it is more difficult to satisfy more criteria, therefore it would be expected that with increasing the number of criteria, the result size should decrease or stay fixed. Should we add too many criteria, none of the points will be able to satisfy all of them, thus resulting into an empty result set. In contrast to this intuition, the cardinality of skyline set increases with increasing dimensionality and can sometimes be comparable to the size of the complete data set.

Most existing approaches focus on the effect of the problem and try to restrict the skyline cardinality, often motivated by the controlled output size of top-k queries [5, 6, 7]. Towards this goal, different categories of approaches have been recently proposed, including (1) selecting k representative skyline points [8, 9, 10, 11], (2) restricting the skyline cardinality by changing the dominance relationship [12], and (3) ranking the skyline points based on different metrics [13, 14, 15] or user-defined functions [16, 17]. More importantly, none of these approaches take into account the semantics of the skyline points.

In this paper, we take a radically different approach. We address what we consider to be the cause of the problem, and not the effect. To this end, we revisit the semantics of skyline queries (first studied by Pei et al. [18]), analyze their complex relationships in subspaces (subsets of dimensions), and focus on the important concept of *decisive subspaces*. Informally, the decisive subspaces of a skyline point are responsible for the point being part of the skyline set, i.e., its values in these dimensions qualify it as skyline point. Capitalizing on this concept, we propose a novel query type, called *decisive skyline query*. Only points that belong to the skyline set due to their values in *all* dimensions belong to the result set of the novel query, and it is guaranteed that none of their values is arbitrarily bad. We investigate two different variants of the decisive skyline query, the *strict* variant, which returns only the subset of skyline points that have the full space as decisive subspace, and the *relaxed* variant, which returns also points with decisive subspaces that *cover* the entire data space. Interestingly, as a by-product, it turns out that the decisive skyline query does not suffer from increased output size for increased dimensionality. We emphasize that this is the first paper that focuses on the significance of retrieving points based on the properties of their decisive subspaces and on efficient processing algorithms for this set of points, since in [18] the aim was to find the subspace skyline points of all subspaces.

Symbol	Description
$\mathcal{D}$	Data space, i.e., $\mathcal{D} = \{d_1, \ldots, d_m\}$
m	Dimensionality of data space $\mathcal{D}$
U, V	Subspaces of the data space $\mathcal{D}$
P	Data set
n	Cardinality of the data set
p, q, r, p_i	Data points
$\mathcal{S}(P)$	Skyline points of data set P
$\mathcal{DS}(P)$	Strict decisive skyline points of data set P
$\widehat{\mathcal{DS}}(P)$	Relaxed decisive skyline points of data set P
$\mathcal{S}_U(P)$	Subspace skyline points of data set P in U
$DecSub(G)$	Decisive subspaces of group of points G

Table 1
Overview of symbols.

In more detail, we make the following contributions in this paper:

- We capitalize on the semantics of decisive subspaces and propose a new query type, termed *decisive skyline query*, which returns points that truly balance *all* user-specified criteria. This alleviates the deficiency of the skyline query that may retrieve many data points that have arbitrarily bad values in some dimensions. We study two variants of the proposed query.
- We present novel pruning properties that drastically reduce the candidate objects for inclusion in the strict decisive skyline set, and boost the performance of query processing. Furthermore, we propose an efficient algorithm for computing the decisive skyline points that exploits the pruning properties.
- We demonstrate the efficiency of our approach by means of a detailed experimental evaluation. We study the output size of the decisive skyline query, the effect of varying different parameters on performance, and the quality of the result of the decisive skyline query, which shows that the decisive skyline set does not contain points that take arbitrarily bad values in a subset of the given criteria. Moreover, we show that existing techniques for selecting k skyline points fail to select the decisive skyline points.

This paper extends our previous work [19] published in DAWAK'22, by providing a more thorough study of the problem, adding theoretical results, an extension of our algorithm for processing relaxed decisive skyline queries, as well as additional experimental results.

The rest of the paper is organized as follows. In Section 2, we present the necessary background knowledge for our approach. In Section 3, we provide the necessary definitions and formally state the problem. In Section 4, we present useful pruning properties for efficient computation of the decisive skyline query, and we describe a novel index-based algorithm. Section 5 follows presenting the results of the experimental evaluation. In Section 6, we discuss related work and we conclude in Section 7.

2. Preliminaries

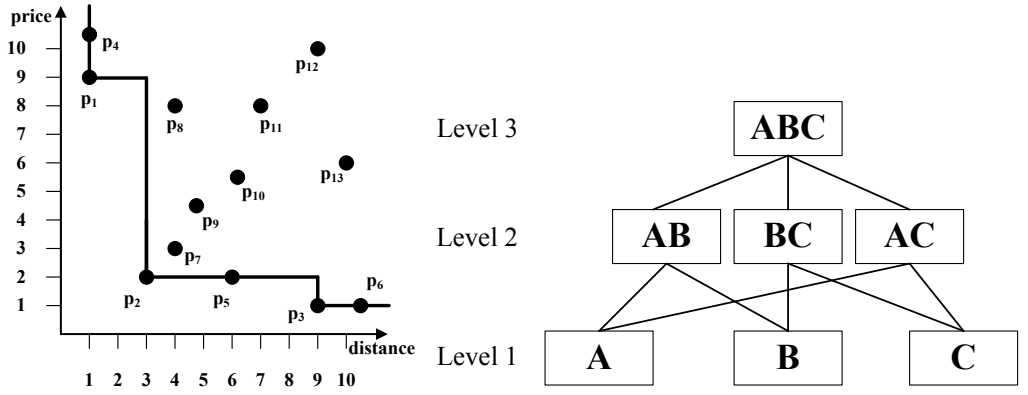
Given a data set P on a data space $\mathcal{D}$ defined by a set of m dimensions $\{d_1, \ldots, d_m\}$, a data object $p \in P$ is represented as an m-dimensional point p=$\{p[1], \ldots, p[m]\}$ where $p[i]$ is the value on dimension d_i. Table 1 offers an overview of the symbols that are used throughout this paper.

Definition 1. *(Skyline) A point $p \in P$ dominates another point $q \in P$, denoted as $p \prec q$, if (1) on every dimension d_i, $p[i] \leq q[i]$; and (2) on at least one dimension d_j, $p[j] < q[j]$. The skyline $\mathcal{S}(P)$ is a set of points which are not dominated by any other point in P.*

Without loss of generality, we assume that skylines are computed with respect to min conditions on all dimensions and that all values are non-negative. The notion of skyline can be extended to subspaces. Each non-empty subset U of $\mathcal{D}$ ($U \subseteq \mathcal{D}$) is referred to as a *subspace*[2] of $\mathcal{D}$. The data space $\mathcal{D}$ is also referred to as full space of the data set P.

Definition 2. *(Subspace skyline) A point $p \in P$ is said to* dominate *another point $q \in P$ on subspace $U \subseteq \mathcal{D}$, denoted as $p \prec_U q$, if (1) on every dimension $d_i \in U$, $p[i] \leq q[i]$; and (2) on at least one dimension $d_j \in U$, $p[j] < q[j]$. The* skyline *of a subspace $U \subseteq \mathcal{D}$ is a set of points $S_U(P) \subseteq P$ which are not dominated by any other point on subspace U. The points in $S_U(P)$ are called* subspace skyline points *in U.*

As shown in [18, 20], the skyline set of the full space does not contain all the subspace skyline points of the different subspaces. A skyline point q in $S_U(P)$ is either a skyline point in $S_V(P)$ (assuming $U \subset V$) or there exists another data point p, such that $p[i] = q[i]$ ($\forall d_i \in U$), that dominates q on the dimension set $V - U$.



(a) Subspace skyline example. (b) Lattice structure.

Figure 2: Subspace skyline example and lattice structure of the data space $\{ABC\}$.

Example 2. *In Figure 2(a), each point represents a hotel where the y-dimension represents the price of a room, while the x-dimension captures the distance of the hotel to the beach. A hotel dominates another hotel because it is cheaper and closer to the beach. In this example, the skyline points are $S(P) = \{p_1, p_2, p_3\}$, while for the (non-empty) subspace $U = \{price\}$ the skyline points on U are $S_U(P) = \{p_3, p_6\}$. Notice that the point p_6 is a skyline point on the subspace $\{price\}$ but it is dominated by the point p_3 in the space $\{price, distance\}$.*

Given an m-dimensional space $\mathcal{D}$, the number of different subspaces of $\mathcal{D}$ is $2^m - 1$. The way different subspaces can be visualized in a lattice structure is shown in Figure 2(b), where all possible non-empty subspaces of a 3-dimensional space are depicted. In addition, the lattice structure shows which subspaces share common dimensions. From the bottom to the top of the lattice, we number each level of subspaces increasingly. For two subspaces U and V, if V is a subset of U ($V \subset U$) and the level of U is equal to the level of subspace V increased by one, then we call the subspace U a *parent subspace* of V, i.e. $U = parent(V)$. Analogously, we call subspace V a *child subspace* of subspace U. Consider the lattice structure of the subspaces shown in Figure 2(b). The subspace $\{A\}$ is a child of $\{AB\}$ and $\{AB\}$ is a parent of $\{A\}$, while the subspace $\{ABC\}$ is not a parent of $\{A\}$.

3. Problem Formulation

We first define the concept of the *decisive subspace*, which was initially introduced in [18]. To ease exposition and provide the intuition for the semantics of the decisive skyline points, we first concentrate on the simple case where the *distinct value condition* [20] holds (Section 3.1). Then, we extend our definitions for the general case that allows non-distinct values (Section 3.2), and define decisive skyline sets formally (Section 3.3).

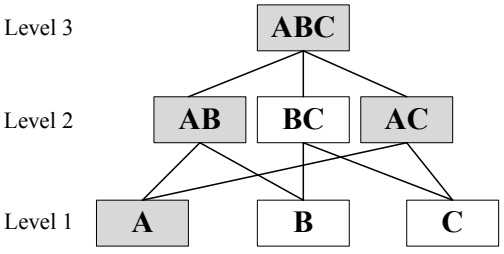
[2]When it is more convenient, we will also use a different notation for subspaces, e.g., ABC refers to the 3-dimensional subspace of the full space D=ABCDE that consists of dimensions A, B and C.

