Group-Based Skyline for Pareto Optimal Groups

Jinfei Liu, Li Xiong, Jian Pei, *Fellow, IEEE*, Jun Luo, Haoyu Zhang, and Wenhui Yu

多标准

帕累托最优的G-Skyline

Abstract—Skyline computation, aiming at identifying a set of skyline points that are not dominated by any other point, is particularly useful for multi-criteria data analysis and decision making. Traditional skyline computation, however, is inadequate to answer queries that need to analyze not only *individual* points but also *groups* of points. To address this gap, we generalize the original skyline definition to the novel group-based skyline (G-Skyline), which represents Pareto optimal groups that are not dominated by other groups. In order to compute G-Skyline groups consisting of s points efficiently, we present a novel structure that represents the points in a directed skyline graph and captures the dominance relationships among the points based on the first s skyline layers. We propose efficient algorithms to compute the first s skyline layers. We then present two heuristic algorithms to efficiently compute the G-Skyline groups: the point-wise algorithm and the unit group-wise algorithm, using various pruning strategies. We observe that the number of G-Skyline groups of a dataset can be significantly large, we further propose the top-k representative G-Skyline groups based on the number of dominated points and the number of dominated groups and present efficient algorithms for computing them. The experimental results on the real NBA dataset and the synthetic datasets show that G-Skyline is interesting and useful, and our algorithms are efficient and scalable.

1.提出了一种新的结构：表示有向Skyline图中的点，并基于前s个Skyline层捕获点之间的优势关系。2.提出了有效的算法来计算前s个天际线层。3.提出了两种启发式算法来有效地计算G-Skyline组：逐点算法和逐单元组算法。4.根据支配点、组的数量提出了Top-K G-Skyline。

Index Terms—Skyline, group, top-k, unit, domination

1 Introduction

Skyline, also known as *Maxima* in computational geometry or *Pareto* in business management field, is important for many applications involving multi-criteria decision making. Assume that we have a dataset of n points, referred to as P. Each point p of d real-valued attributes can be represented as a d-dimensional point $(p[1], p[2], \ldots, p[d]) \in \mathbb{R}^d$ where $p[i]$ is the ith attribute of p. Given two points $p = (p[1], p[2], \ldots, p[d])$ and $p' = (p'[1], p'[2], \ldots, p'[d])$ in $\mathbb{R}^d$, p dominates p' if for every i, $p[i] \leq p'[i]$ and for at least one i, $p[i] < p'[i]$ $(1 \leq i \leq d)$. Given the set of points P, the skyline is defined as the set of points that are not dominated by any other point in P. In other words, the skyline represents the *best points* or Pareto optimal solutions from the dataset since the points within the skyline cannot dominate each other.

Fig. 1a illustrates a dataset $P = \{p_1, p_2, \ldots, p_{11}\}$, each representing a hotel with two attributes: the distance to the destination and the price. Fig. 1b shows the corresponding points in the two dimensional space where the x and y coordinates correspond to the attributes of distance to the destination and price, respectively. We can see that $p_3(14, 340)$ dominates $p_2(24, 380)$ as an example of dominance. The skyline of the dataset contains p_1, p_6, and p_{11}. Suppose the organizers of a conference need to reserve *one* hotel considering both distance to the conference destination and the price for participants, the skyline offers a set of best options or Pareto optimal solutions with various tradeoffs between distance and price: p_1 is the nearest to the destination, p_{11} is the cheapest, and p_6 provides a good compromise of the two factors.

Motivation. While the skyline definition has been extended with different variants and the skyline computation problem for finding the skyline of a given dataset has been studied extensively in recent years, most existing works focus on skyline consisting of *individual* points. One important problem that has been surprisingly neglected to the large extent is the need to find groups of points that are not dominated by others as many real-world applications may require the selection of *a group of* points.

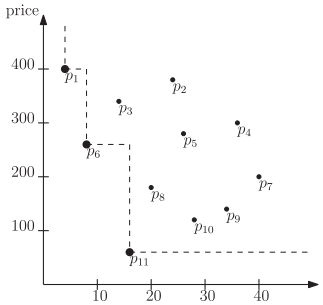
Hotels Example. Consider our hotel example again, suppose the organizers need to reserve *a group of* hotels (instead of one) considering both distance to the conference destination and the price for participants. In contrast to the traditional skyline problem which finds Pareto optimal solutions where each solution is a single point, we are interested in finding Pareto optimal solutions where each solution is a group of points. One may use the traditional skyline definition, and return all subsets from the skyline points p_1, p_6, and p_{11}. If the desired group size is 2, group $\{p_1, p_6\}$, $\{p_1, p_{11}\}$, and $\{p_6, p_{11}\}$ can be returned. However, we show that this definition does not capture all the best groups. For example, $\{p_{11}, p_{10}\}$ should clearly be considered a Pareto optimal group to users who use price as the main criterion, e.g., PhD students with low travel budget, since p_{11} provides the best price and p_{10} the second best price. Note that p_{10} is only second best to p_{11} which is also part of the group, hence no other groups are better than this group in terms of price. As another example, $\{p_6, p_3\}$ also presents a Pareto optimal group, as both p_6 and p_3 provide a good

- *J. Liu and L. Xiong are with Emory University, Atlanta, GA 30322 USA. E-mail: {jinfei.liu, lxiong}@emory.edu.*
- *J. Pei is with Simon Fraser University, Burnaby, BC V5A 1S6, Canada, and also with the JD.com, Beijing 101111, China. E-mail: jpei@cs.sfu.ca.*
- *J. Luo is with the Machine Intelligence Center, Lenovo, Hong Kong. E-mail: jluo1@lenovo.com.*
- *H. Zhang is with Indiana University Bloomington, Bloomington, IN 47405 USA. E-mail: hz30@umail.iu.edu.*
- *W. Yu is with Tsinghua University, Beijing 100084, China. E-mail: yuwh16@mails.tsinghua.edu.cn.*

Manuscript received 25 Feb. 2019; revised 10 July 2019; accepted 12 Dec. 2019. Date of publication 17 Dec. 2019; date of current version 3 June 2021. (Corresponding author: Jinfei Liu.)
Recommended for acceptance by H. Lee.
Digital Object Identifier no. 10.1109/TKDE.2019.2960347

hotel	distance	price
p_1	4	400
p_2	24	380
p_3	14	340
p_4	36	300
p_5	26	280
p_6	8	260
p_7	40	200
p_8	20	180
p_9	34	140
p_{10}	28	120
p_{11}	16	60

(a)



(b)

Fig. 1. A skyline example of hotels.

tradeoff and no other groups are better than this group considering both price and distance. On the other hand, group $\{p_3, p_8\}$ is not a best group because $p_{11}(p_6)$ is better than $p_8(p_3)$, i.e., group $\{p_3, p_8\}$ is dominated by group $\{p_6, p_{11}\}$.

NBA Example. Consider another real example with NBA players. Table 1 shows the top five players on attribute PTS (Points). For other attributes, please see the experimental section for detailed explanations. Suppose the coaches of NBA teams need to choose five players to compose a team. While the traditional skyline will compute *best players* that are not dominated by other players, we need to compute *best teams* that are not dominated by other teams. For example, a coach may prefer PTS as the main selection criteria when building a team in order to maximize the overall points that can be earned by the team. In this case, the top five players on PTS, {Michael Jordan, Anthony Davis, Kyrie Irving, Allen Iverson, Jerry West}, should be considered a best team. However, if we only build groups from skyline players, this team will not be captured because Kyrie Irving is not a skyline player being dominated by Anthony Davis. In essence, taking only skyline players will not capture those teams which may include non-skyline players who are only dominated by another player in the team but are not dominated by any other players outside the team. In summary, we argue that there is a need to define a group skyline notion for group-based decision making such that we can find Pareto optimal groups.

Contributions. In this paper, we formally define a novel group-based skyline, *G-Skyline*, for finding Pareto optimal groups. In order to find the best groups, i.e., groups not dominated by other groups, we will first define the dominance relationship between groups, group dominance. Given two different groups G and G' with s points, we say G g-dominates G', if for any point p'_i in G', we can find a distinct point p_i in G, such that p_i dominates p'_i or $p_i = p'_i$, and for at least one i, p_i dominates p'_i. The G-Skyline are those groups that are not g-dominated by any other group with same size. Intuitively, if we consider the points in each group as a set of dimensions orthogonal to the attributes of each point, the definition of G-Skyline groups with the group dominance is in spirit similar to skyline definition, in that a group is a G-Skyline group if no permutation of any other group exists that is better for at least one point and at least as good for every other point.

G-Skyline not only captures groups of points from traditional skyline points but also groups that may contain non-skyline points. Back to our hotel example, $\{p_{11}, p_{10}\}$ is a G-Skyline group as we discussed earlier even though p_{10} is not a skyline point. Group $\{p_6, p_3\}$ is also a G-Skyline. On the other hand, group $\{p_1, p_3\}$ is not as it is dominated by $\{p_1, p_6\}$. Group $\{p_3, p_8\}$ is also not as it is dominated by $\{p_6, p_{11}\}$. In summary, the G-Skyline in this example consist of all groups composed of skyline points, $\{p_1, p_6\}$, $\{p_1, p_{11}\}$, $\{p_6, p_{11}\}$, as well as groups that contain non-skyline points, $\{p_6, p_3\}$, $\{p_{11}, p_8\}$, and $\{p_{11}, p_{10}\}$.

TABLE 1
Top Five Players on Attribute PTS

Player	PTS	REB	AST	STL	BLK
Michael Jordan	33.4	6.4	5.7	2.1	0.9
Anthony Davis	30.5	8.5	2	1.5	3
Kyrie Irving	30	3	2	1	1
Allen Iverson	29.7	3.8	6	2.1	0.2
Jerry West	29.1	5.6	6.3	0	0
...	...	...	...	...	...

It's non-trivial to solve G-Skyline problem efficiently. To find s-point G-Skyline groups from n points, there can be $\binom{n}{s}$ different possible groups. Unfortunately, the G-Skyline problem is significantly different from the traditional skyline problem, to the extent that algorithms for the latter are inapplicable. A brute force solution is to enumerate all $\binom{n}{s}$ possible groups, then for each group, to compare it with all other groups to determine whether it cannot be dominated. So there are $\binom{n}{s}^2$ comparisons. For each comparison, there are $s!$ possible permutations of the points, and for each permutation, it requires s comparisons. Therefore, the time complexity is in the order of $O(\binom{n}{s}^2 \times s! \times s)$.

In this paper, we present a novel structure that represents the points in a directed skyline graph and captures all the dominance relationship among the points based on the notion of *skyline layers*. Using the directed skyline graph, the G-Skyline problem can be formulated as the classic search problem in a set enumeration tree. We exploit the properties of G-Skyline groups and propose two algorithms with efficient pruning strategies to compute G-Skyline groups.

We briefly summarize our contributions as follows.

- For the first time, we generalize the original skyline definition (for individual points) to permutation group-based skyline (for groups) which is useful for finding Pareto optimal groups in practical applications.
- Given the directed skyline graph, we present two efficient algorithms: the point-wise and the unit group-wise algorithms, to efficiently compute G-Skyline groups. We introduce a novel notion of unit-group for each point which represents the minimum number of points that have to be included with the point in a G-Skyline. Both algorithms employ efficient pruning strategies exploiting G-Skyline properties.
- One potential drawback of the G-Skyline groups is that the number of groups satisfying the definition can be significantly large, especially for anti-correlated datasets; and there is no priority among these groups, which limits the usage of G-Skyline in practice. In this paper, we define top-k representative G-Skyline groups based on the number of dominated points ($topkG^p$) and the number of dominated

groups ($topkG^g$) and present efficient algorithms for computing them.
- We conduct comprehensive experiments on real and synthetic datasets. The experimental results show that G-Skyline is interesting and useful, and our proposed algorithms are efficient and scalable.

Organization. The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 introduces our G-Skyline definitions as well as their properties. Two algorithms for finding G-Skyline groups based on the directed skyline graph are discussed in Section 4. We show how to choose top-k representative G-Skyline groups in Section 5. We report the experimental results and findings for performance evaluation in Section 6. Section 7 concludes the paper.

第一段全关注skyline时间复杂度 可以不管

2 Related Work

The problem of computing skyline (Maxima) is a fundamental problem in computational geometry field because the skyline is an interesting characterization of the boundary of a set of points. The skyline computation problem was first studied in computational geometry [15] which focused on the worst-case time complexity. Kirkpatrick and Seidel [13], [19] proposed output-sensitive algorithms achieving $O(n\log v)$ in the worst-case, where v is the number of skyline points which is far less than n in general. Several works [2], [3], [4], [8] in both computational geometry and database fields focused on how to achieve the best average-case time complexity. For a detailed survey both for worst-case and average-case, please see [10]. Since the introduction of the skyline operator by Börzsönyi *et al.* [4], skyline has been extensively studied in the database field [6], [7], [9], [11], [14], [20], [21], [22], [23], [28], [29], [31], [34], [37], [40].

