Tsinghua Science and Technology

Volume 24 | Issue 2 Article 3

2019

Computing Skyline Groups: An Experimental Evaluation

Haoyang Zhu
the Academy of Military Science of the People's Liberation Army, Beijing 100091, China.

Xiaoyong Li
the Academy of Ocean Science and Engineering, National University of Defense Technology, Changsha 410073, China.

Qiang Liu
the College of Computer, National University of Defense Technology, Changsha 410073, China.

Hao Zhu
the College of Computer, National University of Defense Technology, Changsha 410073, China.

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 05/11 pp171–182
DOI: 10.26599/TST.2018.9010051
Volume 24, Number 2, April 2019

Computing Skyline Groups: An Experimental Evaluation

Haoyang Zhu, Xiaoyong Li*, Qiang Liu, and Hao Zhu

Abstract: Skyline group, also named as combinational skyline or group-based skyline, has attracted more attention recently. The concept of skyline groups is proposed to address the problem in the inadequacy of the traditional skyline to answer queries that need to analyze not only individual points but also groups of points. Skyline group algorithms aim at finding groups of points that are not dominated by any other same-size groups. Although two types of dominance relationship exist between the groups defined in existing works, they have not been compared systematically under the same experimental framework. Thus, practitioners face difficulty in selecting an appropriate definition. Furthermore, the experimental evaluation in most existing works features a weakness, that is, studies only experimented on small data sets or large data sets with small dimensions. For comprehensive comparisons of the two types of definition and existing algorithms, we evaluate each algorithm in terms of time and space on various synthetic and real data sets. We reveal the characteristics of existing algorithms and provide guidelines on selecting algorithms for different situations.

Key words: skyline queries; skyline groups; performance evaluation

1 Introduction

The skyline query is widely used in multi-criteria decision making applications; it retrieves points in a data set that is not dominated by other points. Considering two multi-dimensional points P and Q, P dominates Q iff P is not worse than Q in all dimensions, but is strictly better than Q in at least one dimension. Figure 1 shows a skyline example. The data set in Fig. 1 (left) consists of five points. Assuming that large values are preferred in this paper, as shown in Fig. 1 (right), the skyline contains Q^1,Q^3, and Q^5.

• Haoyang Zhu is with the Academy of Military Science of the People's Liberation Army, Beijing 100091, China. E-mail: zhuhaoyang@nudt.edu.cn.
• Xiaoyong Li is with the Academy of Ocean Science and Engineering, National University of Defense Technology, Changsha 410073, China. E-mail: sayingxmu@nudt.edu.cn.
• Qiang Liu and Hao Zhu are with the College of Computer, National University of Defense Technology, Changsha 410073, China. E-mail: qiangliu06@nudt.edu.cn; haozhu@nudt.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2017-05-22; revised: 2017-12-28; accepted: 2018-01-10

Although skyline computation is particularly useful in multi-criteria decision-making applications, it is inadequate to answer queries that need to analyze not only individual points but also groups of points[1–8]. However, in numerous real-world applications, computing for groups of points that are not dominated by other groups of equal size is highly needed. For example, in fantasy sports games, gamers form their teams by selecting players in the pool of available athletes. Each athlete is represented by a multi-dimensional point. The value of each point dimension is a statistical category of the corresponding athlete. As the gamer can select any athlete from the pool, a team may consist of athletes from different real-world teams. Notably, the gamer will preferably form a team that cannot be dominated by other teams.

Point	A_1	A_2
Q^1	0	4
Q^2	0	2
Q^3	3	2
Q^4	1	1
Q^5	4	0

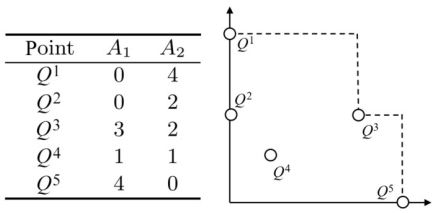


Fig. 1 A skyline example.

The between-group dominance relationship defined in existing works can be divided into two types. In the first type, groups are compared by their point permutations. If one permutation of a group G can dominate a permutation of another group G', then G dominates G'. In the second type, each group is represented by a point. Attribute values of the representative point are aggregated over the corresponding attribute values of all points in the group. Then groups are compared by their representative points as conventional skylines. However, these two definitions have not been compared systematically under the same experimental framework. Hence, difficulty arises during selection of an appropriate definition and the corresponding algorithms. Furthermore, the experimental evaluation in most existing works presents a weakness, that is, the studies only experimented on small data sets or large data sets with small dimensions.

To address these problems, we present comprehensive comparisons on all existing algorithms. We evaluate each algorithm in terms of time and space on various synthetic and real data sets. Based on the experimental results, we reveal the characteristics of existing algorithms and provide guidelines on selecting a suitable definition and the corresponding algorithms for different situations.

We briefly summarize our contributions as follows:

• We provide a comprehensive survey on the two types of skyline group definitions.

• We compare all existing skyline group algorithms through extensive experiments on synthetic and real data sets.

• We report new findings obtained from experimental results. We also analyze the strong and weak points of existing algorithms to guide practitioners in selecting the appropriate algorithms for different situations.

This paper extends a conference paper[9] in several substantial ways. First, we analyze factors in addition to time and space in recommending appropriate algorithms for different situations. Second, we conduct more experiments and provide a more detailed experimental analysis on the performance of existing skyline group algorithms. Finally, we provide new insights into the strengths and weaknesses of existing algorithms and more detailed guidelines on selecting appropriate algorithms for various scenarios.

The rest of the paper is organized as follows. We analyze the two types of skyline group definitions and review related works in Section 2. Section 3 presents the existing skyline group algorithms. We elaborate on the relationships and differences in the existing algorithms in Section 4. We present the experimental results and new insights into the strengths and weaknesses of the existing algorithms; these findings can be used as guidelines in Section 5. Section 6 summarizes the directions of future study and concludes this paper.

2 Preliminary

2.1 Problem definition

First, we introduce the dominance relationship between points. We assume that large values are preferred in this paper. Q^i denotes the i-th point, whereas Q^i_k denotes the value on the k-th dimension of Q^i. For reference, Table 1 provides a summary of frequently used notations.

Definition 1 ($\prec$) Q^i dominates Q^j, denoted as $Q^i \prec Q^j$, iff for each k, $Q^i_k \geqslant Q^j_k$ and for at least one k, $Q^i_k > Q^j_k$ $(1 \leqslant k \leqslant d)$.

The skyline of a data set is a set of points that are not dominated by other points in the data set.

Let $G \prec G'$ denote that G dominates G'. The dominance relationship between groups defined in existing works[1–8] can be divided into two types.