	A	B	C
p_1	4	5	2
p_2	3	1	6
p_3	5	4	4
p_4	7	2	3
p_5	6	6	1
p_6	2	3	8
p_7	1	7	7

(a) Data set

	DecSub( )
p_1	{AC,AB}
p_2	{B,AC}
p_3	{ABC}
p_4	{AC}
p_5	{C}
p_6	{AB}
p_7	{A}

(b) *DecSub*



(c) $p_7 \in S_U(P)$

Figure 3: Example of decisive subspaces.

3.1. Intuition of Decisive Subspaces

Let us first assume that the distinct value condition holds, which means that no two points share the same value in a given dimension (i.e., for any two points p and q of P it holds that $\forall d_i \in \mathcal{D} : p[i] \neq q[i]$). In this case, any subspace skyline point also belongs to the skyline set of the full space, which in turn simplifies the definition of the decisive skyline queries. Under the distinct value condition, the *decisive subspace* of a skyline point p is defined as follows.

Definition 3. *(Decisive subspace) For a skyline point $p \in \mathcal{S}(P)$, a subspace U of $\mathcal{D}$ is called decisive, if (1) p is a subspace skyline in U ($p \in \mathcal{S}_U(P)$), and (2) there exists no subspace $V \subset U$ such that p is a subspace skyline point in V ($\nexists V \subset U$ such that $p \in \mathcal{S}_V(P)$).*

A skyline point p can have multiple decisive subspaces. We use $DecSub(p)$ to denote the set of decisive subspaces for a skyline point p. The significance of the concept of decisive subspace is due to the fact that it justifies why a point belongs to the skyline set.

Corollary 1. *Given a data set P defined in a data space $\mathcal{D}$, if a subspace U belongs to the decisive subspaces of a skyline point p, then p belongs to the subspace skyline of any subspace U' ($p \in \mathcal{S}_{U'}(P)$) such that $U \subseteq U' \subseteq \mathcal{D}$.*

Corollary 1 directly relates to the question why point p belongs to the skyline set. If a point p has a decisive subspace $U \subset \mathcal{D}$, then this fact alone promotes p to become a full space skyline, irrespective of p's values in dimension set $\mathcal{D}-U$. Obviously, such skyline points may not balance the remaining dimensions. These are important properties that have been overlooked in related work.

To address the problems of the semantics of the traditional skyline operator, we define the *strict decisive skyline* set $\mathcal{DS}(P)$ as the set of skyline points that have the full space $\mathcal{D}$ as their decisive subspace, i.e., $\mathcal{DS}(P) = \{p | p \in \mathcal{S}(P)$ and $DecSub(p) = \mathcal{D}\}$. Based on the definition of decisive subspaces for the case of distinct values, a skyline point p belongs to the decisive skyline set, if there does not exist any other subspace $U \subset \mathcal{D}$ for which p belongs to the subspace skyline set ($\nexists U \subset \mathcal{D}$ such that $p \in \mathcal{S}_U(P)$). We argue that points in the decisive skyline set are guaranteed to have good values in all given criteria, in contrast to subspace skyline points.

The above definition imposes the semantics of decisive skyline sets in a strict or rigid way. A more relaxed variant, denoted $\widehat{\mathcal{DS}}(P)$, is also defined as follows: $\widehat{\mathcal{DS}}(P) = \{p | p \in \mathcal{S}(P)$ and $\bigcup_{(\forall U_i \in DecSub(p))} U_i = \mathcal{D}\}$. This *relaxed decisive skyline* set also includes points that belong to subspace skyline sets, as long as their decisive subspaces *cover* the full space. Thus, the relaxed decisive skyline points also balance all criteria, but possibly in different subspaces that cover the full space. Also, notice that by definition $\mathcal{DS}(P) \subseteq \widehat{\mathcal{DS}}(P)$.

Example 3. *Consider a data space $\mathcal{D} = ABC$ and a data set P defined in $\mathcal{D}$ that contains seven points as depicted in Figure 3(a). All points are skyline points and Figure 3(b) depicts their decisive subspaces. Decisive subspaces define why and in which subspaces a point p_i belongs to the subspace skyline set. Figure 3(c) shows – in gray shading – the subspaces in which p_7 is in the skyline. For p_7, subspace A is the decisive subspace, therefore the value of A is sufficient to qualify p_7 as a skyline point in the full space independently of its values in the other dimensions, which could be*

arbitrarily bad. Similar for p_2 and p_5 the decisive subspaces are B and C respectively. On the other hand, AC is also a decisive subspace for p_2, because p_2 appears in the subspace skyline of AC, and AC is not a super-set of B. Only point p_3 has the full space ABC as decisive subspace, and this is the only point in this example that belongs to the decisive skyline set $\mathcal{DS}(P) = \{p_3\}$.

On the other hand, points p_1 and p_2 may also be considered as good options, even though they do not belong to $\mathcal{DS}(P)$. For example, p_2 has the best value in dimension B, but also balances nicely dimensions AC, since it is in the subspace skyline in AC. In contrast, p7 has the best value in subspace A, but fails to balance the remaining criteria. Points p_1 and p_2, together with p_3, belong to the relaxed decisive skyline set $\widehat{\mathcal{DS}}(P) = \{p_1, p_2, p_3\}$.

3.2. Formal Definition of Decisive Subspaces

In the following, we withdraw the restriction on points taking distinct values. In the general case, where multiple data points may share the same value on some dimension, the main difference is that there may exist subspace skylines that do not belong to the full space skyline points. Recall that a subspace skyline point $q \in \mathcal{S}_U(P)$ is either a skyline point in the full space or there exists another data point p, such that $p[i] = q[i]$ $(\forall d_i \in U)$, that dominates q on the dimension set $\mathcal{D}-U$. If such a point p exists, then the remaining $\mathcal{D}-U$ dimensions are important to determine whether q qualifies as a skyline point. This property changes the definition of the decisive skyline sets, as we will elaborate in the sequel. Therefore, we provide definitions for some additional concepts that are necessary, in order to define the decisive skyline sets for the general case.

Definition 4. *(Maximal set of non-distinct points) Given a set of points G and set of dimensions U, we define as maximal set of non-distinct points the set: $\mathcal{O}(G, U) = \{p_i | p_i \in P,\ \forall d_k \in U \text{ and } \forall p_j \in G : p_i[k] = p_j[k]\}$.*

Based on the above definition, $\mathcal{O}(G, U)$ is the maximal set of points of P with identical values with the points of G in U, i.e., there exists no other point $q \in P$ with this property.

Definition 5. *(Maximal skyline group)*[3] *Given a set of points G and set of dimensions U, the pair $\{G, U\}$ is called maximal skyline group and is denoted as $SG(G, U)$, if it holds that*

1. *$\forall p_i \in G$ it holds that $p_i \in \mathcal{S}_U(P)$*
2. *$\forall p_i, p_j \in G$ and $\forall d_k \in U$ $p_i[k] = p_j[k]$*
3. *$\nexists d_k \in \mathcal{D} - U$ such that $\forall p_i, p_j \in G : p_i[k] = p_j[k]$*
4. *$\nexists p_j \in P - G$ such that $\exists p_i \in G$ and $\forall d_k \in U : p_i[k] = p_j[k]$*

Intuitively, $SG(G, U)$ is the maximal set of points with same values in U, these points are subspace skylines in U, and U is the maximal set of dimensions for which this set of points coincide. In more detail, the conditions used to define the concept of maximal skyline group can be interpreted as follows. Condition 1 demands that all points in group G are subspace skyline points in U. Condition 2 requires that all points in group G are identical with respect to U, i.e., they have the same values in all dimensions of U. Condition 3 states that U is the maximal subspace for which all points p_i of G share the same values in all dimensions. Finally, Condition 4 demands that G is a maximal group of points, i.e., there exists no other point that does not belong to G and has the same values in U. Moreover, notice that G may contain only one point, i.e., $G = \{p\}$. More precisely, every skyline point p forms a maximal skyline group $SG(\{p\}, \mathcal{D})$ assuming that there is no point p' such that $\forall d_k \in \mathcal{D} : p[k] = p'[k]$.

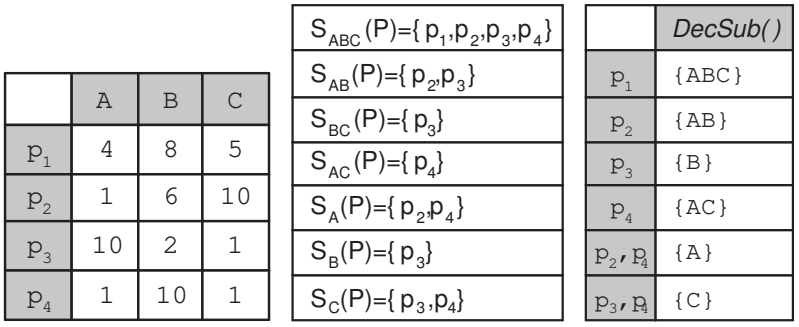
Example 4. *Consider the small 3-dimensional data set P depicted in Figure 4(a). The skyline sets for all subspaces are depicted in Figure 4(b). In this example, the pair of $\{\{p_2, p_4\}, A\}$ is a valid maximal skyline group $SG(\{p_2, p_4\}, A)$, because p_2 and p_4 coincide in A, are subspace skylines in A, and there exist no other points sharing the same values neither another dimension for which p_2 and p_4 have the same value. On the other hand, $\{\{p_3\}, C\}$ is not a valid maximal skyline group, since Condition 4 is violated (p_3 shares the same values with p_4 in dimension C).*

In the following, we define the concept of decisive subspaces for a maximal skyline group.

