Computing Skyline Groups: An Experimental Evaluation

Haoyang Zhu†, Peidong Zhu†§, Xiaoyong Li†‡, Qiang Liu†
†College of computer, ‡Academy of Ocean Science and Engineering,
National University of Defense Technology, Changsha, China
§Department of Math and Computer Science, Changsha University, Changsha, China
{zhuhaoyang,pdzhu,sayingxmu}@nudt.edu.cn,qiangl.ne@hotmail.com

ABSTRACT

Skyline group also denoted as combinational skyline or group-based skyline has attracted more and more attention recently. Skyline group algorithms aim at finding groups of points that are not dominated by any other groups of the same size. There are two types of dominance relationship between groups defined in existing works. However, these two different definitions have not been compared systematically under the same experimental framework. Thus it is rather difficult for a practitioner to select an appropriate definition. In this paper, we present a comprehensive comparison of the two types of definition and algorithms in existing works. We reveal characteristics of existing algorithms and provide guidelines on selecting algorithms for different situations.

CCS CONCEPTS

• **Information systems** → **Retrieval efficiency**; *Information retrieval*; Information systems;

KEYWORDS

Skyline queries, skyline groups, performance evaluation

1 INTRODUCTION

The skyline query [2] is widely used in multi-criteria optimal decision making applications, which aims at retrieving points that are not dominated by other points in a data set. Given two multi-dimensional points P and Q, P dominates Q *iff* P is not worse than Q in all dimensions but strictly better than Q in at least one dimension. Since the skyline operator was introduced, many proposals of improved algorithms [6, 8, 10, 11, 17], query optimizations [1, 3, 7, 19] and variations of skyline query [13–15, 18] have been investigated. However, the above works focus on querying *individual* points thus they are inadequate to answer queries that need to analyze not only *individual* points but also their combinations [4, 5, 9, 12, 16].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ACM TUR-C '17, May 12-14, 2017, Shanghai, China
© 2017 ACM. ISBN 978-1-4503-4873-7/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3063955.3064804

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

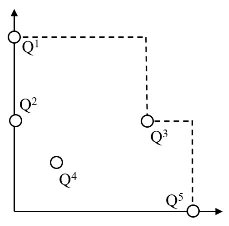


Figure 1: A skyline example

Specifically, in many real-world applications, we need to find groups of points that are not dominated by other groups of equal size. For instance, in the fantasy sports, a gamer forms his or her team by selecting athletes in the pool of available athletes. Thus a team may consist of athletes who are not in the same real-world team. Each athlete is represented as a point consisting of several statistical categories. Obviously, the gamer would like to form a team that cannot be dominated by other teams.

Unlike traditional skyline computation which only checks dominance relationships between *individual* points, the skyline groups computation checks dominance relationships between combinations of k points, where k denotes the size of a group. Since a skyline group may consist of both skyline points and non-skyline points, all points in the data set have a chance to form a skyline group. Therefore, there are total C_n^k combinations, which are far more than the n candidates in traditional skyline computation.

Furthermore, dominance relationships between groups defined in existing works can be divided into two types, which makes skyline groups queries more complicated. In the first type, groups are compared by their permutations of points. If one permutation of a group G can dominated a permutation of another group G', then G dominates G'. In the second type, each group is represented by a point whose attribute values are aggregated over the corresponding attribute values of all points in the group. Then groups are compared by their representative points as conventional skyline. However, these two different definitions have not been compared systematically under the same experimental framework. Hence, it is difficult to select an appropriate definition and corresponding algorithms.

Motivation. To address this problem and provide guidelines on selecting a suitable definition and corresponding algorithms for different situations, we present a comprehensive comparisons on existing skyline groups algorithms. We briefly summarize our contributions as follows:

- We provide a comprehensive survey on all existing skyline groups algorithms.
- We compare existing skyline group algorithms through extensive experiments on synthetic and real data sets.
- We report the strong points and weak points of existing algorithms that can guide practitioners to select appropriate algorithms for different situations.

2 PROBLEM DEFINITION

First, we introduce the dominance relationship between points. We assume that **larger** values are preferred in this paper. Q^i denotes the i^{th} point and Q^i_k denotes the value on the k^{th} dimension of Q^i. For reference, a summary of frequently used notions is given in Table 1.