Definition 2 ($\prec_p$)[4,7] We use $\prec_p$ to denote the dominance relationships between groups under permutation. G and G' are two different groups with l points. Assume that $G = \{Q^1, Q^2, ..., Q^l\}$ and $G' = \{Q^{1'}, Q^{2'}, ..., Q^{l'}\}$. We say that $G \prec_p G'$, iff we can find two permutations of l points for G and G', $G = \{Q^{u1}, Q^{u2}, ..., Q^{ul}\}$ and $G' = \{Q^{u1'}, Q^{u2'}, ..., Q^{ul'}\}$ satisfying that for each i, $Q^{ui} \preceq Q^{ui'}$ and for at least one i, $Q^{ui} \prec Q^{ui'}$ $(1 \leqslant i \leqslant l)$.

For instance in Fig. 1, since $Q^1 \prec Q^2$ and $Q^3 \prec Q^4$, thus $\{Q^1, Q^3\} \prec_p \{Q^2, Q^4\}$.

Definition 3 ($\prec_f$)[1–3,5–8] For an aggregate function

Table 1 Summary of notations.

Notation	Description
D	d-dimensional data set
d	Number of dimensions
n	Number of points in D
Q^i	The i-th point in D
Q^i_j	Value on the j-th dimension of Q^i
$\prec$	Preference/dominance relation
Skyline	Skyline of data set D
l	Size of a group

f and a group $G = \{Q^1, Q^2, ..., Q^l\}$, G is represented by a point Q, where $Q_j = f(Q_j^1, Q_j^2, ..., Q_j^l)$. For two distinct groups G and G', Q and Q' represent G and G', respectively. We define $G \prec_f G'$ iff $Q \prec Q'$.

Two kinds of aggregate functions are studied in existing works. The first one is strictly monotone, which implies that $f(Q_j^1, Q_j^2, ..., Q_j^l) > f(Q_j^{1'}, Q_j^{2'}, ..., Q_j^{l'})$ if $Q_j^i \geqslant Q_j^{i'}$ for every $i \in [1, l]$ and $\exists k$ such that $Q_j^k > Q_j^{k'}$, where $1 \leqslant k \leqslant l$. For the strictly monotone function, existing works focus on SUM. For aggregate functions that are not strictly monotone, existing works focus on MAX and MIN. Figure 2 shows the dominance relations under different aggregate functions.

Based on Definitions 2 or 3, skyline group is defined as follows.

Definition 4 (Skyline groups) The l-point skyline groups consist of groups with l points that are not dominated by any other same-size groups.

Assuming that $l = 2$, the skyline groups based on above definitions are obtained, as shown in Table 2.

2.2 Related work

Since the skyline operator[10] was introduced, various proposals of improved algorithms[11–15], query optimizations[16–19], and variations in skyline query[20–24] have been investigated. However, the above works focus on querying individual points, which are inadequate to answer queries that need to analyze not only individual points but also their combinations[1–6].

The concept of skyline groups is proposed to address this problem. Most related works are in Refs. [1–8]. References [1–3, 5–8] investigate the query of skyline groups based on Definition 3 ($\prec_f$). Although many aggregate functions can be considered in calculating representative points, they focus on SUM, MIN, and MAX, which are commonly used in database applications. However, in other cases, selecting a good or a meaningful aggregate function presents difficulty. Thus, the representative point may not fully represent the corresponding group. To address this problem, Liu et al.[4,7] extended the original skyline definition (for individual points) to permutation group-based skyline (for groups), as described in Definition 2 ($\prec_p$) in our paper.

	Point		SUM	MAX	MIN
G	$Q^1(0,4)$	$Q^5(4,0)$	(4,4)	(4,4)	(0,0)
G'	$Q^2(0,2)$	$Q^4(1,1)$	(1,3)	(1,2)	(0,1)
Dominance relation			$G \prec_f G'$	$G \prec_f G'$	$G' \prec_f G$

Fig. 2 Dominance relations under different aggregate functions.

Table 2 Skyline groups under different definitions.

$\prec$	Skyline group
Permutation	$\{Q^1, Q^3\}, \{Q^1, Q^5\}, \{Q^3, Q^4\}, \{Q^3, Q^5\}$
SUM	$\{Q^1, Q^3\}, \{Q^1, Q^5\}, \{Q^3, Q^5\}$
MAX	$\{Q^1, Q^5\}$
MIN	$\{Q^1, Q^2\}, \{Q^1, Q^3\}, \{Q^2, Q^3\}, \{Q^3, Q^4\}, \{Q^3, Q^5\}$

However, these different types of definitions have not been compared systematically under the same experimental framework. Thus, practitioners face difficulty in selecting an appropriate definition. In this paper, we evaluate all the existing algorithms on various synthetic and real data sets. Based on the experimental results, we reveal the characteristics of existing algorithms and provide guidelines on selecting algorithms for different situations.

3 Computing Skyline Groups

In this section, we compare the different group dominance relationships defined in existing works. We also explain how skyline groups are computed under different definitions and report the relations between different definitions.

3.1 Skyline groups under permutation

From Definition 2, we determine that a point in a skyline group cannot be dominated by other points outside the group. If a point Q^i is dominated by a point Q^j outside a group G, then we replace Q^i with Q^j in G. Thus, the newly formed group G' will dominate G under Permutation. Therefore, if a point Q^i is in a skyline group G under Permutation, all points that dominate Q^i must be contained in G.

Definition 5 (unit) A unit of a point Q^i denotes a set of points including point Q^i itself and the points that dominate Q^i. unit_i is used to denote the unit of Q^i. Then we obtain $\text{unit}_i = \{Q | \forall Q \in D \wedge Q \prec Q^i\} \cup \{Q^i\}$.

For instance, as $Q^3 \in$ Skyline, then $\text{unit}_3 = \{Q^3\}$, and given $Q^3 \prec Q^4$, $\text{unit}_4 = \{Q^3, Q^4\}$.

Based on the above definition, Ref. [4] proposed an algorithm named unit-wise+ to unite units together to obtain l-point skyline groups. Assuming that the group size is l, from Definition 5, we determine that if a unit contains more than l points, then the corresponding point will not be contained in any skyline groups. Thus, unit-wise+ first prunes the input data set. Then, unit-

wise+ unites the units of the residual points to form l-point skyline groups.

As shown in Fig. 3, l is set to 2, and Q^2 is pruned because u_2 contains more than 2 points. As u_4 contains 2 points, it serves as the output of a skyline group. Then, unit-wise+ unites the units of Q^1, Q^3, and Q^5 together to form 2-point skyline groups. All skyline groups are shown in red solid boxes in Fig. 3.