Group-based Skyline. The most related works to our group-based skyline are [12], [16], [24], [39]. [12], [16], [39] formulated and investigated the problem of computing skyline groups. However, the notion of dominance between groups in these works is defined by the dominance relationship between an "aggregate" or "representative" point of each group. More specifically, they calculate for each group a single aggregate point, whose attribute values are aggregated over the corresponding attribute values of all points in the group. The groups are then compared by their aggregate points using traditional point dominance. While many aggregate functions can be considered in calculating aggregate points, they focus on several functions commonly used in database applications, such as, SUM, MIN, and MAX. In addition to the fact that it is difficult to choose a good or meaningful function, more importantly, it will not capture all the Pareto optimal groups. This is essentially similar to the multi-objective optimization or multi-attribute skyline problem where an aggregate function, such as weighted average, can be used to combine the multiple criteria to find a single optimal solution, but it also fails to capture all the Pareto optimal solutions. The work in [24] also defines a group dominance notion. However, their definition is based on the uncertain skyline definition by Pei *et al.* [28]. In our work, we define a deterministic dominance relationship between two groups in order to find "optimal" groups of objects.

This paper is an extended version of our previous work [18] which first proposed the G-Skyline groups and the algorithms for computing them. Wang et al. [36] and [38] proposed enhanced algorithms with better efficiency. The definition in [18] can result in a large number of G-Skyline groups, especially for anti-correlated dataset, which limits its usage in practice. In this paper, we extend [18] and define top-k representative G-Skyline groups based on the number of dominated points ($topkG^p$) and the number of dominated groups ($topkG^g$) and present efficient algorithms for computing them.

Representative Skyline. Since the number of skyline points of a dataset based on traditional skyline definition can be significantly large, especially for anti-correlated datasets, representative skyline has been proposed to reduce the output skyline size. The existing works can be classified into two categories, distance-based representative skyline [5], [25], [26], [27], [33] and dominance-based representative skyline [1], [17], [32].

Tao *et al.* [33] proposed the first distance-based representative skyline. Their goal is to minimize the distance between representative skyline and non-representative skyline, which is very similar to the classic k-center problem. Another classic representative skyline definition was first given by Lin *et al.* [17]. The goal is to find k skyline points to maximize the number of points or area that can be dominated by any of the selected k skyline points. In addition, [35] studied the representative skyline in distributed systems. [30] formulated the problem of displaying k representative skyline points such that the probability that a random user would click on one of them is maximized.

The most related work to top-k representative G-Skyline groups is [41] which proposed the top-k representative G-Skyline groups based on the number of dominated points. In this paper, we present a much more efficient algorithm with an order of magnitude improvement. Furthermore, we propose a top-k representative G-Skyline groups definition based on the number of dominated groups, which is aligned with the group dominance definition used for G-Skyline groups.

3 G-Skyline Definitions

In this section, we introduce our G-Skyline definition and related concepts as well as their properties which will be used in our algorithm design. For reference, a summary of notations is given in Table 2.

Definition 1 (Skyline). *Given a dataset P of n points in d-dimensional space. Let p and p' be two different points in P, p dominates p', denoted by $p \prec p'$, if for all i, $p[i] \leq p'[i]$, and for at least one i, $p[i] < p'[i]$, where $p[i]$ is the ith dimension of p and $1 \leq i \leq d$. The skyline points are those points that are not dominated by any other point in P.*

G-Skyline. The key of skyline is that it consists of all the "best" points that are not dominated by other points. While a linear weighted sum function can be used to combine all the attribute values of each point as a scoring function to find the best points, the relative preferences (weights) for different attributes are not known in advance. The skyline essentially covers all the best points on all linear functions.

TABLE 2
The Summary of Notations

Notation	Definition
P	dataset of n points
n	number of points in P
s	group size/ skyline layers
d	number of dimensions
p_i	ith point in $P, 1 \leq i \leq n$
$\mathbb{S}_s$	the number of points in the first s layers
$p_i[j]$	jth dimensional value of $p_i, 1 \leq j \leq d$
$P \backslash layer_i$	points in P but not in $layer_i$
$p \preceq p'$	p dominates or equals to p'
G-Skyline(i)	G-Skyline group with i points
$p_i.layer$	the skyline layer of p_i
$\|S\|_{p(u)}$	the point(unit group) size of set S
$\mathbb{S}_s$	the number of points in the first s skyline layers
$NP(p_i)$	number of points dominated by point p_i
$NP(G_i)$	number of points dominated by Group G_i
$NG(G_i)$	number of groups dominated by Group G_i

Following this notion, in order to find all the "best" groups of points that are not dominated by other groups, we introduce group dominance definition as follows.

Definition 2 (Group Dominance). *Given a dataset P of n points in a d-dimensional space. Let $G = \{p_1, p_2, \ldots, p_s\}$ and $G' = \{p'_1, p'_2, \ldots, p'_s\}$ be two different groups with s points of P, we say group G* g-dominates *group G', denoted by $G \prec_g G'$, if we can find two permutations of the s points for G and G', $G = \{p_{u_1}, p_{u_2}, \ldots, p_{u_s}\}$ and $G' = \{p'_{v_1}, p'_{v_2}, \ldots, p'_{v_s}\}$, such that $p_{u_i} \preceq p'_{v_i}$ for all i $(1 \leq i \leq s)$ and $p_{u_i} \prec p'_{v_i}$ for at least one i.*

Given the group dominance definition, we define group-based skyline, G-Skyline, as follows.

Definition 3 (G-Skyline). *The s-point G-Skyline consists of those groups with s points that are not g-dominated by any other group with same size.*

Example 1. Consider the dataset in Fig. 1 and $s = 3$. For group $G = \{p_8, p_{10}, p_{11}\}$ and group $G' = \{p_4, p_5, p_7\}$, G g-dominates G' because we can find two permutations, $G = \{p_8, p_{10}, p_{11}\}$ and $G' = \{p_5, p_4, p_7\}$ such that $p_8 \prec p_5$, $p_{10} \prec p_4$, and $p_{11} \prec p_7$. Therefore, $G' = \{p_4, p_5, p_7\}$ is not a G-Skyline group. G is one of the G-Skyline groups as no other group with 3 points can g-dominate G.

Next, we present a few properties of G-Skyline groups and related concepts that will be used in our algorithm design for computing G-Skyline groups.

Property 1 (Asymmetry). *Give two groups G and G' with same size. If $G \prec_g G'$, then $G' \not\prec_g G$.*

Property 2 (Transitivity). *Given three groups G_1, G_2, and G_3 with same size. If $G_1 \prec_g G_2$ and $G_2 \prec_g G_3$, then $G_1 \prec_g G_3$.*

Lemma 1. *A point in a G-Skyline group cannot be dominated by a point outside the group.*

Proof. By contradiction, assume a point p_i in a G-Skyline group G is dominated by a point p_j outside the group. If we use p_j to replace p_i in G, the new group will g-dominate G since p_j dominates p_i and all the other points are the same, which contradicts the G-Skyline definition. □

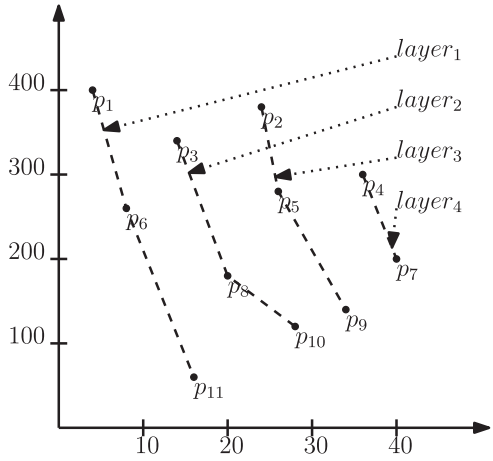


Fig. 2. Skyline layers.

Skyline Layers. Motivated by Lemma 1, we present a structure representing the points and their dominance relationships based on the notion of skyline layers. A formal definition is presented as follows.

Definition 4 (Skyline Layers). *Given a dataset P of n points in a d-dimensional space. The set of skyline layer $layer_1$ contains the skyline points of P, i.e., $layer_1 = skyline(P)$. The set of $layer_2$ contains the skyline points of $P \backslash layer_1$, i.e., $layer_2 = skyline(P \backslash layer_1)$. Generally, the set of $layer_j$ contains the skyline points of $P \backslash \bigcup_{i=1}^{j-1} layer_i$, i.e., $layer_j = skyline(P \backslash \bigcup_{i=1}^{j-1} layer_i)$. The above process is repeated iteratively until $P \backslash \bigcup_{i=1}^{j-1} layer_i = \emptyset$.*

An example of skyline layers of Fig. 1 is shown in Fig. 2. It is easy to see from Definition 4 that for a point p, if there is no point in $layer_{i-1}$ that can dominate p, p should be in $layer_{i-1}$ or a lower layer.

Property 3. *For a point p in $layer_i$, where $2 \leq i \leq l$ and l is the maximum layer number, there must be at least one point in $layer_j$ $(1 \leq j \leq i-1)$ that dominates p.*

We then make an observation that in order to compute s-point G-Skyline groups, we only need to examine the points from the first s skyline layers. We formally present the theorem below.

Theorem 1. *If a group $G = \{p_1, p_2, \ldots, p_s\}$ is a s-point G-Skyline group, then all points in G belong to the first s skyline layers.*

Proof. We prove by contradiction. Assume a point of G is from the jth skyline layer where $j \geq (s+1)$. Without loss of generality, we assume this point is p_s. Since there are s points in G, there is at least one layer $layer_i, 1 \leq i \leq s$, that has no point in G. According to Property 3, p_s should be dominated by at least one point in $layer_i$, denoted by t. Then it is easy to see that group $G' = \{p_1, p_2, .., p_{s-1}, t\}$ by replacing p_s in G with t can dominate group $G = \{p_1, p_2, \ldots, p_{s-1}, p_s\}$. Then group G is not a G-Skyline group which is a contradiction. □

Directed Skyline Graph. We now present a definition of directed skyline graph, a data structure we use to represent the points from the first s skyline layers as well as their dominance relationships, in order to compute s-point G-Skyline groups.

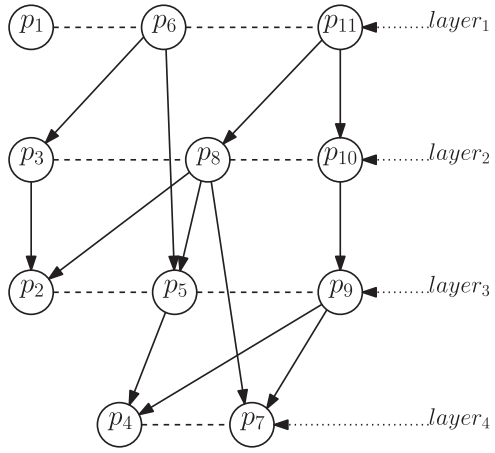


Fig. 3. Directed skyline graph.

Definition 5 (Directed Skyline Graph (DSG)). *A directed skyline graph is a graph where a node represents a point and an edge represents a dominance relationship. Each node has a structure as* $[layer\ index, point\ index, parents, children]$, *where layer index ranging from 1 to* s *indicates the skyline layer that the point lies on, point index ranging from 0 to* $\mathbb{S}_s - 1$ *uniquely identifies the point and* $\mathbb{S}_s$ *is the number of points in the first* s *skyline layers, parents include all the points that dominate this point, and children include all the points that are dominated by the point.*

Example 2. Fig. 3 shows the DSG corresponding to the skyline layers in Fig. 2. Note that $p_6 \prec p_5$ and $p_5 \prec p_4$ imply $p_6 \prec p_4$. For visualization clarity, we omit all indirect dominance edges such as $p_6 \prec p_4$.

Lemma 2. *Given a point* p, *if* p *is in a G-Skyline group,* p*'s parents must be included in this G-Skyline group.*

Proof. By Lemma 1, a point p in a G-Skyline group cannot be dominated by a point outside the group, i.e., all the points that dominate p must be included in this G-Skyline group. □

Verification of G-Skyline. Motivated by Lemma 2, we define a concept of Unit Group and then formally state a theorem for verifying whether a group is a G-Skyline group based on unit group.

Definition 6 (Unit Group). *Given a point* p_i *in DSG,* p_i *and its parents form the unit group* u_i *for* p_i.

Example 3. The unit group of p_5 in Fig. 3, denoted by u_5, contains p_5 and its parents p_6, p_{11}, p_8. Thus, $u_5 = \{p_6, p_{11}, p_8, p_5\}$.