Table 1: The Summary of Notations

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

Definition 2.1. ($\prec$) Q^i dominates Q^j, denoted as $Q^i \prec Q^j$, *iff* for each k, $Q^i_k \geq Q^j_k$ and for at least one k, $Q^i_k > Q^j_k$ $(1 \leq k \leq d)$.

The skyline of a data set is a set of points that are not dominated by other points in the data set. Fig. 1 shows a skyline example. The data set in Fig. 1 (left) consists of 5 points. As shown in Fig. 1 (right), the skyline contains Q^1, Q^3 and Q^5.

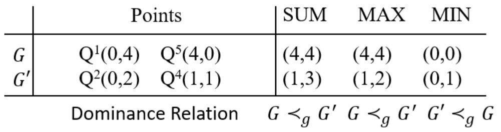
We use $\prec_g$ to denote the dominance relationship between groups. Let $G \prec_g G'$ denote G dominates G'. The dominance relationship between groups defined in existing works [4, 5, 9, 12, 16] can be divided into two types.

Definition 2.2. ($\prec_g$) [12] Assuming that $G = \{Q^1, Q^2, ..., Q^l\}$ and $G' = \{Q'^1, Q'^2, ..., Q'^l\}$ are two different groups with l points. We say that $G \prec_g G'$, *iff* there exist two permutations of the 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, $Q^{ui} \prec Q'^{ui}$ $(1 \leq i \leq l)$.

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

Definition 2.3. ($\prec_g$) [4, 5, 9, 16] For an aggregate function f and a group $G = \{Q^1, Q^2, ..., Q^l\}$, then G is represented by a point Q, where $Q_j = f(Q^1_j, Q^2_j, ..., Q^l_j)$. For two distinct groups G and G', Q and Q' represents G and G' respectively. We define $G \prec_g G'$ *iff* $Q \prec Q'$.

Two kinds of aggregate function are studied in existing works. The first one is strictly monotone, which means $f(Q^1_j, Q^2_j, ..., Q^l_j) > f(Q^{1'}_j, Q^{2'}_j, ..., Q^{l'}_j)$ if $Q^i_j \geq Q^{i'}_j$ for every $i \in [1, l]$ and $\exists k$ such that $Q^k_j > Q^{k'}_j$, where $1 \leq k \leq l$. 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. Fig. 2 shows the dominance relations under different aggregate functions.



	Points		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			$G \prec_g G'$	$G \prec_g G'$	$G' \prec_g G$

Figure 2: Dominance relations under different aggregate functions

Based on the Definition 2.2 or Definition 2.3, skyline group is defined as follows:

Definition 2.4. **(GSkyline)** The l-point GSkyline consists of groups with l points that are not dominated by any other groups of the same size.

Assume that $l = 2$ then the skyline groups based on above definitions are shown in Table 2.

3 COMPUTING GSKYLINE

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

3.1 GSkyline based on Definition 2.2

It is shown in [12] that if a point Q^i is in a skyline group G based on Definition 2.2, all points dominate Q^i must be contained in G. Based on this important conclusion, [12] proposed an algorithm to unite *units* together to get l-point skyline groups. Here, a *unit* of a point contains the point and all points dominate the point.

3.2 GSkyline based on SUM

In order to compute skyline groups based on SUM, existing works [4, 5, 9, 16] adopt similar dynamic programming algorithms. Let Sky^n_l denote the set of l-point skyline groups with regard to $\{Q^1, .., Q^n\}$ and Sky^{n-1}_{l-1} denote the set of $l-1$ point skyline groups with regard to $\{Q^1, ..., Q^{n-1}\}$. Thus we have:

LEMMA 3.1. *Under SUM, given $G \in Sky^n_l$, if $Q^n \in G$, then $G \backslash \{Q^n\} \in Sky^{n-1}_{l-1}$.*

Based on Lemma 3.1, Sky^n_l is computed as follows:

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

Table 2: Skyline groups under different definitions

$\prec_g$	Skyline Groups
Definition 2.2	$\{Q^1,Q^3\},\{Q^1,Q^5\},\{Q^3,Q^4\},\{Q^3,Q^5\}$
SUM	$\{Q^1,Q^3\},\{Q^1,Q^5\},\{Q^3,Q^5\}$
MAX	$\{Q^1,Q^5\}$
MIN	$\{Q^1,Q^2\},\{Q^1,Q^3\},\{Q^2,Q^3\},\{Q^3,Q^4\},\{Q^3,Q^5\}$

3.3 GSkyline based on MAX and MIN

Since many skyline groups based on MAX and MIN share the same aggregate vector, [16] proposed an algorithm to compute distinct aggregate vectors for MAX and MIN. For instance in Table 2, $\{Q^1,Q^2\},\{Q^1,Q^3\}$ and $\{Q^2,Q^3\}$ share the same aggregate vector $(0,2)$ under MIN. The algorithm proposed in [16] is based on the following lemma.