Definition 6. *(Decisive subspaces of maximal skyline group) Given a maximal skyline group $SG(G, U)$, a subspace $V \subseteq U$ is called decisive for $SG(G, U)$ if*

1. *$\forall p_i \in G$ it holds that $p_i \in \mathcal{S}_V(P)$*
2. *$\mathcal{O}(G, V) = G$*
3. *$\nexists V' \subset V$ such that conditions 1) and 2) hold for V'.*

[3] This definition is equivalent to the definition of [18].



	A	B	C
p_1	4	8	5
p_2	1	6	10
p_3	10	2	1
p_4	1	10	1

(a) Data set

$S_{ABC}(P)=\{p_1,p_2,p_3,p_4\}$
$S_{AB}(P)=\{p_2,p_3\}$
$S_{BC}(P)=\{p_3\}$
$S_{AC}(P)=\{p_4\}$
$S_{A}(P)=\{p_2,p_4\}$
$S_{B}(P)=\{p_3\}$
$S_{C}(P)=\{p_3,p_4\}$

(b) $S_U(P)$

	DecSub()
p_1	{ABC}
p_2	{AB}
p_3	{B}
p_4	{AC}
p_2, p_4	{A}
p_3, p_4	{C}

(c) *DecSub*

Figure 4: Example of decisive skyline set.

3.3. Decisive Skyline Points

We denote the decisive subspaces of the maximal skyline group $SG(G,\mathcal{D})$ as $DecSub(G)$. Notice that in the general case, the decisive subspaces are defined based on a group of points that coincide in a subspace U. Given a decisive subspace V of maximal skyline group $SG(G,U)$ it holds that $V \subseteq U$. A decisive subspace V of $SG(G,U)$ means that all points in G share the same values in U and are in the subspace skyline set for every subspace V' such that $V \subseteq V' \subseteq U$. We cannot conclude if all points of G belong to the skyline set of $\mathcal{D}$, since depending on the remaining dimensions some of them may be dominated. Only if they are incomparable in the remaining dimensions, then all of them will belong to the skyline set. On the other hand, skyline points that belong to a maximal skyline group $SG(G,\mathcal{D})$ that has the full space as a decisive subspace are included to the skyline set based on the values of all given dimensions, regardless if these points form groups in some subspaces.

Definition 7. *(Strict decisive skyline points) A skyline point p belongs to the decisive skyline set $\mathcal{DS}(P) \subseteq \mathcal{S}(P)$ of a data set P, if there exists a maximal skyline group $SG(G,\mathcal{D})$ such that $p \in G$ and the decisive subspace of G is the full space ($DecSub(G) = \{\mathcal{D}\}$).*

Definition 8. *(Relaxed decisive skyline points) A skyline point p belongs to the relaxed decisive skyline set $\widehat{\mathcal{DS}}(P) \subseteq \mathcal{S}(P)$ of a data set P, if there exists a maximal skyline group $SG(G,\mathcal{D})$ such that $p \in G$ and the union of the decisive subspaces of G is the full space ($\bigcup_{\forall U_i \in DecSub(G)} U_i = \{\mathcal{D}\}$).*

Example 5. *Consider again the data set P depicted in Figure 4(a). The decisive subspaces for each maximal skyline group $SG(G,U)$ are shown in Figure 4(c). We observe that point p_1 is the only point that belongs to the decisive skyline set or, put differently, only p_1 has the full space ABC as its decisive subspace. Point p_3 has B as decisive subspace because it has the minimum value in B and no other point coincides with p_3 in B. In turn, this means that p_3 belongs to the skyline set independently of its values in the other dimensions. In this example, p_3 has the worst value of all points in dimension A. On the other hand, point p_2 has AB as decisive subspace and not A, even though it is subspace skyline in A. This is because it coincides with p_4 in A and they form a group $\{p_2, p_4\}$ in that subspace. Thus, p_2 could (in theory) be dominated in the full space by p_4, but it qualifies as a skyline point based on its combined values in the dimensions A and B. Still, the value of p_2 in dimension C does not influence whether p_2 belongs to the skyline set or not, thus this value can be arbitrarily high. Note that in this small example, the strict and relaxed decisive skyline points are the same.*

The above problem is different than the one studied in [18], which computes all skyline groups in order to find the skyline points in all subspaces.

4. Decisive Skyline Algorithm

A straightforward way to compute the decisive skyline set is to compute all maximum skyline groups and their decisive subspaces. Then, the points that belong to a group that have the full space as a decisive subspace can be

easily determined. Computing all maximum skyline groups requires evaluating all $2^m - 1$ subspace skyline queries and requires multiple disk accesses on the same data. We refer to this approach as *Naive*.

As we will show in the following, we develop an algorithm for computing the strict decisive skyline set with two salient features. First, our algorithm avoids evaluating all subspace skyline queries, and instead evaluates only $m + 1$ skyline queries. Second, assuming that data is indexed by a multidimensional index, we define an appropriate query that allows our algorithm to traverse the index at most once, retrieve a set of candidate points, and refine the result set in main-memory.

In Section 4.1, we show that the correct $\mathcal{DS}(P)$ of P can be produced by using only the $(m-1)$-dimensional subspace skyline sets. Moreover, we define the *enriched skyline* and prove that this set of points suffices to produce the strict decisive skyline set. Then, in Section 4.2, we propose an algorithm that exploits the pruning properties in order to compute the strict decisive skyline set in an efficient manner. Finally, in Section 4.3, we extend our approach for computing the relaxed decisive skyline set.

4.1. Pruning Properties

One important observation is that the strict decisive skyline points ($p \in \mathcal{DS}(P)$) are exactly those skyline points ($p \in \mathcal{S}(P)$) that for any $(m-1)$-dimensional subspace U they are either dominated ($p \notin \mathcal{S}_U(P)$) or share the same values in U with another data point p' ($p' =_U p$ and $p' \neq p$)[4].

Theorem 1. *A skyline point $p \in \mathcal{S}(P)$ belongs to the strict decisive skyline set $\mathcal{DS}(P)$, if there exists no $(m-1)$-dimensional subspace U in which p is subspace skyline and no other point $p' \neq p$ has identical values with p in U, i.e., $\nexists U \subset \mathcal{D}, |U| = m-1$ such that $p \in \mathcal{S}_U(P)$ and $\nexists p' : p' =_U p$ and $p' \neq p$.*

PROOF. (Sketch) Assume that there exists a strict decisive skyline point $p \in \mathcal{DS}(P)$ for which $\exists U \subset \mathcal{D}, |U| = m-1$ such that $p \in \mathcal{S}_U(P)$ and $\nexists p' : p =_U p'$ and $p \neq p'$. We distinguish two cases. First, let us assume that $\nexists p' \in P$ such that $p' = p$. Then, according to Definition 5, skyline point p forms a maximal skyline group in $\mathcal{D}$, i.e., $SG(\{p\}, \mathcal{D})$, and since p is a decisive skyline point $DecSub(p) = \{\mathcal{D}\}$. Based on Definition 6 this means that there exists no subspace V of $\mathcal{D}$ such that $p \in \mathcal{S}_V(P)$ and $\mathcal{O}(\{p\}, V) = \{p\}$. Nevertheless, it holds that $U \subset \mathcal{D}$ that $p \in \mathcal{S}_U(P)$ and $\mathcal{O}(\{p\}, U) = \{p\}$ (this is because $\nexists p' : p =_U p'$), which is a contradiction. Second, the case $\exists p' \in P$ such that $p' = p$ can be proved in a similar way. The main modification is that the maximal skyline group is defined based on the set of points that share the same values in $\mathcal{D}$.

From the above theorem, we derive that every strict decisive skyline point p is either dominated in U or there exists another point p' ($p' \neq p$) that shares the same values in the dimensions of U, but not in the full space. Obviously, in the latter case, p' is also a subspace skyline point in U.

To increase the efficiency of query processing, we are interested to identify a subset of P that is sufficient for computing $\mathcal{DS}(P)$ of P, in the sense that by taking into account only this subset of points we can decide whether a point belongs or not to $\mathcal{DS}(P)$. Motivated from the above discussion, we identify such a subset of P as the union of the skyline points $\mathcal{S}(P)$ and all subspace skyline points $\mathcal{S}_U(P)$ of any $(m-1)$-dimensional subspace $U \subset \mathcal{D}, |U| = m-1$. We denote this set of points $\mathcal{X}(P)$. The following theorem proves that $\mathcal{X}(P)$ is sufficient for computing the strict decisive skyline set of a data set P.

Theorem 2. *Given a data set P defined over an m-dimensional data space $\mathcal{D}$, the set of points: $\mathcal{X}(P) = (\mathcal{S}(P)) \cup (\cup_{\forall U \subset \mathcal{D}, |U|=m-1} \mathcal{S}_U(P))$ is sufficient for computing the strict decisive skyline set $\mathcal{DS}(P)$.*

PROOF. (Sketch) Based on Theorem 1, a skyline point p belongs to $\mathcal{DS}(P)$ if p is not a subspace skyline in any $(m-1)$-dimensional subspace U and there exists no other point p' with identical values in U. $\mathcal{X}(P)$ contains all skyline points, hence all potential candidates for the decisive skyline set. Further, since $\mathcal{X}(P)$ contains all $(m-1)$-dimensional subspace skyline points, we can easily determine if a skyline point p does not belong to any $(m-1)$-dimensional subspace skyline. If p belongs to the subspace skyline of a subspace U, then again we can use $\mathcal{X}(P)$ to determine the existence of another point p' with identical values to p in U, since both p and p' will belong to the subspace skyline of U ($p =_U p'$ and $p \in \mathcal{S}_U(P)$). Hence, $\mathcal{X}(P)$ suffices to check all cases of Theorem 1, and thus produce the strict decisive skyline set $\mathcal{DS}(P)$.

[4] We denote $p' =_U p$, if it holds that $\forall d_i \in U : p[i] = p'[i]$.