Based on Lemma 2, we have the verification of G-Skyline theorem as follows.

Theorem 2 (Verification of G-Skyline). *Given a group* $G = \{p_1, p_2, \ldots, p_s\}$, *it is a G-Skyline group, if its corresponding unit group set* $S = u_1 \cup u_2 \cup \ldots \cup u_s$ *contains* s *points, i.e.,* $|S|_p = s$.

This theorem is significant because given a group G, in order to check whether it is a G-Skyline group, we do not need to compare G with all other candidate groups any more. Instead, we only need to check whether its corresponding unit group set S has s points.

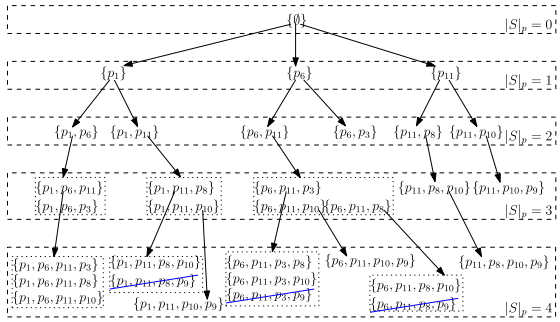


Fig. 4. The point-wise algorithm when $s = 4$.

4 Finding G-Skyline Groups

In this section, we present our algorithms for efficiently finding G-Skyline groups given the DSG built from the first s skyline layers. The first s skyline layers can be constructed in $O(n + \mathbb{S}_s \log s)$ time in two dimensional space and $O(\mathbb{S}^2)$ time in high dimensional space [18]. We first present a point-wise algorithm which builds G-Skyline groups from points (adding one point at a time), then present a unit group-wise algorithm which builds G-Skyline groups from unit groups (adding one unit group at a time). Before beginning the discussion of the two algorithms, we first show a preprocessing step similar to that in [12], [16], [39] to further prune points from the first s skyline layers.

Preprocessing. Theorem 2 shows that a s-point group is a G-Skyline group, if for each point p_i in the group, its parents are also in the group, i.e., the unit group u_i is a subset of the group. Therefore, for a point p_i, if the point size of its unit group is greater than s, i.e., $|u_i|_p > s$, it will not be in any s-point G-Skyline group, and we can remove p_i directly from the DSG without having to consider it. If $|u_i|_p = s$, we can output u_i as one of the G-Skyline groups, and p_i will not be considered either as it will not contribute to any other G-Skyline groups.

Example 4. If we set $s = 4$ (we will use $s = 4$ in all the remaining examples of the paper), the node p_2, p_4, p_7 in Fig. 3 can be removed directly because $|u_2|_p = 5, |u_4|_p = 7$, $|u_7|_p = 5$. Unit group $u_5 = \{p_6, p_{11}, p_8, p_5\}$ can be output as a G-Skyline group. As a result, p_2, p_5, p_4, p_7 will not be considered in our algorithms.

4.1 The Point-Wise Algorithm

The problem of finding G-Skyline groups can be tackled by the classic set enumeration tree search framework. The idea is to expand possible groups over an ordered list of points as illustrated in Fig. 4. Each node in the set enumeration tree is a *candidate* group. The first level contains the root node which is the empty set, while the ith level contains all i-point groups. Naively, we can enumerate all $\binom{\mathbb{S}_s}{s}$ candidates and employ Theorem 2 to check each candidate. However, this baseline method is too time-consuming which will be verified in our experiments.

The main idea of our algorithm is to dynamically generate the set enumeration tree of candidate groups one level at a time while pruning the non-G-Skyline candidates as much as possible without having to check them. For each node, we store a tail set that consists of the points with point index

larger than the points in current node. Each node can be expanded to create a set of new nodes at the next level, each by adding a new point from its tail set. The root node contains an empty set with a tail set composed of all remaining points from the first s skyline layers after the preprocessing. We present our tree expansion and pruning strategies in detail below.

Subtree Pruning. We observe a property of superset monotonicity which allows us to do subtree pruning when a node is not a G-Skyline group.

Theorem 3 (Superset Monotonicity). *If a group G_i with i points is not a G-Skyline group, by adding a new point from its tail set, the new group G_{i+1} with $i+1$ points is not a G-Skyline group either.*

Proof. If G_i is not a G-Skyline group, there is a group G_i' with i points that dominates G_i. For any superset of G_i with a new point p_j added from G_i's tail set, we denote it by $G_i \cup p_j$. Since the points are ordered by skyline layers in our DSG, and any point in G_i's tail set has a larger point index than the points in G_i, hence p_j will not dominate any points in G_i, and will not be in G_i'. It is then easy to see that the group $G_i' \cup p_j$ dominates $G_i \cup p_j$, i.e., $G_i' \cup p_j \prec G_i \cup p_j$. □

This theorem implies that if a candidate group G is not a G-Skyline group, we do not need to expand it further or check its subtree.

Tail Set Pruning. Each node in our tree can be expanded to a set of new nodes by adding a point from its tail set. However, not all tail points need to be considered. Recall Lemma 1, a point in a G-Skyline group cannot be dominated by a point outside the group. In other words, if a point p is to be added to a group G to form a new group that is a G-Skyline group, its parents must be in the group already. This means that p must be either a skyline point (with no parent), or a child of some points in G. Note this is a necessary condition but not sufficient. Once the candidate group is built, we still need to check whether p's parents are all in G (Theorem 2). However, this necessary condition allows us to quickly prune those points from the tail set of G that are not skyline points or not a child of points in G. In addition, given a candidate group G, if all its points are in the first i skyline layers, p must come from the first $i+1$ skyline layers. Otherwise, we can find a point outside G that lies on the $(i+1)^{th}$ layer to dominate p. Hence we can also prune the points beyond the $(i+1)^{th}$ layer. Once a point is pruned from the tail set, it will not be used to expand the current node. This is because based on the superset monotonicity, any node in the subtree of the new node will not be a G-Skyline group either.

Algorithm. Given the above two pruning strategies, we show our complete point-wise algorithm in Algorithm 1. Line 1 initializes the root node and its tail set. Tail set pruning is implemented in Line 4 to Line 10. For each node G_j at level i, it is expanded with a new point p from its tail set to form a new candidate group only if p is a child of some points in the current node or a skyline point and p is in the first $i+1$ layers. The for-loop in Line 4 and Line 6 can be finished in linear time $O(|CS|)$ and $O(|CS|+|TS|)$, respectively, by employing the idea of merge sort, where $|CS|$ and $|TS|$ are the size of children set and tail set. The candidate group is verified in Line 13. If it is not a G-Skyline group, it will be pruned from the tree (subtree pruning).

Algorithm 1. The Point-Wise Algorithm for Computing G-Skyline Groups

input : a DSG and group size s.
output : G-Skyline(s) groups.

```
1  initialize the G-Skyline(0) group at root node as an empty set
   and its tail set as all points from DSG after preprocessing;
2  for i=1 to s do
3    for each G-Skyline(i-1) group G do
4      for each point p_l in G do
5        add p_l's children to Children Set CS;
6      for each point p_j in Tail Set(TS) of G do
7        if p_j is not in CS && p_j is not a skyline point then
8          delete p_j;
9        if p_j.layer − max_l{p_l.layer} ≥ 2 then
10         delete p_j;
11     for each remaining point p in tail set of G do
12       add p to G to form a new candidate G-Skyline(i) group;
13       if the new candidate group is not a G-Skyline group then
14         delete;
```

Example 5. We show a running example of Algorithm 1 in Fig. 4 based on Fig. 3. The root node at level $|S|_p = 0$ has an initial tail set $\{p_1, p_6, p_{11}, p_3, p_8, p_{10}, p_9\}$. Points p_3, p_8, p_{10}, p_9 can be pruned immediately because they are not skyline points, i.e., their parents are not in the root node. The remaining points p_1, p_6, p_{11} are used to create the new nodes at level $|S|_p = 1$. Similarly, for node $\{p_1, p_{11}\}$ at level $|S|_p = 2$, its tail set is $\{p_3, p_8, p_{10}, p_9\}$. Point p_3 can be pruned because p_3 is not a child of either p_1 or p_{11}. Point p_9 also can be pruned as $p_9.layer - max\{p_1.layer, p_{11}.layer\} = 2$. Hence, the remaining points p_8, p_{10} are used to create the new candidate groups at level $|S|_p = 3$, namely $\{p_1, p_{11}, p_8\}$ and $\{p_1, p_{11}, p_{10}\}$. As a result, level $|S|_p = 4$ shows all candidate 4-point groups that need to be checked. After checking, the ones with blue slash are not G-Skyline groups while the remaining ones are G-Skyline groups. Based on our pruning strategies, only 31 candidate groups are generated and checked while the baseline approach needs to enumerate and check 70 candidate groups.

4.2 The Unit Group-Wise Algorithm

The point-wise algorithm expands candidate groups one point at a time. We already showed (in Lemma 1) that a point in a G-Skyline group cannot be dominated by a point outside the group. In other words, for a point in a G-Skyline group, its unit group must be in the group. This motivates our unit group-wise algorithm which expands candidate groups by unit groups, adding one unit group at a time. Similar to the point-wise algorithm, we can represent the entire search space of candidate groups as a set enumeration tree, where each node is a set of unit groups. We can dynamically generate the tree while pruning as much non-G-Skyline groups as possible. We can use similar pruning strategies as in the point-wise algorithm.

Superset Pruning. Given a candidate group G with at least s points, the candidate groups in G's subtree will have more than s points. Hence, we can prune G's subtree directly.

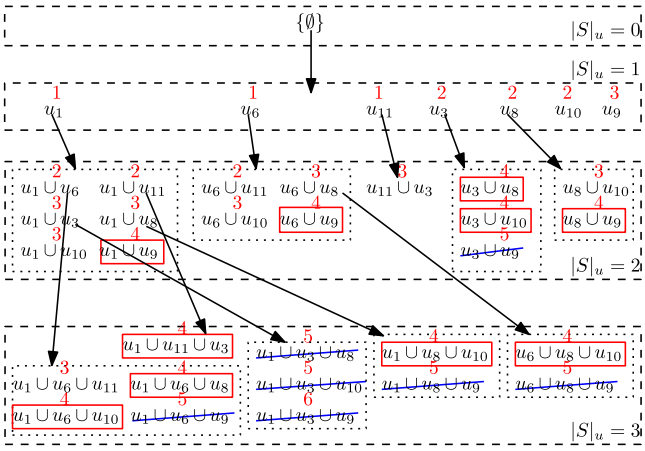


Fig. 5. The basic unit group-wise algorithm when $s = 4$.

Example 6. Fig. 5 shows the dynamically generated set enumeration tree for the unit group-wise algorithm. At level $|S|_u = 2$, $|u_3 \cup u_8|_p = 4$, hence we do not need to check the candidate groups in its subtree, e.g., $u_3 \cup u_8 \cup u_{10}$. This is because $|u_3 \cup u_8 \cup u_{10}|_p = |\{p_6, p_{11}, p_3, p_8, p_{10}\}|_p = 5$.

Tail Set Pruning. For a candidate group G at Level $|S|_u = i$, we do not need to add the children of the unit groups in G to form a new candidate group at Level $|S|_u = i + 1$. Therefore, the children of the unit groups in G can be pruned from the tail set.

Example 7. We show a running example for the unit group-wise algorithm with the two pruning strategies in Fig. 5. The number on each candidate group represents the number of points in this candidate group. All candidate groups with one unit group are checked at Level $|S|_u = 1$. For candidate group u_6 at Level $|S|_u = 1$, its tail set is $\{u_{11}, u_3, u_8, u_{10}, u_9\}$. However, u_3 can be pruned by Tail Set Pruning because u_3 is the child of u_6. From Level $|S|_u = 2$ to Level $|S|_u = 3$, candidate group $u_3 \cup u_8$'s subtree does not need to be checked since $|u_3 \cup u_8|_p = 4$, thanks to Superset Pruning. Based on the two pruning strategies, only 35 candidate groups need to be checked as shown. The resulting G-Skyline(4) groups are shown in red (solid) boxes and we can see that they come from different levels while the point-wise algorithm outputs all G-Skyline(4) groups at Level $|S|_p = 4$.

In addition to the above strategies, we show a few additional refinements that further improve the algorithm.

Unit Group Reordering. We observe that Superset Pruning is important to reduce the candidate groups. This motivates us to reorder the unit groups in order to increase the effectiveness of Superset Pruning. Recall that Superset Pruning can be applied when a candidate group G has $|G|_p \geq s$. Therefore it would be beneficial if we build large candidate groups first which allows us to prune more non-G-Skyline candidate groups at an early stage. A good heuristic for accomplishing this is to reorder the unit groups for each point in the reverse order of their point index. This is because a point with a larger index, i.e., at a higher skyline layer, tends to have more parents, and hence a larger unit group size. By ordering the unit groups this way, they are likely in the (although not monotonically) decreasing order in their unit group size. We simply need to modify the Tail Set Pruning strategy such that for each node G, the parents (instead of the children) of the unit groups of G are pruned from its tail set.