Lemma 3.2. *Under function MAX and MIN, given $G \in Sky_l^n$, if $Q^n \in G$, then there exists a group $G' \in Sky_{l-1}^{n-1}$ such that $F(G' \cup \{Q^n\}) = F(G)$.*

Based on Lemma 3.2, distinct aggregate vectors of skyline groups is computed as follows:

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

When Sky_i is computed, we remove duplicate aggregate vectors in Sky_i then use Sky_i to compute Sky_{i+1}. Thus Sky_l is the set of distinct aggregate vectors of l-point skyline groups.

3.4 Relations between Different Definitions

3.4.1 Relationships between Definition 2.2 and SUM. We find that *GSkyline* of a data set based on SUM is a subset of *GSkyline* based on Definition 2.2.

Lemma 3.3. *GSkyline of a data set based on SUM is a subset of GSkyline based on Definition 2.2.*

Proof. Assume that G is a skyline group based on SUM but G is dominated by a group G' under Definition 2.2, then G' contains at least one point Q' that is better than a point Q in G and the rest points of G' are equal or better than the rest points in G. Thus, $G' \prec_g G$ based on SUM, which contradicts G is a skyline group based on SUM. □

Lemma 3.3 is not only true for SUM but also true for other strictly monotone functions.

3.4.2 Relationships between MAX, MIN and SUM. Based on Lemma 3.2, we can apply Equation (1) to compute aggregate vectors for MAX and MIN. The difference between the results of Equation (1) and Equation (2) is that the result of Equation (1) contains duplicate aggregate skyline vectors. Therefore, the output of Equation (2) is a subset of the output of Equation (1) under MAX and MIN.

3.4.3 Construct Skyline Groups for MAX and MIN. Once we compute the aggregate vectors for skyline groups based on MAX and MIN, We can construct equivalent skyline groups based on an aggregate skyline vector.

For MIN, given a MIN aggregate vector v, the process is to find a set $\Omega(v)$, the set of all points that dominate or are equal to v. Assume that the aggregate vector of arbitrary l points in $\Omega(v)$ is v', we have $v' \not\prec v$, otherwise v is not an aggregate vector of a skyline group. Moreover, v' does not contain smaller values than v on any attribute, by the definition of $\Omega(v)$. Thus we have $v' = v$, which means that any l-point subset of $\Omega(v)$ is a l-point skyline group.

For MAX, it is shown as a NP-hard problem [16]. 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. Obviously, for MAX if $l \geq d$, then there is only one distinct aggregate skyline vector with max values of all attributes. In [16] they proposed a brute-force enumeration method to construct skyline groups.

4 EXPERIMENTAL STUDY

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

DPSG : the dynamic programming algorithm to compute skyline groups based on Equation (1) for SUM.

DASV : the algorithm to compute distinct aggregate skyline vectors based on Equation (2) for MAX and MIN.

Unit-wise+ : the algorithm to compute the skyline groups based on Definition 2.2. Three algorithms *point-wise, unit-wise* and *unit-wise+* are proposed to compute skyline groups based on Definition 2.2, *unit-wise+* is the best among them with respect to time and space.

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

4.1 Study of Different Definitions

4.1.1 Output Analyses. Table 3 and Table 4 show the output sizes of skyline groups and aggregate skyline vectors for correlated synthetic data sets. Obviously, under Definition 2.2, a skyline group does not have an aggregate skyline vector. From the two tables, we can see that G quickly becomes very large under Definition 2.2, SUM and MAX. It should be noted that for MAX, if $l \geq d$, there is only one distinct aggregate skyline vector, but the size of skyline groups can be extremely large. For instance, in Table 4, when $d = 4$ and $l = 6$, it takes 4 points at most to cover the MAX values of all attributes, then the rest 2 points can be arbitrary. Thus there are more than 500 billion skyline groups, which are prohibitive to compute. Moreover, we can see that the output size of skyline groups under SUM is always less than that under Definition 2.2, which coincides with Lemma 3.3. In general Definition 2.2 results in the largest number of skyline groups.