In order to compute the strict decisive skyline points we need the set of skyline points in $\mathcal{D}$ and all subspace skyline points in any $(m-1)$-dimensional subspace. However, retrieving from disk this set entails non-negligible cost, since in the simplest case one skyline query and m subspace skyline queries need to be processed and the index must be accessed multiple times. To avoid this processing overhead, we identify a super-set of this set that can be efficiently retrieved by traversing the index structure at most once.

Definition 9. *(Enriched skyline) A point $p \in P$ is said to* partially dominate *another point $q \in P$ on $\mathcal{D}$, if (1) on every dimension $d_i \in \mathcal{D}$, $p[i] \leq q[i]$; and (2) on at least two dimensions $d_j, d_k \in \mathcal{D}$, $p[j] < q[j]$ and $p[k] < q[k]$. The* enriched skyline *of P is the set of points $e\mathcal{S}(P) \subseteq P$ which are not partially dominated by any other point.*

The above definition assumes that the enriched skyline is defined on a data space that contains at least two dimensions, i.e., $|\mathcal{D}| \geq 2$. An interesting observation is that the enriched skyline uses a slightly modified definition of dominance, that can be supported by any skyline algorithm with marginal overhead, by simply changing the function used for point dominance.

Theorem 3. *The enriched skyline set is sufficient to compute the strict decisive skyline set $\mathcal{DS}(P)$.*

PROOF. It suffices to show that the enriched skyline set is a super-set of $\mathcal{X}(P)$. We will prove this by contradiction. Let p denote a point in $\mathcal{X}(P)$ that does not belong to the enriched skyline set. Since p does not belong to the enriched skyline set, there exists a point q that partially dominates p, i.e., $\forall d_i \in \mathcal{D} : q[i] \leq p[i]$ and $\exists d_j, d_k \in \mathcal{D}$ with $j \neq i$ and $k \neq i$: $q[j] < p[j]$ and $q[k] < p[k]$. Obviously, this means that q also dominates p in the full space. It also means that q dominates p in all $(m-1)$-dimensional subspace U (since any such subspace always contains at least one of the dimensions d_j, d_k). Hence, p does not belong to $\mathcal{S}(P)$ nor to any $\mathcal{S}_U(P)$, which means that $p \notin \mathcal{X}(P)$. This contradicts our assumption.

4.2. Algorithmic Description

Based on the aforementioned pruning properties, we design an efficient algorithm, called *Decisive Skyline Algorithm* and denoted as *DSA*, for computing the strict decisive skyline $\mathcal{DS}(P)$ of a set of points P. The main idea of our algorithm is to first compute the enriched skyline set, that is both easy to compute and sufficient to compute the decisive skyline points. Thereafter, DSA computes $\mathcal{DS}(P)$ by efficient processing of the underlying subspace skyline queries without the need to access the disk repeatedly.

One advantage of our algorithm is that any skyline algorithm can be easily modified to compute the enriched skyline set, by modifying the dominance test. In the following, we assume that the data is indexed by a multi-dimensional index and BBS algorithm [21] is used for the skyline computation. Note that the BBS returns the skyline points progressively and the correctness of BBS relies on the following property of skyline points: a point that has a larger distance cannot dominate a point that has a smaller or equal distance to the origin. The same property holds also for the enriched skyline set, therefore the correctness of BBS is not violated by changing the dominance test.

After retrieving the enriched skyline points, they are inserted in the main-memory R-tree. DSA processes m subspace skyline queries (of dimensionality $m-1$) using the main-memory R-tree, for excluding non-decisive skyline points from the already computed skyline set. Notice that the main-memory R-tree indexes only the enriched skyline points, therefore the execution of subspace skyline queries is more efficient compared to processing on the entire data set P on disk. Since decisive skyline subspaces are defined for groups of points that coincide in the subspace, it is important to process the points in groups. If the subspace skyline points are returned sorted based on a scoring function, this can be done easily by processing all points that have the same score in a batch. Recall that, a point may have as decisive subspace the full space, even if it is a subspace skyline, if there exists another subspace skyline point that shares the same values in all dimensions of the subspace.

The pseudocode describing DSA is shown in Algorithm 1. The algorithm takes as input a data set P indexed by an R-tree $\mathcal{R}$ and produces the strict decisive skyline set $\mathcal{DS}(P)$ as output. First, the modified BBS is executed on the R-tree that indexes P, and it populates the main-memory R-tree $\mathcal{M}$ with the enriched skyline points (and only those). In addition, the skyline set $\mathcal{S}(P)$ is retrieved (line 2). Notice that the two parameters of the $BBS()$ call in the pseudocode correspond to the index used by BBS and the subspace which is processed respectively. DSA exploits the main-memory R-tree internally used by BBS. Thus, DSA takes practically *for free* the index structure that is constructed by BBS during the skyline computation, thereby making the subsequent execution of subspace skyline queries extremely efficient. Moreover, as all decisive skyline points belong to the skyline set, we modify further BBS,

Algorithm 1 *Decisive Skyline Algorithm (DSA)*

input: The R-tree index $\mathcal{R}$ built on data set P
output: The decisive skyline set $\mathcal{DS}(P)$

1: $\mathcal{M} \leftarrow null$, $\mathcal{B} \leftarrow \emptyset$ {$\mathcal{M}$:main-memory R-tree, $\mathcal{B}$:buffer}
2: $(\mathcal{S}(P), \mathcal{M}) \leftarrow BBS(\mathcal{R}, \mathcal{D})$
3: **for** $i = 1...m$ **do**
4: $U \leftarrow \mathcal{D} - d_i$ {U:current subspace of $m-1$ dimensions}
5: $tmpDist \leftarrow -1$
6: **while** has next point $BBS(\mathcal{M}, U)$ **and** $\mathcal{S}(P) \neq \emptyset$ **do**
7: $q \leftarrow$ next point of $BBS(\mathcal{M}, U)$
8: **if** $Dist_U(q) = tmpDist$ **then**
9: $\mathcal{B} = \mathcal{B} \cup q$
10: **else**
11: **for all** $p \in \mathcal{B}$ **do**
12: **if** $(p \in \mathcal{S}(P))$ **and** $(\nexists p' \in \mathcal{B} : \forall d_k \in U\ p[k] = p'[k]$ and $p[i] \neq p'[i])$ **then**
13: $\mathcal{S}(P) \leftarrow \mathcal{S}(P) - p$
14: **end if**
15: **end for**
16: $\mathcal{B} = \{q\}$
17: **end if**
18: $tmpDist \leftarrow Dist_U(q)$
19: **end while**
20: **end for**
21: **return** $\mathcal{S}(P)$

so that only skyline points are reported as result ($\mathcal{S}(P)$), even though the main-memory R-tree indexes the enriched skyline points.

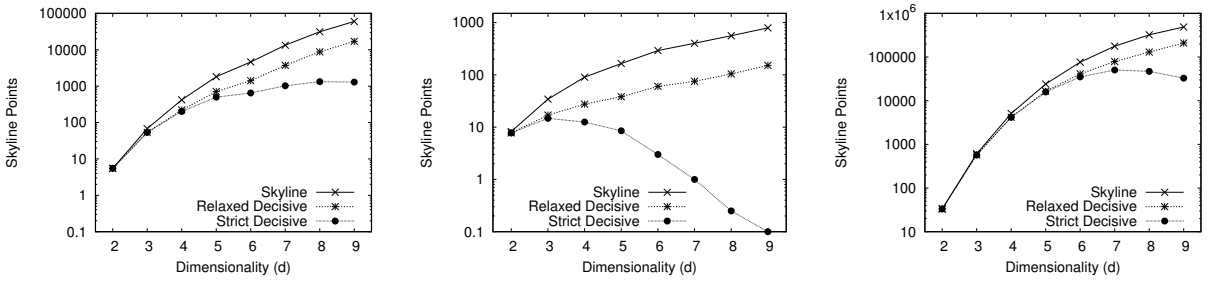
Then, DSA executes m subspace skyline queries on the main-memory R-tree $\mathcal{M}$ iteratively (lines 3-20). BBS returns the skyline points progressively in ascending order of their distances to the origin of the axes. For each $(m-1)$-dimensional subspace U, DSA exploits the progressive property of BBS (line 7) and places them in a buffer $\mathcal{B}$ (line 9). This guarantees that points with identical values in U are processed in a batch. Processing of a batch of points includes examining each point p in $\mathcal{B}$ and checking whether it is a candidate point (lines 11-15). If so, then we test if there exists another point p' with identical values on the $m-1$ dimensions of U, but different on the last dimension. If such a point p' does not exist, then point p can safely be discarded from the candidate list (line 13). The same procedure is repeated until all subspace skyline points are processed or the candidate list ($\mathcal{S}(P)$) gets empty (line 6).

4.3. Processing Relaxed Decisive Skyline Queries

In the following, we show how DSA can be used for computing the relaxed decisive skyline set $\widehat{\mathcal{DS}}(P)$. Recall that decisive skyline points also belong to the relaxed decisive skyline set ($\mathcal{DS}(P) \subseteq \widehat{\mathcal{DS}}(P)$). In order to retrieve the additional points of $\widehat{\mathcal{DS}}(P)$, we have to examine only skyline points (candidates) that belong to at least two $m-1$ dimensional subspace skyline sets. In this case, we have to further evaluate if these subspaces are decisive subspaces for the candidate. If a subspace is not decisive, we need to recursively examine its children subspaces in the lattice structure.