Subset Pruning. Given a candidate group G_i, if $|G_i|_p \leq s$, then any candidate groups that are G_i's subset have at most s points. This motivates us to employ Subset Pruning. For each candidate group G_i with $|G_i|_p < s$ at Level $|S|_u = 1$, we can check a new candidate group by adding the entire tail set of G_i to G_i, i.e., the last or deepest leaf candidate group G_i^{last} in G_i's subtree. If $|G_i^{last}|_p \leq s$, the entire subtree of G_i can be pruned directly and G_i^{last} can be output if $|G_i^{last}|_p = s$. Furthermore, we do not need to check those candidate groups in the subtree of G_i's right siblings G_{sib} because the last candidate group in G_{sib}'s subtree, G_{sib}^{last}, is a subset of G_i^{last}, hence, $|G_{sib}^{last}|_p < s$ too. While both depth-first and breadth-first expansion of the tree will work equally well for Superset Pruning, Subset Pruning benefits from a depth-first expansion such that more siblings can be pruned.

Algorithm 2. The Unit Group-Wise Algorithm for Computing G-Skyline Groups

input : a DSG and group size s.
output : G-Skyline(s) groups.

```
1  build 1-unit group as candidate groups following reverse
   order of point index;
2  for each candidate group G in 1-unit groups do
3    if |G^last|_p = s then
4      output G^last;
5      break;
6    else if |G^last|_p < s then
7      break;
8    i = 2;
9    while the set of candidate group G', such that |G'|_u = i − 1, is
     not empty do
10     for each G' do
11       for each unit group u_i in G' do
12         add u_i's parents to Parents Set PS;
13       for each unit group u_j in tail set of G' do
14         delete u_j if u_j is in PS;
15       for each remaining unit group u in tail set of G' do
16         add u to G' to form a new candidate group G'' that
           |G''|_u = i;
17     delete G';
18     output G'' if |G''|_p = s;
19     delete G'' if |G''|_p ≥s;
20     i++;
```

Algorithm. Given the Superset Pruning, Unit Group Reordering, Subset Pruning, and Tail Set Pruning strategies, the complete unit group-wise algorithm is shown in Algorithm 2. Line 1 applies Unit Group Reordering. Subset Pruning is applied to the 1-unit groups in Lines 3 and 6. We also note that while Subset Pruning can be employed at every node, it also adds additional cost for checking the leaf node. So we only apply them at Level $|S|_u = 1$ to gain the most pruning benefit, as s is far less than n in the usual case. Tail Set Pruning is applied in Line 11 to Line 14 to prune those candidate groups that do not need to be checked. Line 19 applies Superset Pruning to prune those candidate groups with more than s points.

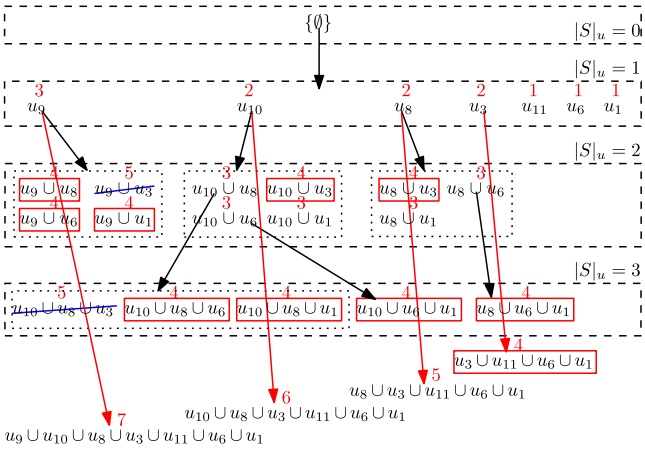


Fig. 6. The enhanced unit group-wise algorithm when $s = 4$.

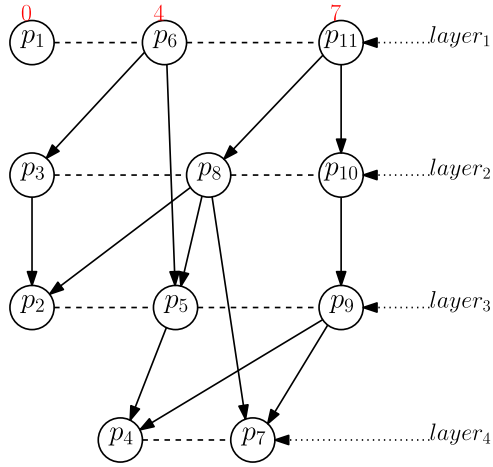


Fig. 7. Top-k G-Skyline groups.

Example 8. We show a running example of Algorithm 2 in Fig. 6. The 1-unit groups are built in reverse order of point index at Level $|S|_u = 1$. For candidate group u_9, we first check the last candidate group (linked by red (thin) arrow) which has $|u_9 \cup u_{10} \cup u_8 \cup u_3 \cup u_{11} \cup u_6 \cup u_1|_p = |u_9 \cup u_8 \cup u_3 \cup u_1|_p = 7 > 4$. So Subset Pruning cannot be applied, and we still need to check u_9's subtree. We build each branch with a depth-first strategy. At candidate group u_3, the last candidate group of its subtree has $|u_3 \cup u_{11} \cup u_6 \cup u_1|_p = |u_3 \cup u_{11} \cup u_1|_p = 4$, so it is a G-Skyline(4) group. Based on the Subset Pruning strategy, the algorithm is terminated because there are no more candidate groups that need to be checked. As a result, only 27 candidate groups need to be checked as shown in Fig. 6.

5 Top-k G-Skyline Groups

One potential drawback of the G-Skyline groups is that the number of groups satisfying the definition can be significantly large, especially for anti-correlated datasets; and there is no priority among these groups, which limits the usage of G-Skyline in practice. In this section, we show how to select top-k representative G-Skyline groups based on the number of dominated points and the number of dominated groups.

5.1 Dominated Points

In this subsection, we define top-k representative G-Skyline groups as those dominate the most number of points. Intuitively, if a G-Skyline group dominates more number of points, the group is more representative.

Definition 7 (Children Set). *Given a point p_i in DSG, p_i and its children form the children set C_i for p_i.*

Definition 8 (Number of Dominated Points (DP)). *Given a G-Skyline group $G = \{p_1, \ldots, p_s\}$ of size s, we say a point p is dominated by group G if p is dominated by one or more points in group G. DP is the number of points dominated by G and $DP(G) = |C_1 \cup \ldots \cup C_s|_p - s$.*

Definition 9 (Top-K G-Skyline Groups based on Dominated Points ($topkG^p$)). *Given a dataset P of n points in d-dimensional space and a parameter k, $topkG^p$ returns top k G-Skyline groups of the dataset $\{G_i | 1 \leq i \leq k\}$ of group size s with highest $DP(G_i)$.*

Example 9. Assume $s = 2$ and $k = 2$ in Fig. 7. We have 6 G-Skyline groups $G_1 = \{p_1, p_6\}, G_2 = \{p_1, p_{11}\}, G_3 = \{p_6, p_{11}\}$, $G_4 = \{p_6, p_3\}$, $G_5 = \{p_{11}, p_8\}$, and $G_6 = \{p_{11}, p_{10}\}$. For $G_1 = \{p_1, p_6\}$, the children set for p_1 is $C_1 = \{p_1\}$, the children set for p_6 is $C_6 = \{p_6, p_3, p_2, p_5, p_4\}$. We have $DP(G_1) = |C_1 \cup C_6| - 2 = 4$. Similarly, we have $DP(G_2) = 7, DP(G_3) = 8, DP(G_4) = 3, DP(G_5) = 6, DP(G_6) = 6$. Therefore, $topkG^p$ returns $G_2 = \{p_1, p_{11}\}$ and $G_3 = \{p_6, p_{11}\}$.

The baseline algorithm to compute $topkG^p$ is straightforward. We first employ the G-Skyline algorithms to compute all the G-Skyline groups $G_i, 1 \leq i \leq m$. For each G_i, we compute $DP(G_i)$, the number of points dominated by G_i, based on the union of its children sets. Finally, we return the k G-Skyline groups with highest $DP(G_i)$. However, this algorithm incurs significant computation involving set unions. In the following, we propose a more efficient algorithm with pruning.

Pruning and Growing Algorithm. The basic idea is to find a candidate set of top k G-Skyline groups and prune the remaining G-Skyline groups as much as possible without having to compute their number of dominated points (DP). We present an upper bound for DP and a subset property for DP as below which will be used for the pruning.

Definition 10 (Upper Bound of Number of Dominated Points (UDP)). *Given a G-Skyline group $G = \{p_1, \ldots, p_s\}$, the upper bound of $DP(G)$ is $UDP(G) = DP(p_1) + \ldots, +DP(p_s)$, where $DP(p_i)$ is the number of points dominated by point p_i.*

Example 10. In Fig. 7, since $C_1 = \{p_1\}$, $C_6 = \{p_6, p_3, p_2, p_5, p_4\}$, and $C_{11} = \{p_{11}, p_8, p_{10}, p_2, p_5, p_9, p_4, p_7\}$, we have $UDP(G_1) = DP(p_1) + DP(p_6) = 0 + 4 = 4$ and $DP(G_1) = |\{p_1, p_6, p_3, p_2, p_5, p_4\}|_p - 2 = 4$. Similarly, we have $UDP(G_4) = 4 + 7 = 11$ and $DP(G_4) = 8$.

It is easy to see that computing UDP is much easier than computing DP. We can use the upper bound $UDP(G_i)$ to prune G_i if it is already lower than those of current top k candidates without computing $DP(G_i)$.

Property 4. *Given a G-Skyline group $G = \{p_1, \ldots, p_s\}$, a G-Skyline group G' is a child G-Skyline group of G if it is a subset of $C_1 \cup \ldots \cup C_s$. And we have $DP(G') \leq DP(G)$.*

Proof. The proof is straightforward. Since G' only consists of the points in G or their children, we have $DP(G') \leq DP(G)$. □

Given the upper bound UDP and the child G-Skyline group property, the basic idea of our algorithm is to first find the top-k representative groups in the first skyline layer, denoted as $topkG^p_{layer_1}$, then prune the remaining groups via the UDP and the child G-Skyline group property. Based on Property 4, we can guarantee that the G-Skyline group in $topkG^p$ is either in $topkG^p_{layer_1}$ or in their child G-Skyline groups. Therefore, we only check the child G-Skyline groups of $topkG^p_{layer_1}$ and prune them if their UDP is lower than the current candidate top k set.

Algorithm 3. The Algorithm for Computing $topkG^p$

input : DSG, group size s, and parameter k.
output : $topkG^p$.
1 **for** *each point* p_i *in* $layer_1$ **do**
2 compute $DP(p_i)$;
3 sort the points in $layer_1$ by $DP(p_i)$ in decreasing order;
4 compute the DP of the top k G-Skyline groups with highest UDP;
5 sort the *candidate groups* by DP in decreasing order;
6 denote the DP of the group with the kth highest DP as $temp$;
7 **for** *each G-Skyline group* G_i *whose points are all from* $layer_1$ **do**
8 **if** $UDP(G_i) > temp$ **then**
9 compute $DP(G_i)$;
10 **if** $DP(G_i) > temp$ **then**
11 insert group G_i into the candidate groups by $DP(G_i)$;
12 update the DP of the group with the kth highest DP in the new candidate groups as $temp$;
13 we have $G_1, G_2, \ldots, G_k$ in candidate groups sorted by DP in decreasing order;
14 set the flag of each candidate group as 0;
15 take the candidate group with highest DP as G_{first};
16 **while** $DP(G_{first}) > temp + 1$ **do**
17 set the flag of G_{first} to 1;
18 enumerate the child G-Skyline groups of G_{first};
19 **for** *each child G-Skyline group* G_j **do**
20 **if** $UDP(G_j) > temp$ **then**
21 compute $DP(G_j)$;
22 **if** $DP(G_j) > temp$ **then**
23 insert child G-Skyline group G_j into the candidate groups by DP;
24 update the DP of the group with the kth highest DP in the new candidate groups as $temp$;
25 take the candidate group with highest DP and flag=0 as G_{first};

The detailed algorithm is shown in Algorithm 3. We search $topkG^p$ from the G-Skyline groups in the first layer in Lines 1-12 and their child G-Skyline groups in Lines 16-25. In order to find $topkG^p_{layer_1}$, we compute the number of points dominated by each p_i, $DP(p_i)$, in Lines 1-2. Then we compose the G-Skyline groups by enumerating all the possible subsets of the points in the first layer in decreasing order of $DP(p_i)$. The reason is that the group consisting of points with higher $DP(p_i)$ is more likely to have a higher $DP(G_i)$. In order to find the top k groups with highest DP, we first compute the UDP in Line 8 as a pruning criteria (since it is much faster to compute than DP) before computing DP in Line 10. Once we find the top k groups in the first layer, we use it as the candidate top k set. Then we sort the groups in decreasing order of DP and refer to the first one as G_{first} and the DP of the kth one as the $temp$. In Line 17, we set the flag of G_{first} to 1, which means the child G-Skyline groups of G_{first} are considered. We then check their child G-Skyline groups following that order and update the top k set accordingly. The reason for the order is that the child G-Skyline groups of a group with higher DP is more likely to have a higher DP which will result in more effective pruning for the remaining groups. Again, we use UDP as the pruning criteria before computing DP when updating the top k set. The algorithm terminates when $DP(G_{first}) \leq temp + 1$, because the child G-Skyline groups of G_{first} cannot have larger DP than G_{first}.