We can see that it is rare for SUM to have two skyline groups share the same aggregate skyline vector, it is because

Table 3: Number of Skyline Groups (G), Distinct Skyline Group Vectors (V), under various n, l, definitions and aggregate functions ($d = 6$)

n		*l=2*				*l=4*				*l=6*			
		SUM	MAX	MIN	*Def.2*	SUM	MAX	MIN	*Def.2*	SUM	MAX	MIN	*Def.2*
200K	G	212	324	101	1747	1461	1129	225	0.5M	5134	6	424	63M
	V	212	235	101		1461	353	225		5134	1	424	
400K	G	258	323	120	2268	2591	752	361	0.9M	11757	10	768	152M
	V	258	284	118		2591	396	357		11757	1	762	
600K	G	248	1584	120	2760	2564	398	369	1.3M	15145	36	742	279M
	V	248	300	120		2564	321	369		15145	1	736	
800K	G	163	302	88	1706	1106	2122	248	0.5M	5036	160	448	83M
	V	163	215	88		1106	196	248		5036	1	448	
1M	G	173	370	89	2004	1244	4318	268	0.7M	5229	450	455	115M
	V	173	228	89		1244	222	268		5229	1	449	

Correlated synthetic data set. M:million, B: billion.

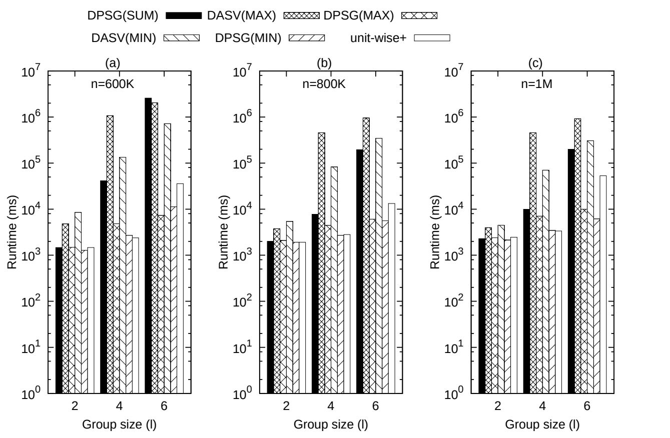


Figure 3: Runtime of different algorithms when varying n and l ($d = 6$)

that the SUM aggregate vector of a skyline group is sensitive to all points in the group. Among the three aggregate functions, MAX results in the most equivalent groups that share the same aggregate skyline vector.

4.1.2 Performance Analyses. We report the runtime of different algorithms to compute skyline groups and distinct aggregate skyline vectors under different settings. From Fig. 3 and Fig. 4, we can see that $DASV(MAX)$ and $DASV$ (MIN) consume the most time among all six algorithms. From Section 3.4.2, we know that we can apply $DPSG(MAX)$ and $DPSG(MIN)$ to compute aggregate skyline vectors for MAX and MIN. We find that when $l \geq 4$, $DPSG(MAX)$ and $DPSG(MIN)$ is about average 92× and 49× speedup over $DASV(MAX)$ and $DASV(MIN)$ respectively. On this other hand, the output sizes of $DPSG(MAX)$ and $DPSG$ (MIN) are small, just a few hundred aggregate vectors. Moreover, the results of $DPSG$ and $DASV$ are close, especially for MIN, which is shown in Table 5.

Therefore, it is more efficient to compute skyline groups under MAX and MIN in the following steps. First apply

Table 4: Number of Skyline Groups (G), Distinct Skyline Group Vectors (V), under various d, l, definitions and aggregate functions ($n = 1M$)

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	1M	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	83155
	V	19	17	9		56	1	12		117	1	16	
5	G	65	118	35	630	245	640	47	75674	591	0.1B	50	4.2M
	V	65	52	35		245	25	47		591	1	50	
6	G	173	370	89	2004	1244	4318	268	0.7M	5229	450	455	115M
	V	173	228	89		1244	222	268		5229	1	449	
7	G	285	700	177	4475	3216	7723	605	3.5M	18541	3912	1276	1.2B
	V	285	537	177		3216	1320	605		18541	99	1276	

Correlated synthetic data set. M:million, B: billion.