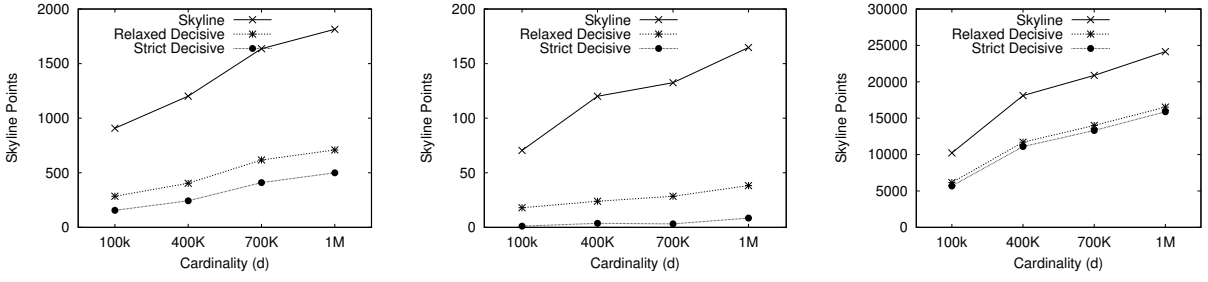
Based on this discussion, we modify DSA in order to retrieve the decisive subspaces too. In more detail, we do not discard points in line 13, but instead keep for each candidate point p a list of subspaces, namely those for which p is subspace skyline. At the end of the algorithm, we need to distinguish three cases for any candidate p:

- If the list of p is empty, then p is a decisive skyline point.
- Otherwise, if the subspaces in the list of p do not cover all dimensions, then p can be discarded.
- Otherwise, DSA is executed recursively for all subspaces in the list.



(a) Skyline points, UN (b) Skyline points, CO (c) Skyline points, AC

Figure 5: Cardinality of the decisive skyline set versus traditional skyline set for varying data dimensionality.



(a) Skyline points, UN (b) Skyline points, CO (c) Skyline points, AC

Figure 6: Cardinality of the decisive skyline set versus traditional skyline set for varying data cardinality.

During recursive operation, if p belongs to the decisive skyline set of a subspace or it is discarded, then we stop the recursion. After, finishing with the recursive DSA evaluations, we test if the subspaces for which p belongs to the decisive skyline set cover all dimensions. If this is the case, then the candidate is added to $\widehat{DS}(P)$.

5. Experimental Evaluation

In this section, we provide a detailed experimental study of the decisive skyline query. All algorithms are implemented in Java and the experiments run on a machine with 2x Intel Xeon X5650 Processors (2.66GHz), 128GB. Our code is publicly available[5].

Data sets. We employed synthetic data sets to examine different distributions, namely uniform (UN), correlated (CO) and anti-correlated (AC). The correlated (CO) and anti-correlated (AC) data sets were generated as described in [1]. For our experiments on synthetic data, we report the average results over 10 different instances of the data set. We generate the different instances by keeping the parameters fixed and changing the seeds of the random number generator. We adopt this approach in order to factor out the effects of randomization. We also used two real data sets. *NBA*[6] consists of 21,961 17-dimensional tuples representing a player's performance per year. We use different variants of this data set, by selecting dimensions from the following 6 criteria, namely minutes played (min), number of points scored (pts), offensive rebounds (oreb), defensive rebounds (dreb), assists (asts), and steals (stl). We also employ *DBLP* which contains data that reflect DBLP entries before 15/10/2008. We use the authors as points represented in a multidimensional space defined by the number of publications in selected conferences (dimensions).

Algorithms. We evaluate the DSA algorithm and compare its performance to Naive (described in the beginning of Section 4) for computing strict decisive skyline points. We also implement a state-of-the-art algorithm for skyline computation, namely BBS [21], to compare its performance to our approach. These algorithms use a disk-based R-tree with block size of 4KB and a buffer of size 100 nodes. Furthermore, we implement two representative skyline

[5] https://doi.org/10.5281/zenodo.8151919
[6] Available at: http://www.databasebasketball.com/

algorithms, namely dominance-based representative (*DoR*) [8] and distance-based representative (*DiR*) [11], in order to evaluate their ability to report decisive skyline points.

Parameters. We vary the dimensionality (m) of the data from 2 to 9, the cardinality (n) of the data set from 100K to 1M, and we test different data sets (UN,AC,CO,NBA,DBLP). Unless explicitly mentioned, the default setup is m=5, $|n|$=1M, and we employ the UN data set.

Methodology. First, we study the cardinality of the decisive skyline sets compare to the skyline set (Section 5.1). Apart from this quantitative study, we also include a qualitative study, by inspecting the decisive skyline points and demonstrating their advantages over skyline points in terms of balancing all criteria (Section 5.2). In addition, we evaluate performance of DSA by measuring the time required for computing the strict decisive skyline query (Section 5.3). For the experiments with representative skyline algorithms, we also measure the achieved recall of these algorithms in terms of retrieving strict decisive skyline points (Section 5.4).

5.1. Cardinality of the Decisive Skyline Query

In this section, we study the cardinality of the decisive skyline query for different data distributions, varying the number of dimensions and cardinality of the data sets. We compare the cardinality of $\mathcal{DS}(P)$ and $\widehat{\mathcal{DS}}(P)$ with the cardinality of traditional skyline set.

Figure 5 shows the size of the decisive skyline sets for increasing dimensionality and for three different data distributions, namely uniform (UN), correlated (CO), and anti-correlated (AC). We employ synthetic data sets containing 1M points each. In Figure 5(a), the data set follows a uniform data distribution. The size of $\mathcal{DS}(P)$ is comparable to the cardinality of the skyline set up to 3 dimensions, and after $|\mathcal{DS}(P)|$ grows slower than in the case of the skyline set. After 6 dimensions, the cardinality of the strict decisive skyline set remains practically stable, while the number of skyline points continues to increase exponentially. This verifies our claim that the skyline set contains a large number of points that balance only a subset of the given criteria, while the number of decisive skyline points remains constant or decreases after a certain number of dimensions, as it is harder to balance an increasing number of criteria. On the other hand, $|\widehat{\mathcal{DS}}(P)|$ increases more rapidly than $|\mathcal{DS}(P)|$, but still is significant smaller than the skyline set.

In Figure 5(b), the data set follows the correlated (CO) data distribution. The cardinality of $\mathcal{DS}(P)$ grows slower than the cardinality of the skyline set and for more than 3 dimensions, it starts to decrease. For more than 6 dimensions, the number of strict decisive skyline points is close to zero. This is because skyline points are close to the origin of the space in the case of correlated data. This means that in each dimension their value is either the minimum value or some value close to the minimum. In turn, this means that that the decisive subspace for such points will not be the full space, but some other subspace. Therefore, these skyline points will not qualify as strict decisive skyline points. Again, this experiment shows that there exist very few points that balance all criteria when the number of dimensions increases. By relaxing the decisive skyline set, more skyline points are included in the result set $\widehat{\mathcal{DS}}(P)$, indicating that many skyline points balance more than one subspace of the criteria due to the correlated nature of the data set.

In the case of anti-correlated (AC) data set (Figure 5(c)), the output size of all queries is significantly larger than UN or CO, due to the nature of the AC data generation, where the data points are generated in such a way that pairwise dominations are rare. When the dimensionality increases to 9 dimensions, the traditional skyline query returns almost the complete data set as result, which is clearly problematic for the user. However, notice that the strict decisive skyline decreases after 7 dimensions, even in the case of AC. This is strong evidence that the decisive skyline query succeeds to exclude many points that do not balance all criteria, even in the case of the challenging AC distribution.

Figure 6 compares the cardinality of traditional and decisive skyline sets when the size of the data set increases up to 1M points, for UN (Figure 6(a)), CO (Figure 6(b)) and AC (Figure 6(c)). Again, in all cases, the cardinality of the decisive skyline sets is much smaller than the cardinality of traditional skyline set.

5.2. Experiments with Real Data Sets

In the following, we perform skyline analysis on data extracted from DBLP, in order to discover researchers with significant number of publications on a combination of conferences. Major conferences from different research areas are selected as criteria that need to be balanced. In particular, we show the results of two queries, one posed with criteria {SIGMOD,PODS,CIKM} (Table 2) and another one with {SIGMOD,PODS,VLDB} (Table 3). We underline the strict decisive skyline points, while relaxed decisive skyline points are shown using bold. Each researcher is represented as a 3-dimensional point with values equal to the number of publications for each of the selected conferences, and higher values are preferable.

Id	Name	SIGMOD	PODS	CIKM	$DecSub()$
1	**Divyakant Agrawal**	**14**	**7**	**11**	{S,P,C}
2	Jeffrey F. Naughton	29	9	1	{S,P}
3	Amr El Abbadi	7	8	12	{P,C}
4	Jiawei Han	26	0	8	{S,C}
5	Dan Suciu	14	15	2	{P,C}
6	Serge Abiteboul	15	24	0	{S,P}
7	Michael J. Carey	36	3	0	{S}
8	Jeffrey D. Ullman	17	16	0	{S,P}
9	**Divesh Srivastava**	**28**	**10**	**3**	{S,C}, {P,C}
10	Yehoshua Sagiv	8	29	1	{P}
11	**Christos Faloutsos**	**19**	**4**	**8**	{S,P,C}
12	Raghu Ramakrishnan	28	14	1	{S,P}
13	Surajit Chaudhuri	33	8	0	{S,P}
14	Philip S. Yu	18	1	18	{C}
15	David J. DeWitt	33	1	1	{S,C}

Table 2
Skyline points of the DBLP data set for dimensions $\{SIGMOD, PODS, CIKM\}$. $\widehat{DS}(P)$: in bold, $DS(P)$: underlined.

Id	Name	SIGMOD	PODS	VLDB	$DecSub()$
1	Jeffrey F. Naughton	29	9	26	{S,P}
2	Michael J. Carey	36	3	26	{S}
3	**Serge Abiteboul**	**15**	**24**	**15**	{S,P}, {P,V}
4	David J. DeWitt	33	1	29	{S,V}
5	**Raghu Ramakrishnan**	**28**	**14**	**30**	{S,P} {S,V}
6	Jeffrey D. Ullman	17	16	3	{S,P}
7	H. V. Jagadish	26	5	35	{V}
8	Surajit Chaudhuri	33	8	26	{S,P}
9	Yehoshua Sagiv	8	29	8	{P}

Table 3
Skyline points of the DBLP data set for dimensions $\{SIGMOD, PODS, VLDB\}$. $\widehat{DS}(P)$: in bold.