In the above algorithm, we assumed there are at least k G-Skyline groups in the first layer, i.e., $\binom{|layer_1|_p}{s} \geq k$ which happens in most cases in practice. For the extreme case of $\binom{|layer_1|_p}{s} < k$, we only need to modify the algorithm by replacing the first layer with the first v layers in Algorithm 3, where $\binom{|layer_1+\ldots+layer_v|_p}{s} \geq k$ and $\binom{|layer_1+\ldots+layer_{v-1}|_p}{s} < k$.

Example 11. Assume $s = 2$ and $k = 2$ in Fig. 7. We first compute $DP(p_1) = 0$, $DP(p_6) = 4$, and $DP(p_{11} = 7)$. Then we sort the points as p_{11}, p_6, p_1 based on the decreasing order of DP. We compute $DP(\{p_{11}, p_6\}) = 8$ and $DP(\{p_{11}, p_1\}) = 7$ for the first two skyline groups. We have $temp = 7$ for the current kth G-Skyline group. We then check the remaining groups in the first layer. Since $UDP(\{p_6, p_1\})$ is smaller than $temp$, the for-loop ends. In Line 15, we take the group $\{p_{11}, p_6\}$ as G_{first}, since $DP(G_{first}) = 8$ is not larger than $temp + 1 = 8$, the algorithm ends.

Approximate Algorithm. The accurate algorithm is prohibitive on a larger dataset with high dimensions due to the curse of dimensionality. For example, there are 1620 points in the first layer in a stock dataset of kaggle.com with 30000 tuples and 6 dimensions. Although our pruning techniques are very efficient, we have to check $\binom{1620}{5} = 9.2 \times 10^{13}$ groups ($s = 5$) for the dominance relationship which is prohibitively large and impractical. We can implement an approximate algorithm by only using the top points with highest dominated number of points from the first layer as the candidate points. The stock dataset can be finished in 10 minutes if we use 100 points. This approximation is reasonable because: 1) for small k, e.g., 1000, the approximation has the same accurate result with the accurate algorithm, and 2) even when k is large and there might be some approximation in the result, it is still acceptable since our goal is to choose skyline groups to best represent the dataset.

5.2 Dominated Groups

In this subsection, we define top-k representative G-Skyline groups as those dominate the most number of groups. Intuitively, if a G-Skyline group dominates more number of groups, the group is more representative.

Definition 11 (Top-K G-Skyline groups based on dominated Groups ($topkG^g$)). *Given a dataset P of n points in d-dimensional space and a parameter k, we have m G-Skyline*

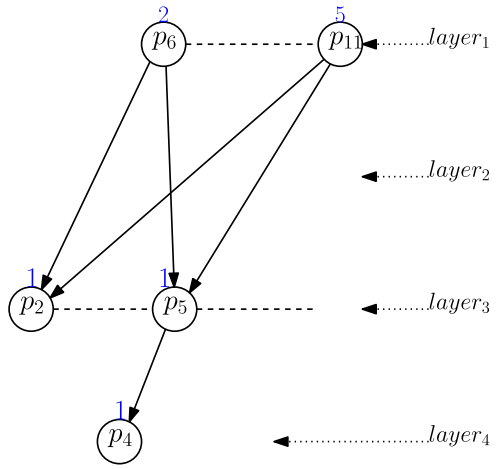


Fig. 8. Representative graph.

groups $G_i, 1 \le i \le m$. Each G-Skyline group G_i g-dominates $DG(G_i)$ groups, where $DG(G_i)$ is the number of groups with size s g-dominated by G_i. $topkG^g$ returns those k G-Skyline groups of size s with highest $DG(G_i)$.

Example 12. Assume $s = 2$ and $k = 2$ in Fig. 7. We have 6 G-Skyline groups $G_1 = \{p_1, p_6\}$, $G_2 = \{p_1, p_{11}\}$, $G_3 = \{p_6, p_{11}\}$, $G_4 = \{p_6, p_3\}$, $G_5 = \{p_{11}, p_8\}$, and $G_6 = \{p_{11}, p_{10}\}$. For $G_1 = \{p_1, p_6\}$, it g-dominates groups $\{p_1, p_3\}$, $\{p_1, p_2\}$, $\{p_1, p_5\}$, and $\{p_1, p_4\}$, thus, $DG(G_1) = 4$. Similarly, we have $DG(G_2) = 7$, $DG(G_3) = 33$, $DG(G_4) = 6$, $DG(G_5) = 24$, $DG(G_6) = 21$. Therefore, $topkG^g$ returns $G_3 = \{p_6, p_{11}\}$ and $G_5 = \{p_{11}, p_8\}$.

The baseline algorithm to compute $topkG^g$ is straightforward. We first employ the G-Skyline algorithms to compute all the G-Skyline groups $G_i, 1 \le i \le m$. Then for each G_i, we compute and count all groups that are g-dominated by G_i as $DG(G_i)$ and then return those k G-Skyline groups with highest $DG(G_i)$. This algorithm clearly incurs significant computations. We propose below more efficient algorithms for computing $DG(G_i)$ and a pruning algorithm utilizing the upper bound for $DG(G_i)$.

Pruning Algorithm. The naive way to compute DG of $G = \{p_1, \ldots, p_s\}$ is to enumerate all the child groups of G and verify if each child group is g-dominated by G. We say a group $G' = \{p'_1, \ldots, p'_s\}$ is a child group of G if p'_i is a point from p_i's children set C_i. For example, given $G = \{p_6, p_{11}\}$, we need to enumerate all its child groups which include $\{p_6, p_3\}$, $\{p_6, p_8\}$, and so on. $\{p_6, p_3\}$ is not g-dominated by G while $\{p_6, p_8\}$ is. However, it is too time-consuming to enumerate all the child groups and verify them. We observe that the points only can be dominated by one point in G have the same contribution for DG(G). For example, p_8 and p_{10} have the same contribution when computing $DG(\{p_6, p_{11}\})$. Therefore, we combine those points in a representative graph which is defined as follows.

Given a directed skyline graph of the dataset and a G-Skyline group $G = \{p_1, \ldots, p_s\}$, for each skyline point p_i whose child is not in G, we recursively merge its child that is not in G and has p_i as the only parent with p_i in the skyline graph and update its weight as the number of merged points in this new node. The unmerged nodes have a default weight as 1. We refer to the resulting graph including the points cannot be merged in G, the merged nodes, and their children as the *representative skyline graph* for G.

Example 13. Fig. 8 illustrates the representative graph for $\{p_6, p_{11}\}$. We merge p_3 with p_6 as p_6 and its weight is 2. Similarly, we merge p_8, p_{10}, p_9, p_7 with p_{11} as p_{11} and its weight is 5. All other nodes p_2, p_5, p_4 have a default weight as 1.

With the representative skyline graph of G, we can easily compute $DG(G)$. Taking $DG(\{p_6, p_{11}\})$ as an example, in the baseline algorithm, we have two corresponding children sets $C_1 = \{p_6, p_3, p_2, p_5, p_4\}$ and $C_2 = \{p_{11}, p_8, p_{10}, p_2, p_5, p_9, p_4, p_7\}$. Therefore, we need to enumerate $5 \times 8 = 40$ child groups and verify if each child group is dominated by $\{p_6, p_{11}\}$. However, based on the representative skyline graph, we have two compact children sets $C_1 = \{p_6, p_2, p_5, p_4\}$ and $C_2 = \{p_{11}, p_2, p_5, p_4\}$. Therefore, we only need to enumerate $4 \times 4 = 16$ child groups and then multiply the weights of nodes in the representative skyline graph.

While the above algorithms for computing DG is more efficient than the baseline, our goal is still to prune the G-Skyline groups as much as possible without having to compute their DG. We present an upper bound for DG as below which will be used for our pruning.

Definition 12 (Upper Bound for Number of Dominated Groups (UDG)). *Given a G-Skyline group $G = \{p_1, \ldots, p_s\}$, the upper bound of DG(G) is $UDG(G) = (DP(p_1) + 1)\times, \ldots, \times(DP(p_s) + 1) - 1$.*

Example 14. In Fig. 7, we have $UDG(G_3) = (DP(p_6) + 1)\times (DP(p_{11}) + 1) - 1 = 39$.

Algorithm 4. The Pruning Algorithm for Computing $topkG^g$

input : DSG, G-Skyline groups, and parameter k.
output : $topkG^p$.
1 **for** *each point p_i with $u_i \le s$* **do**
2 compute $DP(p_i)$;
3 sort the points with $u_i \le s$ by $DP(p_i)$ in decreasing order;
4 compute the DG of the top-k G-Skyline groups with highest UDG;
5 sort the *candidate groups* by DG in decreasing order;
6 denote the DG of the group with the kth highest DG as $temp$;
7 **for** *each G-Skyline group G_i* **do**
8 **if** $UDG(G_i) > temp$ **then**
9 compute $DG(G_i)$;
10 **if** $DG(G_i) > temp$ **then**
11 insert group G_i into the candidate groups by $DG(G_i)$;
12 update the DG of the group with the kth highest DG in the new candidate groups as $temp$;

It is easy to see that computing UDG is much easier than computing DG. We can use the upper bound $UDG(G_i)$ to prune G_i if it is already lower than those of current top k candidates without computing $DG(G_i)$. We propose our algorithm for computing $topkG^g$ in Algorithm 4. Similar to the algorithm for computing $topkG^p$, we compute the DG of the top k G-Skyline groups with highest UDG in Line 4. Because Property 4 is not established for dominated groups, we need to check all the G-Skyline groups in Line 7. In order to find the top k groups with highest DG, we first compute

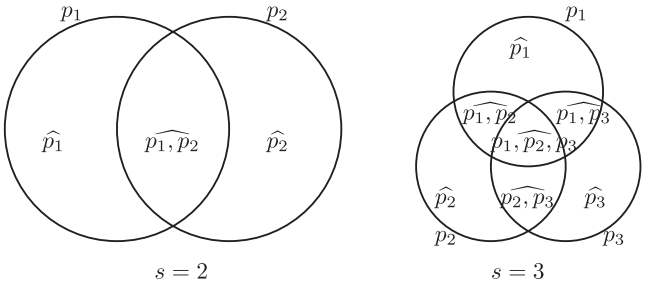


Fig. 9. The special case.

the UDG in Line 8 as a pruning criteria before computing DG in Line 10.

Example 15. Assume $s=2$ and $k=2$ in Fig. 7. We first compute $DP(p_1)=0$, $DP(p_6)=4$, $DP(p_{11})=7$, $DP(p_3)=1$, $DP(p_8)=4$, and $DP(p_{10})=3$. Then we sort the points as $p_{11},p_6,p_8,p_{10},p_3,p_1$ based on the decreasing order of DP. We compute $DG(\{p_{11},p_6\})=33$ and $DG(\{p_{11},p_8\})=24$ for the first two skyline groups. We have $temp=24$ for the current kth G-Skyline group. We then check the remaining G-Skyline groups. Since $UDG(\{p_{11},p_1\})=7$, $UDG(\{p_1,p_6\})=4$, and $UDG(\{p_3,p_6\})=9$ are smaller than $temp$, we only need to compute $DG(\{p_{11},p_{10}\})=21$ which is smaller than $temp=24$. Finally, $topkG^g$ returns $\{p_{11},p_6\}$ and $\{p_{11},p_8\}$.