3.2 Skyline groups under SUM

To compute the skyline groups based on SUM, existing works[1–3,5,6] adopt similar dynamic programming algorithms. Let Sky_l^n denote the set of l-point skyline groups with regard to $\{Q^1, Q^2, \ldots, Q^n\}$ and Sky_{l-1}^{n-1} the set of $l-1$ point skyline groups with regard to $\{Q^1, Q^2, \ldots, Q^{n-1}\}$. Thus, we obtain the following:

Lemma 1 Under SUM, given $G \in \text{Sky}_l^n$, if $Q^n \in G$, then $G \backslash \{Q^n\} \in \text{Sky}_{l-1}^{n-1}$.

Based on Lemma 1, Sky_l^n is computed as follows:

$$\text{Sky}_l^n = \text{skyline}(\text{Sky}_l^{n-1} + \{G \cup \{Q^n\} | G \in \text{Sky}_{l-1}^{n-1}\}) \quad (1)$$

Similar to Permutation, according to Refs. [1–3,5,6], if a point is dominated by more than $l-1$ points, then it will not be contained in any skyline groups under SUM. Thus, in Algorithm 1, we first prune the input data set. Then, the skyline groups are computed based on Formula (1).

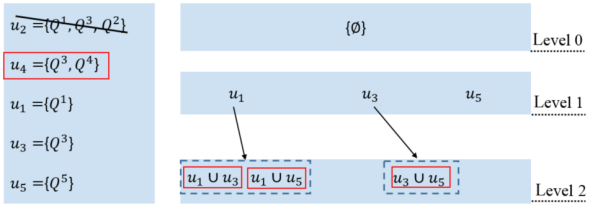


Fig. 3 An example of computing 2-point skyline groups under Permutation.

Algorithm 1 Skyline groups under SUM

Input D, l;
Output Sky_l^n;

```
D ← Input prunning(D);
for j ← 1; j ⩽ |D|; j++ do
    for i ← min(j, l); i ⩾ 1; i−− do
        if i = 1 then
            Sky_1^j = skyline(Sky_1^{j−1} ∪ {D^j});
        else
            Sky_i^j = skyline(Sky_i^{j−1} + {G ∪ {D^j} | G ∈ Sky_{i−1}^{j−1}});
        end if
    end for
end for
return Sky_l^n
```

3.3 Skyline groups under MAX and MIN

3.3.1 Compute aggregate vectors for skyline groups

As many skyline groups based on MAX and MIN share the same aggregate vector, Ref. [5] proposed an algorithm to compute the distinct aggregate vectors for MAX and MIN.

Let Sky_i denote the set of distinct aggregate vectors of i-point skyline groups with regard to D. Thus, we obtain the following:

Lemma 2 Under functions MAX and MIN, given $G \in \text{Sky}_l^n$, if $Q^n \in G$, then a group $G' \in \text{Sky}_{l-1}^{n-1}$ exists, such that $\text{MAX}(G' \cup \{Q^n\}) = \text{MAX}(G)$, and $\text{MIN}(G' \cup \{Q^n\}) = \text{MIN}(G)$.

Based on Lemma 2, the distinct aggregate vectors of skyline groups are computed as follows:

$$\text{Sky}_l = \text{skyline}(\{G \cup \{Q^i\} | G \in \text{Sky}_{l-1} \text{ and } Q^i \notin G\}) \quad (2)$$

Algorithm 2 is proposed in Ref. [5] to compute distinct aggregate vectors of skyline groups under MAX and MIN. Algorithm 2 first prunes the input data set as Algorithm 1. Then, Algorithm 2 computes Sky_i. When Sky_i is computed, then we use Sky_i to compute Sky_{i+1}. Sky_l is the set of distinct aggregate vectors of l-point skyline groups.

3.3.2 Construct skyline groups for MAX and MIN

Once we compute the aggregate vectors for skyline groups based on MAX and MIN, we can construct the

Algorithm 2 Aggregate vectors under MAX and MIN

Input D, l;
Output Sky_l;

```
D ← Input prunning(D);
Sky_1 ← skyline (D);
for i ← 2; i ⩽ l; i++ do
    C_i ← ∅;
    for each G ∈ Sky_{i−1} do
        for each Q ∈ D do
            if Q ∉ G then
                G' ← G ∪ {Q};
            end if
            if G' ∉ C_i then
                C_i ← C_i ∪ {G'};
            end if
        end for
    end for
    Sky_i ← skyline(C_i);
end for
return Sky_l
```

equivalent skyline groups based on an aggregate skyline vector.

For MIN, given a MIN aggregate vector v, the process involves finding a set $\Omega(v)$ with the set of all points that dominate or are equal to v. Assuming that the aggregate vector of arbitrary l points in $\Omega(v)$ is v', we obtain $v' \not\prec v$. Otherwise, v is not an aggregate vector of a skyline group. On the other hand, v' contains no values smaller than v on any attribute, according to the definition of $\Omega(v)$. Thus we achieve $v' = v$, which simplying that any l-point subset of $\Omega(v)$ is an l-point skyline group.

MAX is shown as an Non-deterministic Polynomial hard (NP-hard) problem[5]. The NP-hardness directly follows from the NP-completeness of SET-COVER. For an aggregate skyline vector v under MAX, we need to find points in the data sets that cover all v's attribute values. Notably, for MAX, if $l \geqslant d$, then only one distinct aggregate skyline vector with max values exists for all attributes. In Ref. [5], Zhang et al. proposed a brute-force enumeration method to construct the skyline groups.

4 Comparing Different Definitions

4.1 Relationships between Permutation and SUM

We find that skyline groups of a data set under SUM is a subset of skyline groups under Permutation.

Lemma 3 The skyline groups of a data set based on SUM are a subset of skyline groups based on Permutation.

Proof: Assume that G is a skyline group based on SUM but G is dominated by a group G' under Permutation. Then, G' contains at least one point Q' that is better than another point Q in G, and the remaining points of G' are equal or better than the rest of points in G. Thus, $\text{SUM}(G') \prec \text{SUM}(G)$, contradicting the notation that G is a skyline group based on SUM.

We can prove that Lemma 3 is not only true for SUM but also for other strictly monotone functions.

The above lemma indicates that the dominance relationship of SUM is less strict than Permutation, and more candidate groups can be dominated by other groups. Thus, the output of SUM is a subset of the output of Permutation with the same group size on the same input data set.

4.2 Relationships between MAX, MIN, and SUM

We find that Algorithm 1 can be used to compute aggregate vectors for skyline groups under MAX and MIN.

Lemma 4 The aggregate vectors of skyline groups that are derived from employing Formula (1) can cover all the distinct aggregate vectors of skyline groups under MAX and MIN.