Table 2 shows the skyline set for {SIGMOD,PODS,CIKM}. *Divyakant Agrawal* and *Christos Faloutsos* are the strict decisive skyline points and they balance nicely all criteria, compared to the remaining skyline points. By inspecting the result set, we observe that several researchers do not truly balance all given criteria (dimensions). Instead, they may balance subsets of the available dimensions only, but not all of them. For instance, *Jeffrey D. Ullman* nicely balances SIGMOD and PODS, but not CIKM. The same holds for both *Serge Abiteboul* and *Raghu Ramakrishnan*, who are also experts in data management. On the other hand, *Yehoshua Sagiv* is included in the result due to the extremely high number of PODS publications. The decisive skyline query manages to exclude these points from the result set, thus returning only points that truly balance all criteria. It is likely that if a user were interested in only a subset of the criteria, she would have posed a 2-dimensional query with only the criteria of interest instead. On the other hand, *Divesh Srivastava* will be included in the relaxed decisive skyline set, because he is a subspace skyline in subspaces {PODS,CIKM} and {SIGMOD,CIKM} that cover the full space, thus he manages to balance all given criteria.

Table 3 shows the skyline set for {SIGMOD,PODS,VLDB}. Some researchers balance subsets of the available dimensions only, but not all of them. Nevertheless, none of the researchers are in the decisive skyline set, because the criteria are highly correlated. The reader may notice that also *Serge Abiteboul* seems to balance all given criteria, but is not in the strict decisive skyline set. This is because he has as decisive subspaces {SIGMOD,PODS} and {PODS,VLDB}, but not the full space. Similarly, *Raghu Ramakrishnan* has as decisive subspaces {SIGMOD,PODS} and {SIGMOD,VLDB} and is excluded from the strict decisive skyline set. However, both *Serge Abiteboul* and *Raghu Ramakrishnan* are included in the relaxed decisive skyline set.

m	$\lvert\mathcal{S}(P)\rvert$	$\lvert\mathcal{DS}(P)\rvert$	$\lvert\widehat{\mathcal{DS}}(P)\rvert$	Criteria
3	35	10	13	pts,oreb,dreb
4	94	8	28	pts,oreb,dreb,asts
5	171	12	42	pts,oreb,dreb,asts,stl
6	251	3	45	pts,oreb,dreb,asts,stl,minutes

Table 4
Cardinality of decisive skyline set for varying number of criteria in NBA data set.

m	$\lvert\mathcal{S}(P)\rvert$	$\lvert\mathcal{DS}(P)\rvert$	$\lvert\widehat{\mathcal{DS}}(P)\rvert$	Criteria
4	94	8	28	pts,oreb,dreb,asts
4	137	17	38	pts,dreb,asts,stl
4	63	6	18	oreb,dreb,asts,stl
4	75	5	17	pts,oreb,asts,stl

Table 5
Cardinality of decisive skyline set for different selection of 4 criteria in NBA data set.

Id	Year	Name	pts	oreb	dreb
1	1975	**Julius Erving**	423	207	160
2	1982	**Magic Johnson**	829	176	47
3	1993	**David Robinson**	381	139	265
4	1993	**Mookie Blaylock**	789	212	44
5	1987	**Alvin Robertson**	557	243	69
6	1983	**Isiah Thomas**	914	204	33
7	1986	**Clyde Drexler**	566	204	71
8	1988	**Michael Jordan**	650	234	65
9	1985	**Paul Pressey**	623	168	71
10	1990	**Scottie Pippen**	511	193	93
11	1991	**John Stockton**	1126	244	22
12	1987	**Michael Jordan**	485	259	131
13	1989	**Hakeem Olajuwon**	234	174	376

Table 6
Decisive skyline points of NBA data set for criteria (pts,oreb,dreb). $\widehat{\mathcal{DS}}(P)$: in bold, $\mathcal{DS}(P)$: underlined.

In addition, we study the cardinality of the decisive skyline set as well as the quality of the retrieved decisive skyline points in the case of the NBA data set, which is typically used in skyline research. Table 4 compares the cardinality of the decisive skyline set to the skyline cardinality for increasing number of criteria. Clearly, the skyline cardinality increases with dimensionality and the skyline query returns too many points to the user to inspect, even for 4 dimensions. Instead, the size of $\mathcal{DS}(P)$ is more stable and not overwhelming for the user. The cardinality of $\widehat{\mathcal{DS}}(P)$ increases faster than $\mathcal{DS}(P)$, but is still smaller than the skyline set.

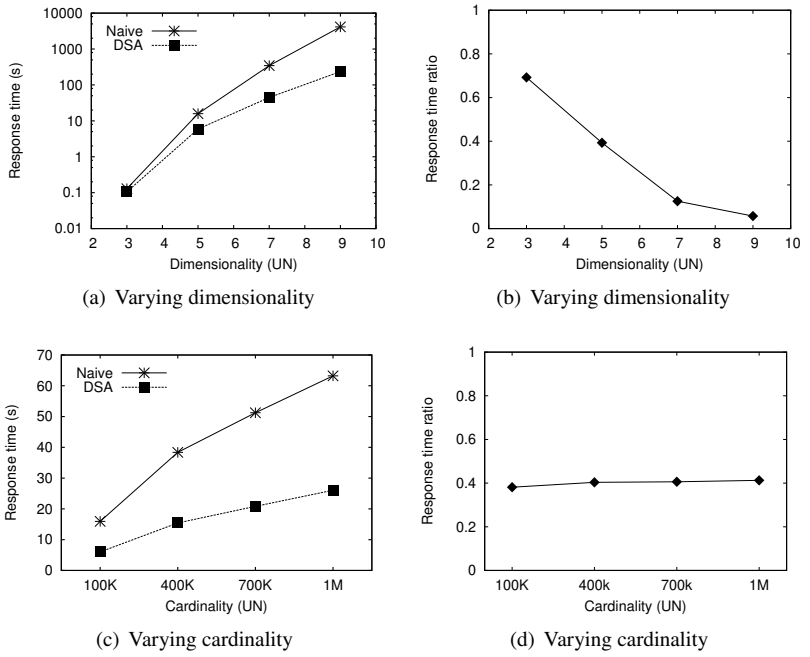
In Table 5, we fix the dimensionality to 4 and try different variations of the available criteria, in order to examine the cardinality of the result set under different correlations between the given criteria. The size of both sets $\mathcal{DS}(P)$ and $\widehat{\mathcal{DS}}(P)$ is influenced slightly and $\mathcal{DS}(P)$ always contains $6-12.5\%$ of the skyline points, while $\widehat{\mathcal{DS}}(P)$ contains $22-29\%$ of the skyline points. Thus, the strict decisive skyline set succeeds to restrict more the size of the result set, and at the same time ensures that all returned points balance all given criteria.

In Table 6, we depict the 10 decisive skyline points in the case of criteria: number of points scored (pts), offensive rebounds (oreb), and defensive rebounds (dreb). We also list the 3 points that belong to the relaxed decisive skyline set (points with IDs 11-13). We do not show all 35 skyline points in the interest of space. This list of decisive skyline points allows us to qualitatively judge the appropriateness of the decisive skyline query to retrieve players that truly balance all criteria. We observe that all players demonstrate interesting tradeoffs in the three dimensions. Also, the knowledgeable

reader can clearly see that all players are well-known legends of basketball, which further demonstrates the ability of the decisive skyline query to retrieve interesting points for the user.

5.3. Performance of Decisive Skyline Algorithm

In this section, we study the performance of the algorithms for processing the decisive skyline query. To this end, we compare the performance of DSA against the Naive algorithm. Then, we study the cost of computing the strict decisive skyline query (using DSA), compared with the cost of computing the skyline query (using the BBS algorithm). Although in this latter experiment DSA and BBS compute two different result sets, the experiment aims to answer the following interesting question: how much is the overhead of computing decisive skyline points, compared to the computation of the skyline set?



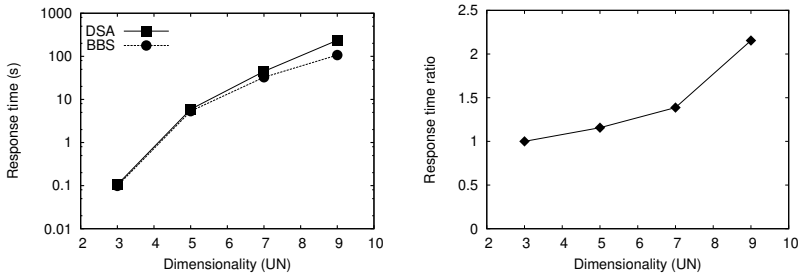
(a) Varying dimensionality (b) Varying dimensionality

(c) Varying cardinality (d) Varying cardinality

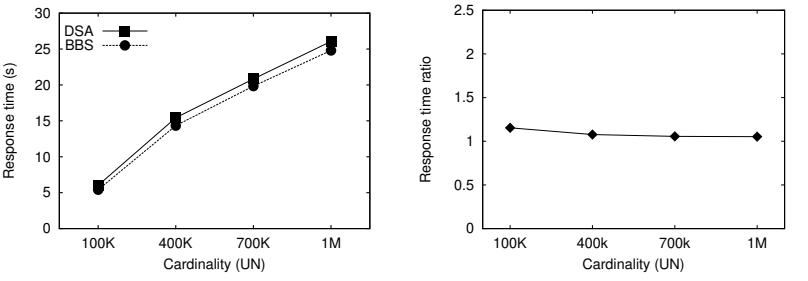
Figure 7: Performance of DSA versus Naive (response time to the left, response time ratio to the right).