Counting Algorithm. In the above method, we need to enumerate all the child groups of G, which is time-consuming. Our observation is that the number of dominated groups is determined by the number of points dominated by each point and each subset (multi-points) of G. Given a G-Skyline group $G=\{p_1,\ldots,p_s\}$, we can compute the number of groups generated by $C_1\cup\ldots\cup C_s$, i.e., $\binom{|C_1\cup\ldots\cup C_s|}{s}$. However, there are some groups that cannot be dominated by G, e.g., two points in a group G' can be dominated by only one point in G. Therefore, what we need to do is to eliminate the groups that cannot be dominated by G from $\binom{|C_1\cup\ldots\cup C_s|}{s}$ groups.

Given a G-Skyline group $G=\{p_1,\ldots,p_s\}$, for any subset G_{sub} of G, we have $G_{sub}=\{p_1,\ldots,p_{s'}\}$, where $s'\le s$. We use $p_1,\widehat{\ldots},p_{s'}$ to denote the points that can be dominated by $\{p_1,\ldots,p_{s'}\}$ simultaneously but cannot be dominated by any other point outside of $\{p_1,\ldots,p_{s'}\}$ in G and use $|p_1,\widehat{\ldots},p_{s'}|$ to denote the size of $p_1,\widehat{\ldots},p_{s'}$. For example, in Fig. 7, $|\widehat{p_6}|=1$ because p_3 is exclusively dominated by p_6 but p_2,p_5,p_4 are also dominated by p_{11}. For another example, $|\widehat{p_6,p_{11}}|=3$ because p_2,p_5,p_4 are exclusively dominated by p_6 and p_{11}. Point p_3 is not counted because it is only dominated by p_6 but not p_{11}.

The Venn diagrams of $s=2,3$ are shown in Fig. 9. Based on the Venn diagrams, we have $DG(\{p_1,p_2\})$ as follows.

$$DG(\{p_1,p_2\})=\binom{|\widehat{p_1}|+1+|\widehat{p_2}|+1+|\widehat{p_1,p_2}|}{2}-\binom{|\widehat{p_1}|+1}{2}-\binom{|\widehat{p_2}|+1}{2}-1.$$

We can choose any two points from $C_1\cup C_2$, i.e., we have $\binom{\widehat{p_1}+1+\widehat{p_2}+1+\widehat{p_1,p_2}}{2}$ groups. However, if we choose two points from the same set $\widehat{p_i}$, the new group is not dominated by $\{p_1,p_2\}$. We have $\binom{\widehat{p_1}+1}{2}+\binom{\widehat{p_2}+1}{2}$ such types of groups, which should be eliminated. Finally, the group $\{p_1,p_2\}$ cannot be dominated by itself, thus, we need to minus this group. Similarly, for $s=3$, we can choose any three points from $C_1\cup C_2\cup C_3$ and eliminate four types of groups. We show four types of groups as follows: (1) choose two points from the same set $\widehat{p_i}$, (2) choose three points from the same set $\widehat{p_i}$, (3) choose three points from the same set $\widehat{p_i,p_j}$, and (4) choose two points from the same set $\widehat{p_i,p_j}$ and choose one point from $\widehat{p_i}$ or $\widehat{p_j}$. Therefore, we have $DG(\{p_1,p_2,p_3\})$ as follows.

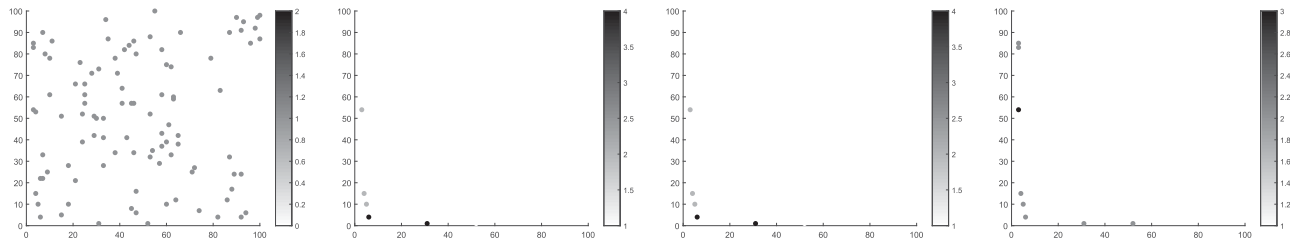
$$\begin{aligned}
&DG(\{p_1,p_2,p_3\})=\\
&\binom{|\widehat{p_1}|+1+|\widehat{p_2}|+1+|\widehat{p_3}|+1+\sum_{1\le i<j\le 3}|\widehat{p_i,p_j}|+|p_1,\widehat{p_2,p_3}|}{3}\\
&-\binom{|\widehat{p_1}|+1}{2}\times\binom{|\widehat{p_2}|+1+|\widehat{p_3}|+1+\sum_{1\le i<j\le 3}|\widehat{p_i,p_j}|+|p_1,\widehat{p_2,p_3}|}{1}\\
&-\binom{|\widehat{p_2}|+1}{2}\times\binom{|\widehat{p_1}|+1+|\widehat{p_3}|+1+\sum_{1\le i<j\le 3}|\widehat{p_i,p_j}|+|p_1,\widehat{p_2,p_3}|}{1}\\
&-\binom{|\widehat{p_3}|+1}{2}\times\binom{|\widehat{p_1}|+1+|\widehat{p_2}|+1+\sum_{1\le i<j\le 3}|\widehat{p_i,p_j}|+|p_1,\widehat{p_2,p_3}|}{1}\\
&-\binom{|\widehat{p_1}|+1}{3}-\binom{|\widehat{p_2}|+1}{3}-\binom{|\widehat{p_3}|+1}{3}-\binom{|\widehat{p_1,p_2}|}{3}-\binom{|\widehat{p_1,p_3}|}{3}-\binom{|\widehat{p_2,p_3}|}{3}\\
&-\binom{|\widehat{p_1,p_2}|}{2}\times(|\widehat{p_1}|+1+|\widehat{p_2}|+1)-\binom{|\widehat{p_1,p_3}|}{2}\times(|\widehat{p_2}|+1+|\widehat{p_3}|+1)\\
&-\binom{|\widehat{p_2,p_3}|}{2}\times(|\widehat{p_1}|+1+|\widehat{p_3}|+1)-1.
\end{aligned}$$

It is hard to draw the Venn diagram for a large s, but we can still present the algebraic expression to compute the number of groups dominated by group $G=\{p_1,\ldots,p_s\}$. For group size s, we have $\binom{s}{1}+\binom{s}{2}+\cdots+\binom{s}{s}=2^s-1$ different types of points in $C_1\cup\ldots\cup C_s$. For each point p'_i in dominating candidate group $G'=\{p'_1,\ldots,p'_s\}$, p'_i can be chosen from 2^s-1 types. Therefore, we have $(2^s-1)^s$ different types of dominating candidate groups formed by s points from $C_1\cup\ldots\cup C_s$. We then prune those duplicate types. However, there are still some types that cannot be dominated by group G, e.g., the types that have all the s points chosen from $\widehat{p_1}$. Therefore, we need to eliminate those types by bipartite matching. The detailed algorithm is shown in Algorithm 5.

Algorithm 5. The Counting Algorithm for Computing $topkG^g$

input : group $G=\{p_1,\ldots,p_s\}$.
output : number of groups dominated by G.
1 **for** *each point in* $C_1\cup\ldots\cup C_s$ **do**
2 determine its type from 2^s-1 different types;
3 count=0;
4 we have $(2^s-1)^s$ different types of dominating candidate groups formed by s points;
5 prune those duplicate types;
6 **for** *each remaining type* **do**
7 **if** *there is a bipartite matching between this type and* $\{p_1,\ldots,p_s\}$ **then**
8 count=count+number of groups satisfying this type;
9 **return** count;

Taking group $\{p_6,p_{11}\}$ as an example, we have three different types of points: $\widehat{p_6}$, $\widehat{p_{11}}$, and $p_6,\widehat{p_{11}}$. For those three



(a) Dataset (b) $topkG^p$ (c) $topkG^g$ (d) contour skyline

Fig. 10. Comparing $topkG^p$ and $topkG^g$ with contour skyline.

different types of points, we have 3^2 types of groups: type 1), one point from $\widehat{p_6}$ and another point from $\widehat{p_6}$; type 2), one point from $\widehat{p_6}$ and another point from $\widehat{p_{11}}$; type 3), one point from $\widehat{p_6}$ and another point from $p_6,\widehat{p_{11}}$; type 4), one point from $\widehat{p_{11}}$ and another point from $\widehat{p_6}$; type 5), one point from $\widehat{p_{11}}$ and another point from $\widehat{p_{11}}$; type 6), one point from $\widehat{p_{11}}$ and another point from $p_6,\widehat{p_{11}}$; type 7), one point from $p_6,\widehat{p_{11}}$ and another point from $\widehat{p_6}$; type 8), one point from $p_6,\widehat{p_{11}}$ and another point from $\widehat{p_{11}}$; and type 9), one point from $p_6,\widehat{p_{11}}$ and another point from $p_6,\widehat{p_{11}}$. We prune those duplicate types: types 4), 7), 8). We then use bipartite matching to eliminate those types without bipartite matching, i.e., types 1), 5). Finally, we have four types that can be dominated by group G, i.e., types 2), 3), 6), 9). Therefore, we have

$$DG(p_6,p_{11}) = \binom{|\widehat{p_6}|+1}{1}\binom{|\widehat{p_{11}}|+1}{1}+ \binom{|\widehat{p_6}|+1}{1}\binom{|p_6,\widehat{p_{11}}|}{1}+\binom{|\widehat{p_{11}}|+1}{1}\binom{|p_6,\widehat{p_{11}}|}{1}+\binom{|p_6,\widehat{p_{11}}|}{2}-1$$

Because $|\widehat{p_6}|=1$, $|\widehat{p_{11}}|=4$, and $|p_6,\widehat{p_{11}}|=3$, we have $DG(p_6,p_{11}) = 2\times 5+2\times 3+5\times 3+3-1=33$.

5.3 Comparing with Contour Skyline

In this section, we compared the results of $topkG^p$ and $topkG^g$ with contour skyline [?] in Figs. 10a, 10b, 10c, and 10d. We show the original dataset with 100 points in Fig. 10a. In Figs. 10b, 10c, and 10d, we show the top-3 representative G-Skyline groups with group size 3 based on the number of dominated points, the number of dominated groups, and contour, respectively. We count the number of occurrence for each point. The more count number for each point, the deeper the color in the figures. We can see that the results of $topkG^p$ and $topkG^g$ are very similar but both of them are different from contour skyline. Contour skyline contains more points than both $topkG^p$ and $topkG^g$ because its main propose is to capture the "contour" of the G-Skyline groups. In contrast, our proposed definitions $topkG^p$ and $topkG^g$ capture the "representative" G-Skyline groups that can dominate the most number of points or groups.

6 Experiments

In this section, we present experimental studies evaluating our approach.

6.1 Experiment Setup

We perform an extensive empirical study to examine the point-wise and unit group-wise algorithms, and then evaluate the algorithms for computing $topkG^p$ and $topkG^g$ using both synthetic and real datasets.

Since this is the first work for group-based skyline with the new definition of G-Skyline, our performance evaluation was conducted against the enumeration method as a baseline. We implemented the following algorithms in Java and ran experiments on a machine with Intel Core i7 running Ubuntu with 8 GB memory.

- *PWise*: Point-wise algorithm presented in Subsection 4.1.
- *UWise*: Basic unit group-wise algorithm with Superset Pruning and Tail Set Pruning.
- *UWise+*: Refined unit group-wise algorithm with Unit Group Reordering and Subset Pruning.
- *BL*: We enumerate all $\binom{\mathbb{S}_s}{s}$ candidates, and use Theorem 2 to verify each candidate.

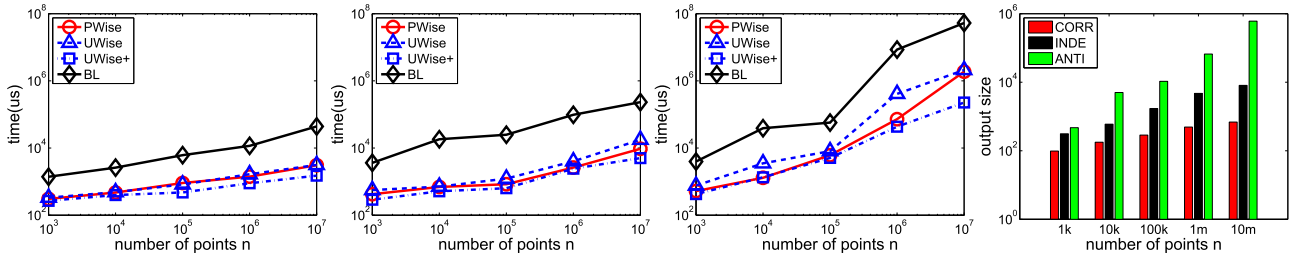
We used both synthetic datasets and a real NBA dataset in our experiments. To study the scalability of our methods, we generated independent (INDE), correlated (CORR), and anti-correlated (ANTI) datasets following the seminal work [4]. We also built a dataset that contains 2384 NBA players who are league leaders of playoffs. The data was extracted from http://stats.nba.com/leaders/alltime/?ls=iref:nba:gnav on 04/15/2015. Each player has five attributes that measure the player's performance. Those attributes are Points (PTS), Rebounds (REB), Assists (AST), Steals (STL), and Blocks (BLK).