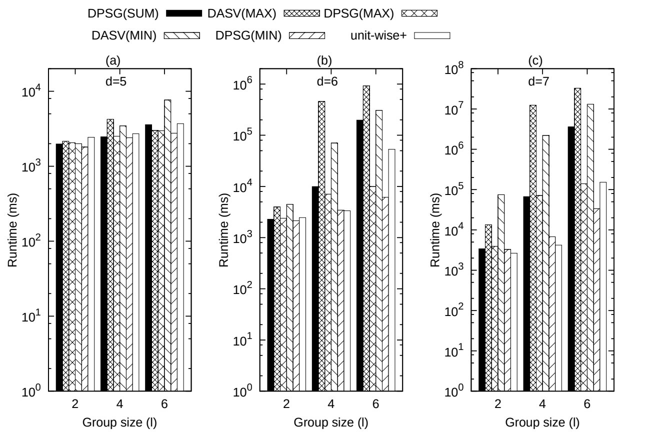


Figure 4: Runtime of different algorithms when varying d and l ($n = 1M$)

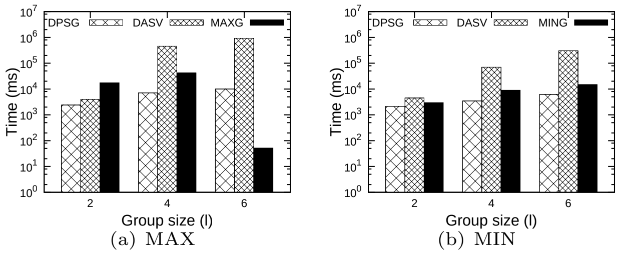
Table 5: Comparing output between DPSG and DASV under MAX and MIN ($d = 6$)

n	*l=2*				*l=4*				*l=6*			
	MAX		MIN		MAX		MIN		MAX		MIN	
	DPSG	DASV	DPSG	DASV	DPSG	DASV	DPSG	DASV	DPSG	DASV	DPSG	DASV
600K	316	300	120	120	454	321	369	369	2	1	737	736
800K	232	215	88	88	348	196	248	248	2	1	448	448
1M	271	228	89	89	729	222	268	268	4	1	449	449

Correlated synthetic data set. M:million, B: billion.

$DPSG$ to compute all aggregate skyline vectors. Second remove duplicate vectors in the above result. Finally, construct skyline groups based on the residual aggregate skyline vectors.

From Fig. 3 and Fig. 4, we find that $DPSG(SUM)$ is most time consuming among *unit-wise+*, $DPSG(SUM)$, $DPSG$ (MAX) and $DPSG(MIN)$. However, the output size of *unit-wise+* is the largest. This indicates that the runtime is not positively related to the output size for existing skyline groups algorithms. We summary characteristics of existing algorithms in Table 6.



(a) MAX (b) MIN

Figure 5: Constructing skyline groups based on aggregate skyline vectors ($n = 1M, d = 6$)

Table 6: Characteristics of existing algorithms

Algorithm	unit-wise+	DPSG			DASV	
Definition	*Def.2*	SUM	MAX	MIN	MAX	MIN
Runtime	short	medial	short	short	long	long
Output	large	medial	small	small	small	small

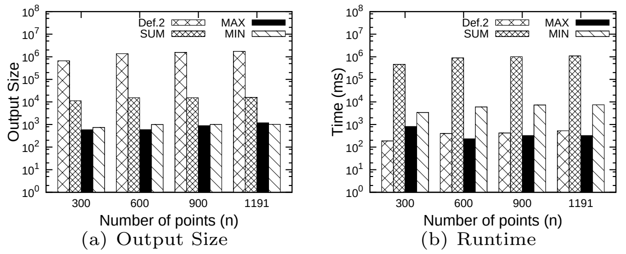
Next, we report the performance of *MAXG* and *MING*. As shown in Fig. 5(a) and Fig. 5(b), $DPSG$ is much more efficient than $DASV$ under MAX and MIN. We find that the runtime of $MAXG$ is not very long though we apply a brute-force method. It is because the number of points that "hit" the MAX attribute values in an aggregate skyline vector is typically small. Thus it is efficient to construct skyline groups under MAX in practice. Fig. 5(b) shows that it is also efficient to construct skyline groups by applying $MING$.