Figure 7(a) and Figure 7(b) shows the response time and response time ratio for varying the data dimensionality. The response time ratio is calculated as the response time for DSA divided with the time for baseline, i.e., t_{DSA}/t_{Naive}, shown to better illustrate the relative performance. Practically, it shows how much faster DSA is compared to Naive as a fraction. DSA is more efficient than Naive in all setups. The performance of Naive drops significantly for increasing dimensionality, thus the advantage of DSA over Naive increases, as shown in Figure 7(a) and Figure 7(b). In fact, DSA is more than one order of magnitude faster than Naive for the largest dimensionality tested. Figure 7(c) and Figure 7(d) shows the performance of DSA and Naive for varying the cardinality of the data set. Clearly, DSA outperforms Naive and more importantly, the gain of DSA in terms of response time increases with the cardinality of the data set (Figure 7(c) and Figure 7(d)). Overall, this experiment shows the efficiency of DSA for larger data sets.

We also investigate the overhead of DSA compared to a state-of-the-art skyline algorithm (BBS). The results presented in Figure 8 show the response time for DSA and BBS for varying dimensionality and cardinality of the data sets. When increasing the dimensionality (Figure 8(a) and Figure 8(b)), the difference in response time between DSA and BBS is small and increases slowly. When varying the cardinality (Figure 8(c) and Figure 8(d)), the difference in time between DSA and BBS is small and constant, which is the expected result since the major impact in DSA is the number of subspace computations that is fixed in this setting. In summary, our finding is that DSA retrieves the strict decisive skyline set with a slightly increased cost compared to a state-of-the-art skyline algorithm that retrieves the traditional skyline set.

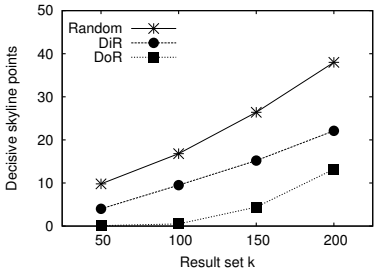


(a) Varying dimensionality (b) Varying dimensionality

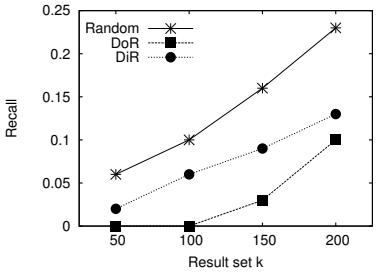


(c) Varying cardinality (d) Varying cardinality

Figure 8: Performance of DSA versus BBS (response time to the left, response time ratio to the right).



(a) Number of decisive skyline points



(b) Recall wrt. decisive skyline points

Algorithm	Response time (s)
DoR	34.9
DiR	353.3
DSA	6.0
BBS	5.6

(c) Response time

Figure 9: Comparison with skyline representative algorithms.

5.4. Comparison with Representative Skylines

Thereafter, we try to answer the following research question: can the set of decisive skyline points be obtained by existing algorithms proposed for representative skyline computation? To this end, we provide a comparison of our algorithm for computing the strict decisive skyline set against two well-known skyline representative algorithms, namely dominance-based representative (*DoR*) [8] and distance-based representative (*DiR*) [11]. As mentioned, on the one hand we compare qualitatively these approaches to ours, in order to see if these approaches manage to retrieve decisive skyline points. On the other hand, we also compare the performance of these approaches to ours, as an indication of the efficiency of our algorithm. DoR selects k skyline points, so that the number of points dominated by at least one of these k skyline points is maximized. DiR retrieves k representative skyline points, which are defined as the set of k points that minimize the maximum distance between a non-representative skyline point and its nearest representative. In addition, we use a *Random* algorithm that selects k representative skyline points from the skyline set at random.

Figure 9(a) shows the number of strict decisive skyline points retrieved by the representative skyline algorithms as the value of k increases. In this experiment, the size of strict decisive skyline set is 192 and the skyline cardinality is 952. As shown in the chart, both DoR and DiR fail to retrieve the decisive skyline points. In fact, even a random

selection of skyline points (Random) retrieves more decisive skyline points than DoR and DiR. This demonstrates that the existing skyline representative algorithms do not take into account the semantics of the skyline points and select completely different skyline points compared to our approach. Figure 9(b) shows the same result in terms of the recall that each representative skyline algorithm achieves, when using the strict decisive skyline points as correct result. As depicted in the chart, the recall of the representative skyline algorithms is very low, which demonstrates that these algorithms do not try (not even implicitly) to identify decisive skyline points.

Figure 9(c) shows the comparative performance of the skyline representative algorithms against DSA and BBS, using a fixed value of $k = 100$. In terms of time, DSA is significantly more efficient than both DoR and DiR, even though we employ the greedy variant of DiR which is very efficient and is able to report the representative skyline points, without computing the entire skyline set. For completeness, we mention that for smaller values, e.g. $k = 10$, DiR performs more efficiently than the other algorithms. All in all, this comparison shows that the performance of DSA is comparable to existing skyline representative algorithms and that the skyline representative skyline algorithms fail to retrieve the decisive skyline points.

6. Related Work

Skyline computation has received considerable attention in the database research community, and was first investigated in the context of databases by Börzsönyi et al. [1]. Afterwards, several approaches have been proposed in order to design efficient algorithms; we refer to [22, 23] and also to [24] for surveys. For example, Papadias et al. [21] propose a branch and bound algorithm to progressively output skyline points of a data set indexed by an R-Tree, with guaranteed minimum I/O cost. Theoretical results about skyline queries have also been investigated, related to external memory [25], parallel processing [26], and in a context of noisy comparisons [27].

Pei et al. [18, 28] discussed subspace skyline queries primarily from the view of *query semantics*. They solved the skyline membership query, namely why and in which subspaces an object belongs to the skyline, by using the notions of *skyline group* and of *decisive subspaces*. The paper introduces some very important semantics of the skyline points that have not been studied in detail in the related work. Differently to [18, 28] that focuses on finding the subspace skyline points of all subspaces, ours is the first paper that points out the importance of *skyline points that have the full space as decisive subspace*.

Many papers have focused on algorithms to support *subspace skyline retrieval* [21], motivated by the fact that different users may issue queries regarding different subsets of the available attributes, depending on their interests. In [29], *SUBSKY* is presented, which transforms the multi-dimensional data to one-dimensional values, and then indexes the data set with a B-Tree. The authors in [20] present a pre-processing approach, called *skycube*, which is defined as the union of all skyline points of all possible non-empty subspaces. For this purpose, they explore sharing strategies for answering multiple skyline queries, by identifying computational dependencies among skyline queries. Xia and Zhang [30] address the issue of supporting updates in skycube by introducing the *compressed skycube*. Efficient skycube computation is addressed in [31] by Lee and Hwang where the QSkycube approach is introduced, and extended later in [32]. Also, in [33], a subset-based approach is presented that exploits subspaces in order to compute more efficiently the skyline set using sorting-based algorithms.

It has been shown [4, 2, 3] that the expected number of skyline points is $\Theta(\ln^{m-1} n/(m-1)!)$ for a random independent data set, and this has motivated several approaches to *restricting the skyline cardinality*. One possibility is to restrict the cardinality of the result set, by selecting a subset of the entire skyline set. Towards this goal, Chan et al. [12] propose the *k-dominant* skyline query. The authors relax the idea of dominance to k-dominance, in order to increase the probability of one point dominating another point, thereby restricting the skyline cardinality. Nevertheless, k-dominant skyline queries return the skyline points that belong to all subspace skyline sets of dimensionality k, which is totally different than selecting the skyline points that have the full space as a decisive subspace. Another approach proposed by Lu et al. [14], *skyline ordering*, produces arbitrary size constrained skyline sets by employing skyline-based partitioning on the data set and applying a ranking method within each partition. However, depending on the size constraint, this method may also return non-skyline points to the user.

Selecting *representative skyline points* aims at retrieving exactly k points from the skyline set. In [8], the authors study the problem of selecting k skyline points, so that the number of points dominated by at least one of these k skyline points is maximized. In [11], an approach is presented for retrieving k representative skyline points, which are defined as the set of k points that minimize the maximum distance between a non-representative skyline point and its nearest representative. In [10], representative skylines are studied under the assumption that user preferences are expressed as

thresholds (this is also the reason why we did not include it in our experimental evaluation in Section 5.4). None of these approaches take into account the semantics of the skyline points, and as shown in our experimental evaluation they fail to discover the decisive skyline points.

Another family of approaches focuses on *ranking skyline points* and reporting the k highest ranked points. In [13], the authors introduce a metric called *skyline frequency*, to compare and rank the interestingness of data points based on how often they are returned in the skyline, when different subspaces are considered. Ranking based on skyline frequency obviously favors skyline points that are subspace skyline points, thus avoids to discover the points that balance all available criteria. In [15], a skyline ranking approach based on the dominance relationships between skyline points in all possible subspaces is proposed. Similar to skyline frequency, points that appear in many subspace tend to be ranked higher. Several approaches [17, 16] rely on user-defined functions or impose arbitrary preferences on some dimensions in order to rank the skyline points. In [17], the authors present the *Telescope algorithm* that ranks the skyline points by user-specified preferences on the available dimensions. In [16], a ranking approach based on user-defined regions that dominate all other regions is proposed. In contrast to those approaches, the decisive skyline set explores the semantics of the skyline points and does not rely on arbitrary user-defined scoring functions.

Finally, several useful extensions of skyline queries have been studied in the literature recently. In [34], an experimental survey is provided of the regret minimization query and other variants of this query, which try to combine skyline queries with restricted output-size queries such as top-k. Another approach that combines features of top-k queries with skyline queries is presented in [6]. Group skyline queries intend to retrieve groups of points that are not dominated by other groups [35, 36]. Probabilistic skyline queries have been studied both in centralized [37] and distributed settings [38]. Skyline processing over incomplete data is studied in [39]. ReSKY is a subarray skyline query processing algorithm targeting the context of array databases [40, 41]. A skyline algorithm that combines a CPU with a GPU context is presented in [42]. Skyline queries over encrypted data have been studied in [43, 44].