6.2 G-Skyline Groups in the Synthetic Data

In this subsection, we report the experimental results for computing G-Skyline groups based on synthetic data.

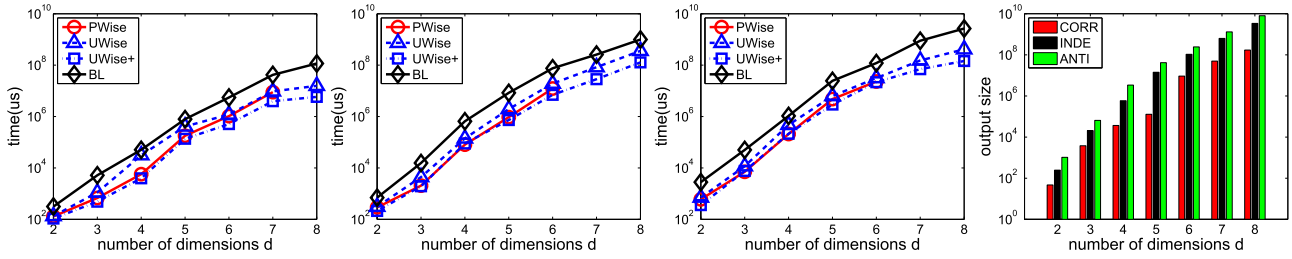
Figs. 11a, 11b, and 11c present the time cost of UWise, UWise+, PWise, and BL with varying number of points n for the three datasets ($d=2, s=4$). Fig. 11d shows the output size with varying n on the three datasets. Because only the points in the first s skyline layers (total number is $\mathbb{S}_s$) are used to compute G-Skyline groups and $\mathbb{S}_s \ll n$ in general, we can see that the time cost and output size are not significantly impacted by n. Figs. 11a, 11b, 11c, and 11d show that the time cost and output size grow approximately linearly with n.

Figs. 12a, 12b, and 12c show the time cost of UWise, UWise+, PWise, and BL with varying number of dimensions d on the three datasets ($n=10000, s=3$). Fig. 12d shows the output size with varying d on the three datasets. The time cost and output size increase exponentially with respect to the increasing d. This is largely due to the increasing number of points in the first s skyline layers. We did not report the result of the PWise algorithm in some figures due



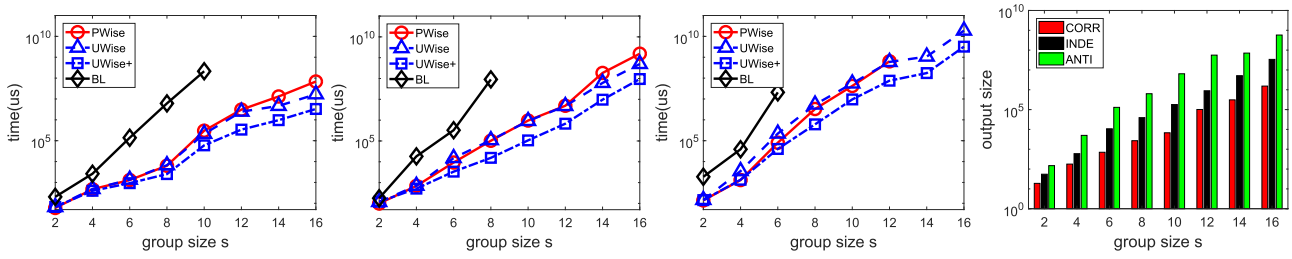
(a) time cost of CORR (b) time cost of INDE (c) time cost of ANTI (d) output size

Fig. 11. Computing G-Skyline groups on synthetic datasets of varying n ($d = 2, s = 4$).



(a) time cost of CORR (b) time cost of INDE (c) time cost of ANTI (d) output size

Fig. 12. Computing G-Skyline groups on synthetic datasets of varying d ($n = 10000, s = 3$).



(a) time cost of CORR (b) time cost of INDE (c) time cost of ANTI (d) output size

Fig. 13. Computing G-Skyline groups on synthetic datasets of varying s ($n = 10000, d = 2$).

to the high space cost of PWise since it needs to generate much more candidates than UWise.

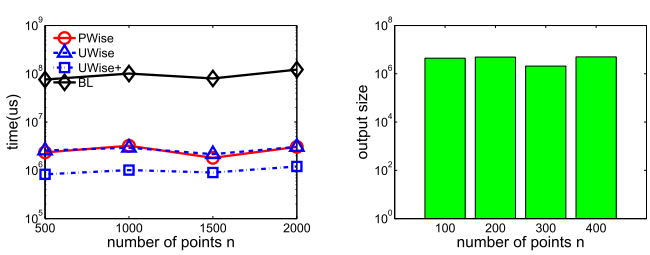
Figs. 13a, 13b, and 13c show the time cost of UWise, UWise+, PWise, and BL with varying group size s on the three datasets ($n = 10000, d = 2$). Fig. 13d shows the output size with varying s on the three datasets. We did not report the result of the BL algorithm in some figures due to the high cost when s is big. The time cost increases exponentially with respect to the increasing s. Furthermore, the group size s has a significant impact on the output size because there are $\binom{\mathbb{S}_s}{s} \approx {\mathbb{S}_s}^s$ candidate groups. Empirically, the result shows that the output size also grows exponentially with s as shown in Fig. 13d.

From the viewpoint of different datasets, the time cost and output size are in increasing order for CORR, INDE, and ANTI, due to the increasing number of points in the first s skyline layers. Comparing different algorithms, UWise, UWise+, and PWise significantly outperform BL, which validates the benefit of our pruning strategies. PWise is better than UWise when s is small but worse when s is big. Furthermore, PWise is highly space-consuming. Both UWise and UWise+ outperform PWise when s is big, which shows the benefit of the unit group notion. UWise+ outperforms UWise, thanks to the Unit Group Reordering and Subset Pruning strategies.

6.3 G-Skyline Groups in the NBA Data

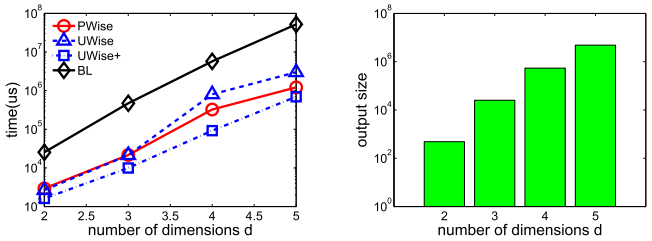
In this subsection, we report the experimental results on the NBA real data.

Fig. 14a shows the time cost with varying n when $d = 5, s = 5$. For each value of $n = 500, 1000, 1500, 2000$, we took the average result based on 100 experiment runs and for each experiment, we randomly chose n out of 2384 players. UWise, UWise+, and PWise again significantly outperform BL due to the efficient pruning strategies and UWise+ performs the best. However, varying n does not have a significant impact on the runtime and output size, as shown in Fig. 14b. The reason is that only the number of points in the



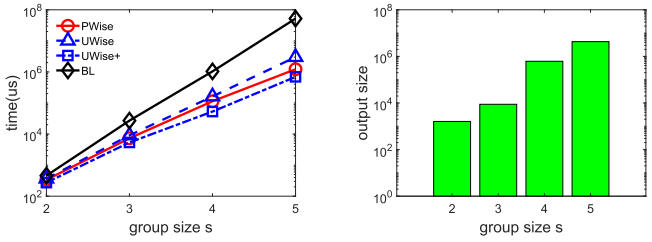
(a) impact of n. (b) impact of n.

Fig. 14. G-Skyline on NBA dataset of varying n.



(a) impact of d. (b) impact of d.

Fig. 15. G-Skyline on NBA dataset of varying d.



(a) impact of s. (b) impact of s.

Fig. 16. G-Skyline on NBA dataset of varying s.

first s layers is used to compute the G-Skyline groups and $\mathbb{S}_s \ll n$ in general.

Figs. 15a and 15b show the time cost and output size for different d when $n = 2384, s = 5$. We took the average result of all possible dimension combinations. We see the number of dimensions d has a large impact on both the runtime and output size which increase with increasing d.

Fig. 16 shows the time cost and output size for different s when $n = 2384, d = 5$. s also has a large impact since the number of points in the first s skyline layers increases significantly as s increases while our approaches are less impacted than the Baseline.

We also report a sample of the final G-Skyline groups in Table 3. There are $\binom{2384}{5} \approx 6.4 \times 10^{14}$ candidate groups, but our algorithm only returns 4865073 G-Skyline groups, i.e., 1 out of 1.3×10^8. We can see the sample groups are formed by elite players with different strengths. For example, G3 is excellent in PTS, REB, AST, STL, and BLK while G1 excels in PTS, and G4 is a good balanced group.

6.4 $topkG^p$/$topkG^g$ in the Synthetic Data

In this subsection, we verify the efficiency and scalability of our proposed algorithms for computing $topkG^p$/$topkG^g$ based on synthetic datasets. We compare our proposed pruning and growing algorithm (PG) with the baseline algorithm (BL) in [41] for computing $topkG^p$ and our proposed pruning algorithm (PA) and counting algorithm (CA) with the baseline algorithm (BL) without pruning for computing $topkG^g$.

6.4.1 $topkG^p$ in the Synthetic Data

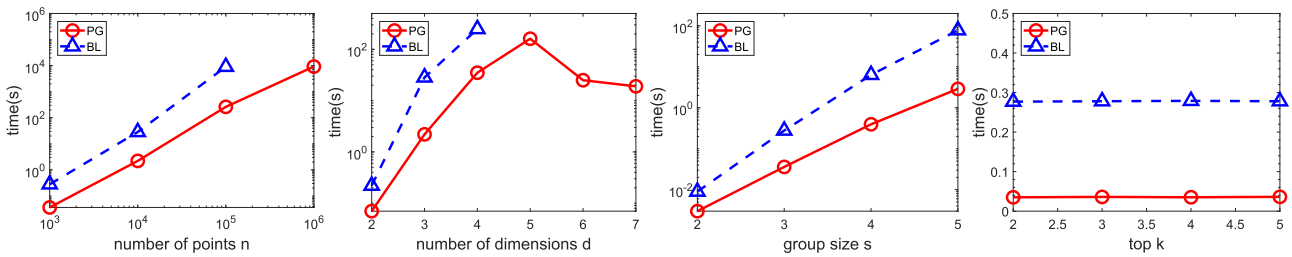
Figs. 17a, 17b, 17c, and 17d show the time cost of varying number of points n, number of dimensions d, group size s, and parameter k on the INDE datasets, respectively. Fig. 17a presents the time cost of PG and BL with varying number of points n for the INDE dataset $(d = 3, s = 3, k = 3)$. The figure shows that PG is about 10 times faster than BL. We did not report the result of the BL algorithm in some figures due to the high cost.

Fig. 17b presents the time cost of PG and BL with varying number of dimensions d for the INDE dataset $(n = 10000, s = 3, k = 3)$. The figure shows PG is much faster than BL again. Furthermore, the time cost decreases with respect to the increasing d when $d > 5$. The reason is that when $d > 5$, most of the points lie in the first s layers and it is very easy to count the number of dominated points.

Fig. 17c presents the time cost of PG and BL with varying group size s for the INDE dataset $(n = 1000, d = 3, k = 3)$. The time cost increases linearly with respect to the increasing s. Fig. 17d presents the time cost of PG and BL with varying top k for the INDE dataset $(n = 1000, d = 3, s = 3)$. The figure shows the parameter k does not have much impact in terms of time cost for both PG and BL.