Proof: Assume that G is an l-point skyline group with aggregate vector v and $Q \in G$. Consider $G' = G \backslash \{Q\}$. If G' is an $(l-1)$-point skyline group, it satisfies Formula (1). Otherwise, there must be an $(l-1)$-point skyline group G^* over $D \backslash \{Q\}$ satisfying $G^* \prec G'$. For MAX and MIN, we prove that $\text{MAX}(G) = \text{MAX}(G^* \cup \{Q\})$ and $\text{MIN}(G) = \text{MIN}(G^* \cup \{Q\})$. Since $G^* \prec G'$, we have $G^* \cup \{Q\} \prec G$ or $G^* \cup \{Q\} = G$. As G is a skyline group over D, then $G^* \cup \{Q\} \not\prec G$. Thus, $G^* \cup \{Q\} = G$. Therefore, we get that $\text{MAX}(G) = \text{MAX}(G^* \cup \{Q\})$ and $\text{MIN}(G) = \text{MIN}(G^* \cup \{Q\})$.

The above lemma indicates that we can apply Algorithm 1 to compute the aggregate vectors for MAX and MIN. The difference between the results of Algorithms 1 and 2 is that the result of Algorithm 1 contains the skyline groups that share the same aggregate skyline vector. Therefore, the output of Algorithm 2 is a subset of the output of Algorithm 1 under MAX and MIN.

Based on the above discussion, Algorithm 1 can be applied to compute a mixture of different aggregate functions on different attributes. Algorithm 1 can deal with arbitrary mixture of SUM, MIN, and MAX, whereas Algorithm 2 can deal with any mixture of MIN and MAX.

4.3 Output size analysis

From Definition 2, we know that any l skyline points can form a skyline group under Permutation. Thus the output size of skyline groups under Permutation can be extremely large. For instance, 1000 skyline points are available, and l is set to 5. Then, more than $\text{C}_{1000}^{5} \approx 10^{13}$ skyline groups are available, and this number is almost prohibitive to compute. Therefore, the large output size is a potential limitation of skyline group operator under Permutation.

Similar to Permutation, the output size of MAX can be extremely large when $l \geqslant d$, as it takes at most d points to cover the maximum values of all dimensions. Thus, after finding the points that cover the maximum values, the remaining points can be arbitrary. For instance, $n = 10^6$, $d = 3$, and $l = 5$. Then more than $\text{C}_{n}^{l-d} \approx 10^{12}$ skyline groups are obtained. Therefore,

the large output size is also a potential limitation of skyline group operator under MAX.

Although the output sizes of SUM and MIN are small compared with those of Permutation and MAX, the output sizes of SUM and MIN are always larger than the size of the input data set. Thus, the large output size is a potential limitation shared by all skyline group definitions. Employing top-k queries on skyline groups is one way to solve this problem. We plan to investigate this problem in the future.

4.4 Time complexity analysis

Consider unit-wise+. Let $|U|$ denote the number of units used in unit-wise+. From Fig. 1, we know that for each new built group in unit-wise+, we need to check whether it is a skyline group. Therefore, the time complexity of unit-wise+ is $C^1_{|U|} + C^2_{|U|} + \cdots + C^l_{|U|} = \sum_{i=1}^{l} C^i_{|U|}$ in the worst case.

Consider Algorithm 1. Assume that n points are left in D after pruning. Thus, for each j ($j \leqslant n$), l subproblems exist: $\{\mathrm{Sky}^j_1, \mathrm{Sky}^j_2, ..., \mathrm{Sky}^j_l\}$. In total, $n \times l$ sub problems exist. For each sub problem Sky^j_i, at most C^i_j groups. Thus, the time complexity of finding skyline on these groups is $O((\mathrm{C}^i_j)^2)$. Therefore, the time complexity of Algorithm 1 in the worst case is $O(n \times l \times (\mathrm{C}^l_n)^2)$.

Consider Algorithm 2. Assume that n points are left in D after pruning. Thus, for each i ($i \leqslant l$), the time complexity of computing C_i is $O(n \times |\mathrm{Sky}_{i-1}| \times |C_i|) = O(n \times (\mathrm{C}^i_n)^2)$. The time complexity of computing skyline of C_i is $O(|C_i|^2) = O((\mathrm{C}^i_n)^2)$. Thus, for each i ($i \leqslant l$), the time complexity of computing Sky_i is $O(n \times (\mathrm{C}^i_n)^2 + (\mathrm{C}^i_n)^2)$. Therefore, the time complexity of Algorithm 2 is $O(n \times l \times (\mathrm{C}^l_n)^2)$.

From the above analysis, we know that the time complexity of Algorithm 1 is larger than that of unit-wise+. This result coincides with Lemma 3, as Algorithm 1 requires more time to refine the output.

In general, compared with the traditional skyline computation, computing skyline groups is much more complicated. One of our future works aims to design parallel algorithms using modern computing platforms.

5 Experimental Study

In this section, we conduct extensive experiments to compare the performance and the scalability of existing skyline group algorithms. All our experiments are carried out on the same machine with 64 GB memory and dual octa-core Intel Xeon E7-4820 processors clocked at 2.0 GHz. We implement the following algorithms in C++.

unit-wise+: The algorithm to compute the skyline groups based on Permuation. Three algorithms, point-wise, unit-wise, and unit-wise+, are proposed to compute the skyline groups based on Permutation. unit-wise+ exhibits the best performance with respect to time and space.

DPSG (Algorithm 1): The dynamic programming algorithm to compute skyline groups based on Formula (1).

DPAV (Algorithm 2): The algorithm to compute distinct aggregate skyline vectors based on Formula (2).

MAXG and **MING**: MAXG and MING are the algorithms to construct skyline groups based on aggregate skyline vectors under MAX and MIN respectively.

To study the scalability of our algorithms, we generate correlated, independent, and anti-correlated data sets using the standard skyline data generator from Ref. [10]. For real data sets, we use the NBA data set described in Ref. [25]. The data set consists of 17 264 players, and each player features 8 attributes.

5.1 Output analysis

In Tables 3 and 4, G denotes the number of skyline groups, and V denotes the number of distinct aggregate vectors of the skyline groups. Tables 3 and 4 show the output sizes of skyline groups and aggregate skyline vectors for correlated synthetic data sets, respectively. Similar experimental results are obtained on the NBA data set.

Notably, a skyline group under Permuation processes on aggregate skyline vector. We can observe that SUM rarely features two skyline groups that share the same aggregate skyline vector, because the SUM aggregate vector of a skyline group is sensitive to all group points. Among the three aggregate functions, MAX results in the most equivalent groups that share the same aggregate skyline vector. In computing the aggregate vector of a skyline group, SUM reflects the power of all points in the group, whereas MIN (MAX) selects the weakest (strongest) point on each dimension. As a consequence, skyline groups are almost formed by the points with the lowest (highest) value on one dimension.