7. Conclusions

In this paper, we exploit the semantics of skyline points and propose the *decisive skyline query*. Capitalizing on the concept of decisive subspaces, we define two variants of the decisive skyline set that are subsets of the skyline set. Points belong to the decisive skyline set due to their values in *all* user-specified criteria and provide interesting trade-offs for the user. As a positive by-product and in contrast to skyline cardinality, the cardinality of the decisive skyline query is not affected in the same way by the data set dimensionality, thus leading to smaller result sets even for high-dimensional data. We studied the semantics of the proposed query and presented novel pruning properties that allow efficient computation of the strict decisive skyline set. Exploiting these properties, we designed a novel algorithm for processing decisive skyline queries that has minor overhead when compared to a state-of-the-art skyline algorithm. By means of an experimental evaluation, we demonstrated the performance of the proposed algorithm and also the characteristics of the decisive skyline points, thus providing empirical evidence that the decisive skyline query returns interesting points to the user.

As for future research directions, it would be of interest to propose other algorithms for discovering decisive skyline points, either by adapting existing algorithms for skyline queries or by introducing new algorithms. Also, studying the problem in different settings (e.g., main-memory context, online setting, distributed environment) is also of interest. Moreover, it would be interesting to study in more depth whether decisive skyline points can be used as skyline representatives.

References

[1] S. Börzsönyi, D. Kossmann, K. Stocker, The skyline operator, in: Proc. of ICDE, 2001, pp. 421–430.
[2] S. Chaudhuri, N. N. Dalvi, R. Kaushik, Robust cardinality and cost estimation for skyline operator., in: Proc. of ICDE, 2006, p. 64.
[3] P. Godfrey, Skyline cardinality for relational processing., in: Proc. of FoIKS, 2004, pp. 78–97.
[4] Z. Zhang, Y. Yang, R. Cai, D. Papadias, A. Tung, Kernel-based skyline cardinality estimation, in: Proc. of SIGMOD, 2009, pp. 509–522.
[5] P. Ciaccia, D. Martinenghi, Reconciling skyline and ranking queries, Proc. VLDB Endow. 10 (11) (2017) 1454–1465.
[6] K. Mouratidis, K. Li, B. Tang, Marrying top-k with skyline queries: Relaxing the preference input while producing output of controllable size, in: Proc. of SIGMOD, 2021, pp. 1317–1330.
[7] M. Xie, R. C. Wong, A. Lall, An experimental survey of regret minimization query and variants: bridging the best worlds between top-k query and skyline query, VLDB J. 29 (1) (2020) 147–175.
[8] X. Lin, Y. Yuan, Q. Zhang, Y. Zhang, Selecting stars: the k most representative skyline operator., in: Proc. of ICDE, 2007.

[9] M. Magnani, I. Assent, M. L. Mortensen, Taking the big picture: representative skylines based on significance and diversity, VLDB J. 23 (5) (2014) 795–815.
[10] A. D. Sarma, A. Lall, D. Nanongkai, R. J. Lipton, J. J. Xu, Representative skylines using threshold-based preference distributions, in: Proc. of ICDE, 2011, pp. 387–398.
[11] Y. Tao, L. Ding, X. Lin, J. Pei, Distance-based representative skyline, in: Proc. of ICDE, 2009, pp. 892–903.
[12] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, Z. Zhang, Finding k-dominant skylines in high dimensional space., in: Proc. of SIGMOD, 2006, pp. 503–514.
[13] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, Z. Zhang, On high dimensional skylines., in: Proc. of EDBT, 2006, pp. 478–495.
[14] H. Lu, C. S. Jensen, Z. Zhang, Flexible and efficient resolution of skyline query size constraints, IEEE TKDE (2011).
[15] A. Vlachou, M. Vazirgiannis, Ranking the sky: Discovering the importance of skyline points through subspace dominance relationships, DKE 69 (9) (2010) 943–964.
[16] I. Bartolini, P. Ciaccia, V. Oria, M. T. Özsu, Flexible integration of multimedia sub-queries with qualitative preferences, Multimedia Tools Appl. 33 (3) (2007) 275–300.
[17] J. Lee, G. won You, S. won Hwang, Personalized top-k skyline queries in high-dimensional space, Information Systems 34 (1) (2009) 45–61.
[18] J. Pei, W. Jin, M. Ester, Y. Tao, Catching the best views of skyline: A semantic approach based on decisive subspaces., in: Proc. of VLDB, 2005, pp. 253–264.
[19] A. Vlachou, C. Doulkeridis, J. B. Rocha-Junior, K. Nørvåg, On decisive skyline queries, in: Proc. of DAWAK, 2022, pp. 61–73.
[20] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, Q. Zhang, Efficient computation of the skyline cube., in: Proc. of VLDB, 2005, pp. 241–252.
[21] D. Papadias, Y. Tao, G. Fu, B. Seeger, Progressive skyline computation in database systems., ACM TODS 30 (1) (2005) 41–82.
[22] J. Chomicki, P. Ciaccia, N. Meneghetti, Skyline queries, front and back, SIGMOD Record 42 (3) (2013) 6–18.
[23] K. Hose, A. Vlachou, A survey of skyline processing in highly distributed environments, VLDB J. 21 (3) (2012) 359–384.
[24] C. Kalyvas, T. Tzouramanis, A survey of skyline query processing, CoRR abs/1704.01788 (2017).
[25] C. Sheng, Y. Tao, On finding skylines in external memory, in: Proc. of PODS, 2011, pp. 107–116.
[26] F. N. Afrati, P. Koutris, D. Suciu, J. D. Ullman, Parallel skyline queries, in: Proc. of ICDT, 2012, pp. 274–284.
[27] B. Groz, T. Milo, Skyline queries with noisy comparisons, in: Proc. of PODS, 2015, pp. 185–198.
[28] J. Pei, Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, Q. Zhang, Towards multidimensional subspace skyline analysis, ACM TODS 31 (4) (2006) 1335–1381.
[29] Y. Tao, X. Xiao, J. Pei, Efficient skyline and top-k retrieval in subspaces, IEEE TKDE 19 (8) (2007) 1072–1088.
[30] T. Xia, D. Zhang, Refreshing the sky: the compressed skycube with efficient support for frequent updates., in: Proc. of SIGMOD, 2006, pp. 491–502.
[31] J. Lee, S. Hwang, Qskycube: Efficient skycube computation using point-based space partitioning, PVLDB 4 (3) (2010) 185–196.
[32] J. Lee, S. Hwang, Toward efficient multidimensional subspace skyline computation, VLDB J. 23 (1) (2014) 129–145.
[33] D. H. Li, Subset approach to efficient skyline computation, in: Proceedings 26th International Conference on Extending Database Technology (EDBT), 2023, pp. 391–403.
[34] M. Xie, R. C.-W. Wong, A. Lall, An experimental survey of regret minimization query and variants: bridging the best worlds between top-k query and skyline query, The VLDB Journal 29 (1) (2020) 147–175.
[35] J. Liu, L. Xiong, J. Pei, J. Luo, H. Zhang, Finding pareto optimal groups: Group-based skyline, PVLDB 8 (13) (2015) 2086–2097.
[36] X. Han, J. Wang, J. Li, H. Gao, Efficient computation of g-skyline groups on massive data, Information Sciences 587 (2022) 300–322.
[37] J. Pei, B. Jiang, X. Lin, Y. Yuan, Probabilistic skylines on uncertain data, in: Proc. of VLDB, 2007, pp. 15–26.
[38] A. Kuo, H. Chen, L. Tang, W. Ku, X. Qin, Probsky: Efficient computation of probabilistic skyline queries over distributed data, IEEE TKDE 35 (5) (2023) 5173–5186.
[39] J. He, X. Han, Efficient skyline computation on massive incomplete data, Data Sci. Eng. 7 (2) (2022) 102–119.
[40] D. Choi, H. Yoon, Y. D. Chung, Subarray skyline query processing in array databases, in: Proc. of SSDBM, 2021, pp. 37–48.
[41] D. Choi, H. Yoon, Y. D. Chung, Resky: Efficient subarray skyline computation in array databases, Distributed Parallel Databases 40 (2-3) (2022) 261–298.
[42] J. C. Romero, A. G. Navarro, A. Rodríguez, R. Asenjo, Skyflow: Heterogeneous streaming for skyline computation using FlowGraph and SYCL, Future Gener. Comput. Syst. 141 (2023) 269–283.
[43] S. Bothe, A. Cuzzocrea, P. Karras, A. Vlachou, Computing skylines over encrypted data in cloud environments, in: Proc. of SEBD, 2016, pp. 222–229.
[44] A. Cuzzocrea, P. Karras, A. Vlachou, Effective and efficient skyline query processing over attribute-order-preserving-free encrypted data in cloud-enabled databases, Future Gener. Comput. Syst. 126 (2022) 237–251.

Akrivi Vlachou received the BSc and MSc degrees from the Department of Computer Science and Telecommunications of University of Athens in 2001 and 2003, respectively, and the PhD degree in 2008 from the Athens University of Economics and Business (AUEB). She is currently an Associate Professor at the University of Aegean. Her research interests include query processing and data management in large-scale distributed systems.

Christos Doulkeridis received the BSc degree in electrical engineering and computer science from the National Technical University of Athens and the MSc and PhD degrees in Information Systems from the Department of Informatics of Athens University of Economics and Business. He is currently an Associate Professor in the Department of Digital Systems of the University of Piraeus. His research interests include parallel and distributed data management, and data analytics. Further details concerning his work can be found in https://www.ds.unipi.gr/prof/cdoulk/

João B. Rocha-Junior is Professor at the Department of Exact Sciences at the State University of Feira de Santana (UEFS). He received his M.Sc. in Computer Science from the Federal University of Pernambuco (UFPE) and his PhD from the Norwegian University of Science and Technology (NTNU). His research interests include spatial keyword query processing, processing of preference queries, distributed query processing, spatial databases, and information retrieval.

Kjetil Nørvåg received his MSc and PhD degrees from the Norwegian University of Science and Technology (NTNU). He is currently Professor in the Department of Computer Science at NTNU. His research interests include distributed and parallel database systems, query processing, information retrieval, and Big Data in general. Further details concerning his work can be found in https://www.ntnu.no/ansatte/noervaag