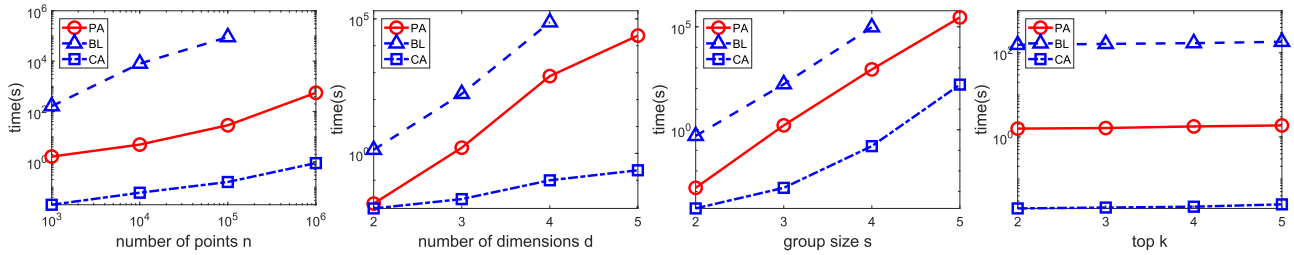
TABLE 3
Sample of G-Skyline Groups on the NBA Dataset

G1	Michael Jordan	Anthony Davis	Kyrie Irving	Allen Iverson	Jerry West	high PTS
G2	Magic Johnson	John Stockton	Isaiah Thomas	Chris Paul	Rajon Rondo	high AST
G3	Michael Jordan	Bill Russell	Magic Johnson	Lance Blanks	Hakeem Olajuwon	high PTS,REB,AST,STL,BLK
G4	Maurice Cheeks	Rich Barry	Slick Watts	Baron Davis	Brad Daugherty	very balanced
G5	Julius Erving	Elvin Hayes	Michael Jordan	Khris Middleton	Alvin Robertson	high STL,BLK



(a) impact of n. (b) impact of d. (c) impact of s. (d) impact of k.

Fig. 17. Computing $topkG^p$ on synthetic datasets.



(a) impact of n. (b) impact of d. (c) impact of s. (d) impact of k.

Fig. 18. Computing $topkG^g$ on synthetic datasets.

6.4.2 $topkG^g$ in the Synthetic Data

Because $topkG^g$ takes too much time to compute the number of dominated groups, in the experiment, we only compute the number of dominated groups in the first s skyline layers. Figs. 18a, 18b, 18c, and 18d show the time cost of varying number of points n, number of dimensions d, group size s, and parameter k on the INDE datasets with the default of $n = 1000, d = 3, s = 3, k = 3$. Figs. 18a, 18b, 18c, and 18d show that our proposed algorithms are much faster than the baseline algorithm.

6.5 $topkG^p$/$topkG^g$ in the NBA Data

In this subsection, we report the top-3 $topkG^p$ groups and top-3 $topkG^g$ groups based on the real NBA data in Tables 4 and 5, respectively. Although there is no overlap group between $topkG^p$ and $topkG^g$, they share most of the same NBA players. For example, Hakeem Olajuwon and Magic Johnson lie in all the groups of both $topkG^p$ and $topkG^g$, which means they are elite NBA players. Furthermore, the top-10 $topkG^g$ groups out of 4865073 G-Skyline groups lie in the top-900 $topkG^p$ groups out of 4865073 G-Skyline groups, which means the definitions of $topkG^p$ and $topkG^g$ are similar to some extent.

7 Conclusions

In this paper, we proposed the problem of G-Skyline groups for finding Pareto optimal groups. This is the first work to extend the original skyline definition to group level which captures the quintessence of original skyline definition. To compute the G-Skyline groups efficiently, we presented a novel structure based on skyline layers that not only partitions the points efficiently but also captures the dominance relationship between the points. We then presented point-wise and unit group-wise algorithms to compute the G-Skyline groups efficiently. To mitigate the drawback of too many returned G-Skyline groups, we proposed the top-k representative G-Skyline groups based on the number of dominated points and the number of dominated groups and presented efficient algorithms for computing them. A comprehensive experimental study is reported demonstrating the benefit of our algorithms.

TABLE 4
Top-3 $topkG^p$ in the NBA Dataset

G1	Michael Jordan	Hakeem Olajuwon	Larry Bird	Wilt Chamberlain	Magic Johnson
G2	Michael Jordan	Hakeem Olajuwon	Larry Bird	Charles Barkley	Magic Johnson
G3	LeBron James	Hakeem Olajuwon	Larry Bird	Wilt Chamberlain	Magic Johnson

TABLE 5
Top-3 $topkG^g$ in the NBA Dataset

G1	LeBron James	Hakeem Olajuwon	Larry Bird	Charles Barkley	Magic Johnson
G2	LeBron James	Hakeem Olajuwon	Larry Bird	Kareem Abdul-Jabbar	Magic Johnson
G3	LeBron James	Hakeem Olajuwon	Charles Barkley	Kareem Abdul-Jabbar	Magic Johnson

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. This work was partially supported by NSF grants IIS-1838200, CNS-1618932, and Air Force Office of Scientific Research (AFOSR) DDDAS Program FA9550-12-1-0240.

References

[1] M. Bai *et al.*, "Discovering the k representative skyline over a sliding window," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2041–2056, Aug. 2016.
[2] J. L. Bentley, K. L. Clarkson, and D. B. Levine, "Fast linear expected-time algorithms for computing maxima and convex hulls," in *Proc. 1st Annu. ACM-SIAM Symp. Discrete Algorithms*, 1990, pp. 179–187.
[3] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson, "On the average number of maxima in a set of vectors and applications," *J. ACM*, vol. 25, no. 4, pp. 536–543, 1978.
[4] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, 2001, pp. 421–430.
[5] T. Cai, R. Li, J. X. Yu, R. Mao, and Y. Cai, "Efficient algorithms for distance-based representative skyline computation in 2D space," in *Proc. Asia-Pacific Web Conf.*, 2015, pp. 116–128.
[6] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2006, pp. 503–514.
[7] L. Chen and X. Lian, "Dynamic skyline queries in metric spaces," in *Proc. Int. Conf. Extending Database Technol.*, 2008, pp. 333–343.
[8] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proc. 19th Int. Conf. Data Eng.*, 2003, pp. 717–719.
[9] E. Dellis and B. Seeger, "Efficient computation of reverse skyline queries," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 291–302.
[10] P. Godfrey, R. Shipley, and J. Gryz, "Algorithms and analyses for maximal vector computation," *VLDB J.*, vol. 16, no. 1, pp. 5–28, 2007.
[11] Z. Huang, H. Lu, B. C. Ooi, and A. K. H. Tung, "Continuous skyline queries for moving objects," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 12, pp. 1645–1658, Dec. 2006.
[12] H. Im and S. Park, "Group skyline computation," *Inf. Sci.*, vol. 188, pp. 151–169, 2012.
[13] D. G. Kirkpatrick and R. Seidel, "Output-size sensitive algorithms for finding maximal vectors," in *Proc. 1st Annu. Symp. Comput. Geom.*, 1985, pp. 89–96.

[14] H. Köhler, J. Yang, and X. Zhou, "Efficient parallel skyline processing using hyperplane projections," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2011, pp. 85–96.
[15] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *J. ACM*, vol. 22, no. 4, pp. 469–476, 1975.
[16] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das, "On skyline groups," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 2119–2123.
[17] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting stars: The K most representative skyline operator," in *Proc. EEE 23rd Int. Conf. Data Eng.*, 2007, pp. 86–95.
[18] J. Liu, L. Xiong, J. Pei, J. Luo, and H. Zhang, "Finding pareto optimal groups: Group-based skyline," *Proc. VLDB Endowment*, vol. 8, no. 13, pp. 2086–2097, 2015.
[19] J. Liu, L. Xiong, and X. Xu, "Faster output-sensitive skyline computation algorithm," *Inf. Process. Lett.*, vol. 114, no. 12, pp. 710–713, 2014.
[20] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 633–644.
[21] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure and efficient skyline queries on encrypted data," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 7, pp. 1397–1411, Jul. 2019.
[22] J. Liu, J. Yang, L. Xiong, J. Pei, and J. Luo, "Skyline diagram: Finding the voronoi counterpart for skyline queries," in *Proc. IEEE 34th Int. Conf. Data Eng.*, 2018, pp. 653–664.
[23] J. Liu, H. Zhang, L. Xiong, H. Li, and J. Luo, "Finding probabilistic k-skyline sets on uncertain data," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 1511–1520.
[24] M. Magnani and I. Assent, "From stars to galaxies: Skyline queries on aggregate data," in *Proc. 16th Int. Conf. Extending Database Technol.*, 2013, pp. 477–488.
[25] M. Magnani, I. Assent, and M. L. Mortensen, "Taking the big picture: Representative skylines based on significance and diversity," *VLDB J.*, vol. 23, no. 5, pp. 795–815, 2014.
[26] R. Mao, T. Cai, R. Li, J. X. Yu, and J. Li, "Efficient distance-based representative skyline computation in 2D space," *World Wide Web*, vol. 20, no. 4, pp. 621–638, 2017.
[27] H. T. H. Nguyen and J. Cao, "Preference-based top-k representative skyline queries on uncertain databases," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2015, pp. 280–292.
[28] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 15–26.
[29] J. Pei, W. Jin, M. Ester, and Y. Tao, "Catching the best views of skyline: A semantic approach based on decisive subspaces," in *Proc. 31st Int. Conf. Very Large Data Bases*, 2005, pp. 253–264.
[30] A. D. Sarma, A. Lall, D. Nanongkai, R. J. Lipton, and J. J. Xu, "Representative skylines using threshold-based preference distributions," in *Proc. IEEE 27th Int. Conf. Data Eng.*, 2011, pp. 387–398.
[31] M. Sharifzadeh and C. Shahabi, "The spatial skyline queries," in *Proc. 32nd Int. Conf. Very Large Data Bases*, 2006, pp. 751–762.
[32] M. Søholm, S. Chester, and I. Assent, "Maximum coverage representative skyline," in *Proc. Int. Conf. Extending Database Technol.*, 2016, pp. 702–703.
[33] Y. Tao, L. Ding, X. Lin, and J. Pei, "Distance-based representative skyline," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2009, pp. 892–903.
[34] Y. Tao, X. Xiao, and J. Pei, "Efficient skyline and top-k retrieval in subspaces," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 8, pp. 1072–1088, Aug. 2007.
[35] A. Vlachou, C. Doulkeridis, and M. Halkidi, "Discovering representative skyline points over distributed data," in *Proc. Int. Conf. Sci. Statistical Database Manage.*, 2012, pp. 141–158.
[36] C. Wang, C. Wang, G. Guo, X. Ye, and P. S. Yu, "Efficient computation of g-skyline groups," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 4, pp. 674–688, Apr. 2018.
[37] G. Wang, J. Xin, L. Chen, and Y. Liu, "Energy-efficient reverse skyline query processing over wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 7, pp. 1259–1275, Jul. 2012.
[38] W. Yu, Z. Qin, J. Liu, L. Xiong, X. Chen, and H. Zhang, "Fast algorithms for pareto optimal group-based skyline," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 417–426.
[39] N. Zhang, C. Li, N. Hassan, S. Rajasekaran, and G. Das, "On skyline groups," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 942–956, Apr. 2014.
[40] W. Zhang, X. Lin, Y. Zhang, W. Wang, and J. X. Yu, "Probabilistic skyline operator over sliding windows," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2009, pp. 1060–1071.
[41] H. Zhu, P. Zhu, X. Li, and Q. Liu, "Top-k skyline groups queries," in *Proc. Int. Conf. Extending Database Technol.*, 2017, pp. 442–445.

Jinfei Liu is a joint postdoctoral research fellow at Emory University and Georgia Institute of Technology. His research interests include skyline queries, data privacy and security, and machine learning. He has published more than 20 papers in premier journals and conferences including the *IEEE Transactions on Knowledge and Data Engineering*, VLDB, ICDE, CIKM, and IPL.

Li Xiong is a professor of Computer Science at Emory University. She conducts research that addresses both fundamental and applied questions at the interface of data privacy and security, spatio-temporal data management, and health informatics. She has published more than 100 papers in premier journals and conferences. She currently serves as associate editor for the *IEEE Transactions on Knowledge and Data Engineering* (TKDE) and on numerous program committees for data management and data security conferences.

Jian Pei is currently a canada research chair (Tier 1) in Big Data Science at Simon Fraser University, Canada. He is one of the most cited authors in data mining, database systems, and information retrieval. Since 2000, he has published one textbook, two monographs, and more than 200 research papers in refereed journals and conferences, which have been cited by more than 83,000 in literature. He was the editor-in-chief of TKDE in 2013–2016, is currently a director of SIGKDD of ACM. He is a fellow of the ACM and of the IEEE.

Jun Luo received the PhD degree in computer science from the University of Texas at Dallas, in 2006. He is a principal researcher at Lenovo Machine Intelligence Center in Hong Kong. His research interests include big data, machine learning, spatial temporal data mining, and computational geometry. He has published more than 90 journal and conference papers in these areas.

Haoyu Zhang is currently working toward the PhD degree at Indiana University Bloomington. His research interests include algorithms and databases. He has published several papers in premier conferences including VLDB, KDD, and CIKM.

Wenhui Yu is currently working toward the PhD degree at Tsinghua University. His research interests include machine learning and data mining. He has published several papers in premier conferences including TKDE, WWW, and CIKM.