From the two tables, we can identify that G quickly

Table 3 G and V, under various n, l (d=6).

n		$l=2$				$l=4$				$l=6$			
		SUM	MAX	MIN	Def.2	SUM	MAX	MIN	Def.2	SUM	MAX	MIN	Def.2
2×10^5	G	212	324	101	1747	1461	1129	225	5×10^5	5134	6	424	6.3×10^7
	V	212	235	101		1461	353	225		5134	1	424	
4×10^5	G	258	323	120	2268	2591	752	361	9×10^5	11 757	10	768	1.52×10^8
	V	258	284	118		2591	396	357		11 757	1	762	
6×10^5	G	248	1584	120	2760	2564	398	369	1.3×10^6	15 145	36	742	2.79×10^8
	V	248	300	120		2564	321	369		15 145	1	736	
8×10^5	V	163	302	88	1706	1106	2122	248	5×10^5	5036	160	448	8.3×10^7
	V	163	215	88		1106	196	248		5036	1	448	
1×10^6	G	173	370	89	2004	1244	4318	268	7×10^5	5229	450	455	1.15×10^8
	V	173	228	89		1244	222	268		5229	1	449	

Table 4 G and V, under various d, l (n=1×10^6).

d		$l=2$				$l=4$				$l=6$			
		SUM	MAX	MIN	Def.2	SUM	MAX	MIN	Def.2	SUM	MAX	MIN	Def.2
3	G	3	1×10^6	4	11	7	∞	21	159	16	∞	3006	813
	V	3	1	2		7	1	3		16	1	4	
4	G	19	64	11	167	56	70	12	5375	117	∞	22	83 155
	V	19	17	9		56	1	12		117	1	16	
5	G	65	118	35	630	245	640	47	75 674	591	1×10^8	50	4.2×10^6
	V	65	52	35		245	25	47		117	1	50	
6	G	173	370	89	2004	1244	4318	268	7×10^5	5229	450	455	1.15×10^8
	V	173	228	89		1244	222	268		5229	1	449	
7	G	285	700	177	4475	3216	7723	605	3.5×10^6	18 541	3912	1276	1.2×10^9
	V	285	537	177		3216	1320	605		18 541	99	1276	

enlarges under Permutation, SUM, and MAX. Notably, for MAX, if $l \geqslant d$, only one distinct aggregate skyline vector exists, but the size of the skyline groups can be extremely large. For instance, in Table 4, when $d = 4$ and $l = 6$, 4 points at most are needed to cover the MAX values of all attributes. Then, the remaining 2 points can be arbitrary. Thus, more than 500 billion skyline groups are prohibitive to compute. Moreover, the output size of skyline groups under SUM is consistently less than that under Permutation, and this condition coincides with Lemma 3. In general, Permutation results in the largest number of skyline groups.

The large output size is less informative, and users may face difficulty int making a good, and quick selection. Thus, large output size is a potential limitation for all skyline groups definitions. One way to solve this problem is to select the best k skyline groups. We plan to introduce the top-k dominating queries on the skyline groups in the future.

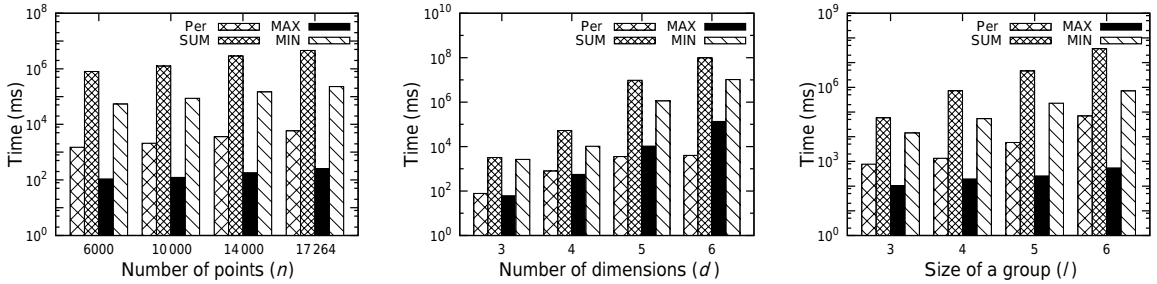
5.2 Experiments on NBA data set

In this section, we report the performance of our algorithms under the different skyline groups definitions. We experiment with the NBA data set and explore several factors, n, d, and l.

5.2.1 Study of runtime

As shown in Lemma 4, DPSG can be used to compute the aggregate vectors for the skyline groups under MAX and MIN. We report the runtime of unit-wise+, DPSG(SUM), DPSG(MAX), and DPSG(MIN) in Fig. 4. We find that SUM is the most expensive among all definitions. Compared with Permutation, as shown in Lemma 3, the output under SUM is a subset of that under Permutation, indicating that SUM requires more time to refine the output. Compared with MAX and MIN, as SUM is sensitive to all the points in the group, it features the largest number of skyline vectors. Thus, SUM needs to conduct much more dominance tests between groups. Therefore, SUM consumes much more time than MAX and MIN. In general, Permutation, MAX, and MIN outperform SUM by three, four, and two orders of magnitude, respectively.

For MAX and MIN, the number of skyline vectors



(a) Runtime on the NBA data set of varying n($d = 4, l = 5$)

(b) Runtime on the NBA data set of varying d ($n = 17\,264, l = 3$)

(c) Runtime on the NBA data set of varying l ($n = 17\,264, d = 4$)

Fig. 4 Runtime on NBA data set.

of MAX results is much lower than that of MIN. For instance, if $l \geqslant d$, one skyline vector is under MAX. Therefore, as shown in Fig. 4, the runtime of MAX is much shorter than that of MIN.

Considering the three subfigures in Fig. 4, the runtime of all definitions grows exponentially with increasing d and l and grows linearly with increasing n. Visibly, the runtime grows exponentially with increasing l. For d, the number of candidate points increases with increasing d. Thus the input pruning is less effective when d is large. Furthermore, more candidate points can build a skyline group. As a result, the runtime grows exponentially with increasing d. For n, the number of candidate points shows no significant with increasing n for a fixed d. Thus, the runtime grows linearly with increasing l.

5.2.2 Study of output

We report the number of skyline groups of different definitions in Fig. 5. Similar to Fig. 4, the output size of all definitions grows exponentially with increasing d and l and grows linearly with increasing n. We observe that when $l \leqslant d$, Permutation results in the largest number of skyline groups. Among the three aggregate functions, SUM exhibits the largest number of skyline groups when $l \leqslant d$, whereas MAX features the smallest number of skyline groups.

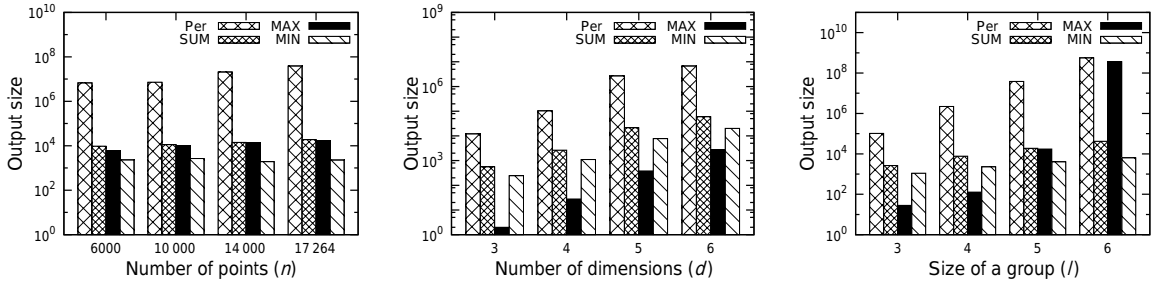
However, when $l > d$, as at most d points are needed to cover the MAX values on all attributes, the remaining points in a skyline group under MAX can be arbitrary. Thus, the output size can be extremely large under MAX when $l > d$. For instance, as shown in Fig. 5c, when $d = 6$, the output size of MAX is the largest among SUM, MAX, and MIN.

As discussed in Section 4.3, any l skyline points can form a skyline group under Permutation. Thus, the output size of the skyline groups under Permutation grows exponentially with increasing l. Compared with MAX, the output size of Permutation is less than that of MAX when l is far larger than d. As for MAX, any points can be used to form a skyline points, but only skyline points can be used to build a skyline group under Permuation.

5.2.3 Comparison of DPSG and DPAV

From Lemma 4, we can apply DPSG (MAX) and DPSG (MIN) to compute the aggregate skyline vectors for MAX and MIN, respectively. We report the runtime of DPSG and DPAV for MAX and MIN in Figs. 6 and 7, respectively. When $l \geqslant 4$, DPSG (MAX) and DPSG (MIN) is about on average, 10× and 2× speedup over DPAV (MAX) and DPAV (MIN), respectively.

Next, we report the performances of MAXG and MING. As shown in Fig. 6, the runtime of MAXG is not very long although we apply a brute-force method. This result is attributed to the small number of points

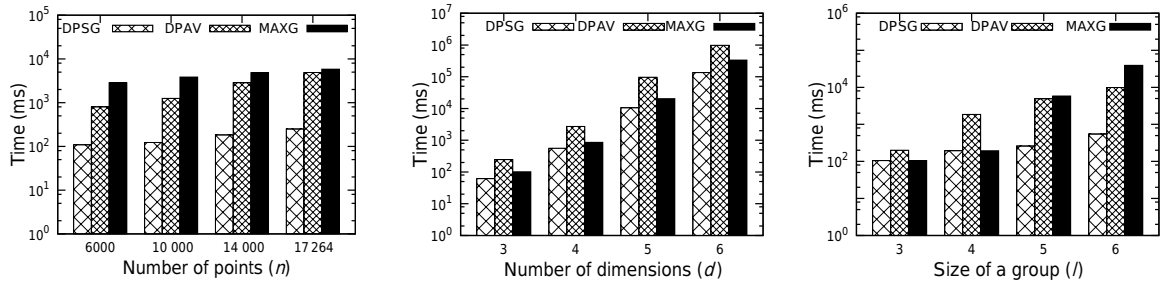


(a) Output size on NBA data set of varying n ($d = 4, l = 5$)

(b) Output size on NBA data set of varying d ($n = 17\,264, l = 3$)

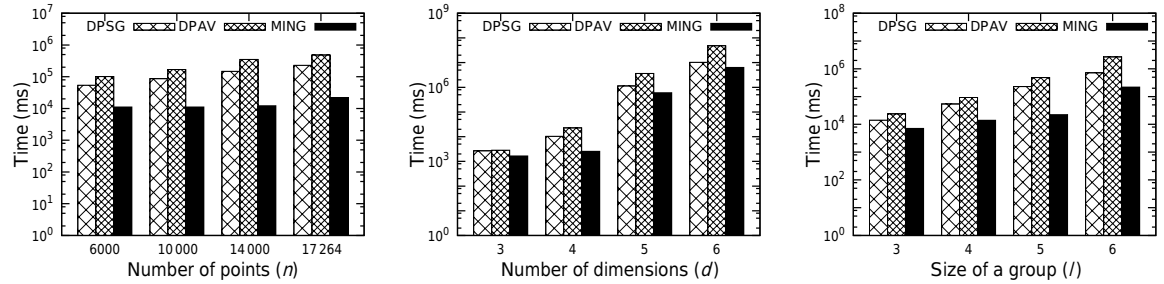
(c) Output size on NBA data set of varying l ($n = 17\,264, d = 4$)

Fig. 5 Output size on NBA data set.



(a) Results on the NBA data set of varying n ($d = 4, l = 5$) (b) Results on the NBA data set of varying d ($n = 17\,264, l = 3$) (c) Results on the NBA data set of varying l ($n = 17\,264, d = 4$)

Fig. 6 Experimental results on NBA data set under MAX.



(a) Results on the NBA data set of varying n ($d = 4, l = 5$) (b) Results on the NBA data set of varying d ($n = 17\,264, l = 3$) (c) Results on the NBA data set of varying l ($n = 17\,264, d = 4$)

Fig. 7 Experimental results on NBA data set under MIN.

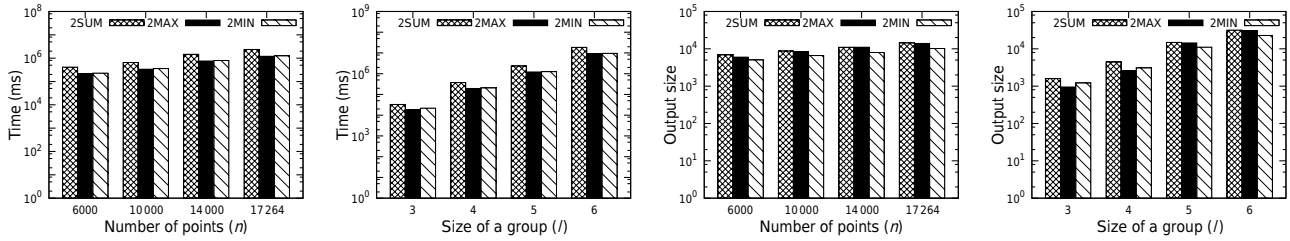
that "hit" the MAX attribute values in an aggregate skyline vector. Thus, in practice, constructing skyline groups under MAX becomes efficient. However, given an only one skyline vector, we observe that when $l > d$, MAXG consumes much more time than DPSG (MAX). The skyline vector takes the MAX value on every attribute. Therefore, when building the skyline groups, after finding the points that achieve the skyline vector, the remaining points can be arbitrary. As a result, the number of skyline groups can be extremely large when $l > d$. For instance, in Fig. 6c, when $l = 6$, more than 3×10^8 skyline groups exist. Thus, the runtime of MAXG is extremely large than that of DPSG (MAX). We report the runtime of MING in Fig. 7. Construct the skyline groups by applying MING becomes very efficient, as the number of points that dominate a skyline vector v under MIN is small. In return, remarkably efficient building of the skyline groups based on v is also achieved.

5.2.4 Mixture of different aggregate functions

Figure 8 shows the runtime and the output size of Algorithm 1 over the NBA data set. A mixture of SUM, MAX, and MIN is applied on the four attributes. For instance, 2SUM means the SUM function on the first two attributes and MAX and MIN on the remaining two attributes. As observed in Figs. 8a and 8b, 2SUM is much more expensive than 2MAX and 2MIN, coinciding with the result in Fig. 4. Moreover, from Figs. 8c and 8d, 2SUM results in a larger output size than 2MAX and 2MIN. This result is attributed to increased difficulty for a group to dominate other groups under SUM.

5.3 Experiments on synthetic data sets

To evaluate the scalability of existing algorithms, we experiment on correlated, independent, and anti-



(a) $l = 5$ (b) $n = 17\,264$ (c) $l = 5$ (d) $n = 17\,264$

Fig. 8 (a) and (b): Runtime; (c) and (d): output size, mixture of SUM, MAX, and MIN.

correlated data sets using the standard skyline data generator from Ref. [10]. The experimental results are shown in Figs. 9 – 11.

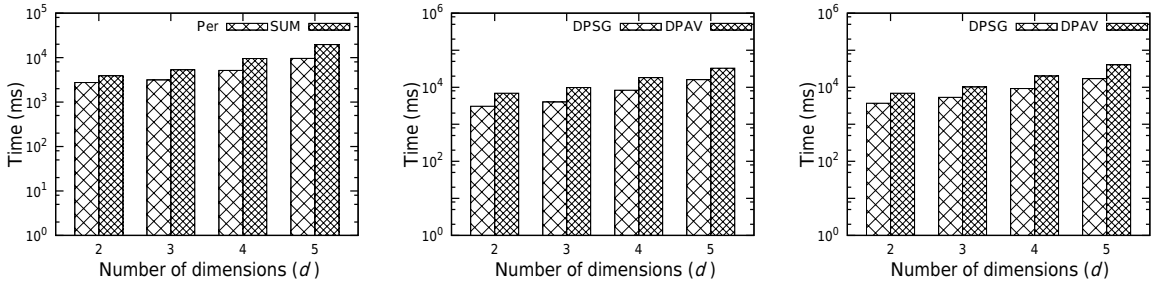
We observe that the runtime varies dramatically under different data distributions. We evaluate the algorithms on three data sets of equal cardinalities with different number of dimensions. As shown in Fig. 9, all the algorithms perform efficiently on the correlated data sets, as the input pruning is also very effective. On independent data sets and anti-correlated data sets, the runtime increases quickly, and the algorithm cannot finish within reasonable amount of time. As shown in Fig. 11, we omit the bars that the algorithm cannot finish within reasonable amount of time. On independent data sets and anti-correlated data sets, a point experiences difficulty in dominating other points. Thus, the input pruning is less effective in these data sets. As a result, the runtime on independent data sets and anti-correlated data sets are extremely larger than that of correlated data sets.

5.4 Characteristics of existing algorithms

Based on the experimental results on the NBA data sets and synthetic data sets, we reveal the characteristics of existing algorithms; these characteristics can serve as guidelines on selecting the appropriate algorithms for various scenarios. We summarize the characteristics of existing algorithms in Table 5.

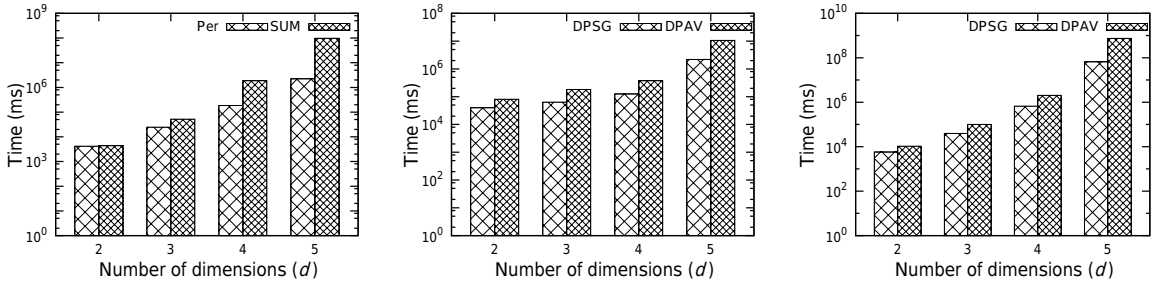
Table 5 Characteristics of existing algorithms.

Algorithm	Definition	Runtime	Output
unit-wise+	Permutation	short	large
DPSG	SUM	medial	medial
	MAX	short	small
	MIN	long	small
DPAV	MAX	long	small
	MIN	long	small



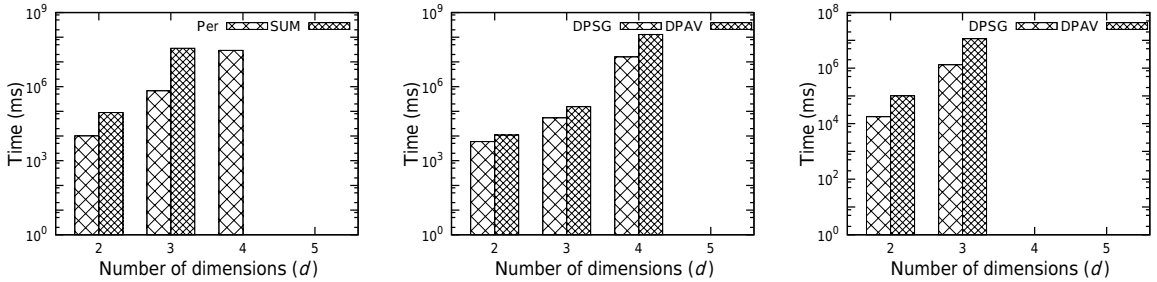
(a) Permutation and SUM (b) MAX (c) MIN

Fig. 9 Experimental results on correlated data sets (n=1×10^6, l=2).



(a) Permutation and SUM (b) MAX (c) MIN

Fig. 10 Experimental results on independent data sets (n=1×10^6, l=2).



(a) Permutation and SUM (b) MAX (c) MIN

Fig. 11 Experimental results on anti-correlated data sets (n=1×10^6, l=2).

We recommend employing the DPSG to compute the aggregate skyline vectors under MAX and MIN, because it is much more efficient than DPAV. For MAX and MIN, we recommend the use of aggregate skyline vectors instead of skyline groups, as the number of equivalent groups can be extremely large when $l > d$.

To select an appropriate skyline group definition, the intrinsic of a definition should be the first consideration. For instance, if the problem is based on cask principle, MIN is the best option. If we could not find an appropriate aggregate function, Permutation is a good option.

In practice, besides the intrinsic, the characteristics of a definition should be considered. If the application focuses on the performance and disregards the output size, we recommend using Permutation. On the other hand, if the application needs to balance between performance and output size, we recommend DPSG (SUM), because features moderate performance and output size. Moreover, as the data distributions will significantly influence the performance of the algorithms, we occasionally need to preprocess the data set before computing the skyline groups. For instance, we can select the important dimensions and omit the less important ones or use the data subset.

6 Conclusion and Future Work

We provide a detailed survey on the existing skyline groups algorithms. We also reveal the characteristics of existing algorithms; these characteristic can be used as guidelines on selecting the appropriate algorithms for various scenarios. We determine that the output sizes of the skyline groups under all the definitions can be extremely large, which is a potential limitation of the skyline group operator. A direction for future study is to design top-k algorithms to help users in making a good and rapid selection. Moreover, computing the skyline groups is much more expensive than computing the traditional skyline. Thus, we plan to design parallel algorithms using modern computing platforms to save computational time.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61502511, 61501482, and 61502513).

References

[1] H. Im and S. Park, Group skyline computation, *Inf. Sci.*, vol. 188, no. 1, pp. 151–169, 2012.

[2] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das, On skyline groups, in *21st International Conference on Information and Knowledge Management*, 2012, pp. 2119–2123.

[3] Y. Chung, I. Su, and C. Lee, Efficient computation of combinatorial skyline queries, *Inf. Syst.*, vol. 38, no. 3, pp. 369–387, 2013.

[4] J. Liu, L. Xiong, J. Pei, J. Luo, and H. Zhang, Finding pareto optimal groups: Group-based skyline, *PVLDB*, vol. 8, no. 13, pp. 2086–2097, 2015.

[5] 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, 2014.

[6] I. Su, Y. Chung, and C. Lee, Top-$k$combinatorial skyline queries, in *Database Systems for Advanced Applications, 15th International Conference*, 2010, pp. 79–93.

[7] H. Zhu, P. Zhu, X. Li, and Q. Liu, Top-k skyline groups queries, in *Proceedings of the 20th International Conference on Extending Database Technology*, 2017, pp. 442–445.

[8] X. Guo, H. Li, A. Wulamu, Y. Xie, and Y. Fu, Efficient processing of skyline group queries over a data stream, *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 29–39, 2016.

[9] H. Zhu, P. Zhu, X. Li, and Q. Liu, Computing skyline groups: An experimental evaluation, in *Proceedings of the ACM Turing 50th Celebration Conference*, 2017, pp. 1–48.

[10] S. B¨orzs¨onyi, D. Kossmann, and K. Stocker, The skyline operator, in *Proceedings of the 17th International Conference on Data Engineering,* 2001, pp. 421–430.

[11] P. Zhang, C. Zhou, P. Wang, B. J. Gao, X. Zhu, and L. Guo, E-tree: An efficient indexing structure for ensemble models on data streams, *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 461–474, 2015.

[12] D. Kossmann, F. Ramsak, and S. Rost, Shooting stars in the sky: An online algorithm for skyline queries, in *Proceedings of 28th International Conference on Very Large Data Bases*, 2002, pp. 275–286.

[13] X. Li, Y. Wang, X. Li, X. Wang, and J. Yu, GDPS: An efficient approach for skyline queries over distributed uncertain data, *Big Data Research*, vol. 1, no.1, pp. 23–36, 2014.

[14] X. Li, Y. Wang, X. Li, and Y. Wang, Parallel skyline queries over uncertain data streams in cloud computing environments, *IJWGS*, vol. 10, no. 1, pp. 24–53, 2014.

[15] K. C. K. Lee, B. Zheng, H. Li, and W. Lee, Approaching the skyline in Z order, in *Proceedings of the 33rd International Conference on Very Large Data Bases*, 2007, pp. 279–290.

[16] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, Skyline with presorting, in *Proceedings of the 19th International Conference on Data Engineering*, 2003, pp. 717–719.

[17] I. Bartolini, P. Ciaccia, and M. Patella, Efficient sort-based skyline evaluation, *ACM Trans. Database Syst.*, vol. 33, no. 4, pp. 1–31, 2008.

[18] S. Zhang, N. Mamoulis, and D. W. Cheung, Scalable skyline computation using object-based space partitioning, in *Proceedings of the International Conference on Management of Data*, 2009, pp. 483–494.

[19] J. Lee and S. Hwang, Scalable skyline computation using a balanced pivot selection technique, *Inf. Syst.*, vol. 39, no. 1, pp. 1–21, 2014.
[20] J. Pei, B. Jiang, X. Lin, and Y. Yuan, Probabilistic skylines on uncertain data, in *Proceedings of the 33rd International Conference on Very Large Data Bases*, 2007, pp. 15–26.
[21] H. Lu, C. S. Jensen, and Z. Zhang, Flexible and efficient resolution of skyline query size constraints, *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 7, pp. 991–1005, 2011.
[22] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. Yu, and Q. Zhang, Efficient computation of the skyline cube, in *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005, pp. 241–252.
[23] Q. Zhang, P. Ye, X. Lin, and Y. Zhang, Skyline probability over uncertain preferences, in *Joint 2013 EDBT/ICDT Conferences*, 2013, pp. 395–405.
[24] X. Zhang, G. Li, and J. Feng, Crowd-sourced top-k algorithms: An experimental evaluation, *PVLDB*, vol. 9, no. 8, pp. 612–623, 2016.
[25] S. Chester, D. Sidlauskas, I. Assent, and K. S. Bøgh, Scalable parallelization of skyline computation for multi-core processors, in *31st IEEE International Conference on Data Engineering*, 2015, pp. 1083–1094.

Haoyang Zhu received the PhD degree in computer science and technology from National University of Defense Technology, China in 2017. He is a member of CCF and ACM. Now he is a research associate in the Academy of Military Science of the People's Liberation Army. His current research interests include massive data processing and query processing and optimization.

Xiaoyong Li received the PhD degree in computer science and technology from National University of Defense Technology, China in 2013. He is a member of CCF, IEEE, and ACM. Now he is a research associate in the National University of Defense Technology. His current research interests include data stream management, parallel computing, uncertain queries, and cloud computing.

Qiang Liu received the PhD degree in computer science and technology from National University of Defense Technology, China in 2014. He is a member of CCF and IEEE. Now he is an assistant professor in the National University of Defense Technology. His current research interests include 5G network, Internet of Things, and machine learning.

Hao Zhu received the PhD degree in the SNE group from Delft University of technology, in 2015. Now he is a researcher in the College of Computer Science, National University of Defense Technology. His research interests are in the area of green computing and GPU computing. To be particular, he studies power estimation models, power management of heterogeneous system, and real time GPU computing.