4.2 Experiments on the NBA Data Set

We continue to use the NBA data set adopted in [12]. The data was extracted from http://stats.nba.com/ leaders/alltime /?ls=iref:nba:gnav on 11/01/2016. The data set contains 1191 players and 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).

Since it is shown in Fig. 3 and Fig. 4 that $DPSG$ is much more efficient than $DASV$ to compute aggregate skyline vectors under MAX and MIN, we apply $DPSG$ here. The experimental results are shown in Fig. 6(a) and Fig. 6(b). From Fig. 6(b), we can see that $DPSG(SUM)$ consumes most time among *DPSG(SUM), DPSG(MAX), DPSG(MIN)* and *unit-wise+*, which is same with results shown in Fig. 3 and Fig. 4. Moreover, Fig. 6(a) shows that *unit-wise+* and *DPSG(SUM)* result in the most and the second most skyline groups respectively, which is also same with the experimental results in Table 3 and Table 4.

The above experiments on the NBA data set are coincide with our conclusions drawn in Table 6, which indicates that our conclusions in Table 6 can be used as guidelines to select appropriate algorithms for real-life situations.



(a) Output Size (b) Runtime

Figure 6: Computing skyline groups on the NBA data set ($d = 5, l = 5$)

5 CONCLUSION

We provide a detailed survey on the existing skyline groups algorithms. Based on the experimental results we reveal characteristics of existing algorithms which can be guidelines on selecting appropriate algorithms for various scenarios. We recommend to employ $DPSG$ to compute aggregate skyline vectors under MAX and MIN, because it is much more efficient than $DASV$. For MAX and MIN, we recommend to use aggregate skyline vectors instead of skyline groups, since the number of equivalent groups can be extremely large. If you focus on the performance and do not care about the output size, *unit-wise+* is a good option. If you need to balance needs of performance and output size, we recommend to select the $DPSG(SUM)$, because it is moderate with respect to performance and output size. Since sometimes the output sizes of *unit-wise+* and *DPSG(SUM)* are extremely large, which is a potential limitation of the skyline group operator, we need to design top-k algorithms to help users to make a good and quick selection.

6 ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant No.61502511, No.61572514 and No.61502513.

REFERENCES

[1] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. 2008. Efficient sort-based skyline evaluation. *TODS* 33, 4 (2008), 31.
[2] S Borzsony, Donald Kossmann, and Konrad Stocker. 2001. The skyline operator. In *Proc. ICDE*. IEEE, 421–430.
[3] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. 2003. Skyline with presorting. In *Proc. ICDE*, Vol. 3. 717–719.
[4] Yu-Chi Chung, I-Fang Su, and Chiang Lee. 2013. Efficient computation of combinatorial skyline queries. *Information Systems* 38, 3 (2013), 369–387.
[5] Hyeonseung Im and Sungwoo Park. 2012. Group skyline computation. *Information Sciences* 188 (2012), 151–169.
[6] Donald Kossmann, Frank Ramsak, and Steffen Rost. 2002. Shooting stars in the sky: An online algorithm for skyline queries. In *Proc. VLDB*. VLDB Endowment, 275–286.
[7] Jongwuk Lee and Seung-won Hwang. 2014. Scalable skyline computation using a balanced pivot selection technique. *Information Systems* 39 (2014), 1–21.
[8] Ken CK Lee, Baihua Zheng, Huajing Li, and Wang-Chien Lee. 2007. Approaching the skyline in Z order. In *Proc. VLDB*. VLDB Endowment, 279–290.
[9] Chengkai Li, Nan Zhang, Naeemul Hassan, Sundaresan Rajasekaran, and Gautam Das. 2012. On skyline groups. In *Proc.*

CIKM. ACM, 2119–2123.

[10] Xiaoyong Li, Yijie Wang, Xiaoling Li, Xiaowei Wang, and Jie Yu. 2014. Gdps: an efficient approach for skyline queries over distributed uncertain data. *Big Data Research* 1 (2014), 23–36.

[11] Xiaoyong Li, Yijie Wang, Xiaoling Li, and Yuan Wang. 2014. Parallel skyline queries over uncertain data streams in cloud computing environments. *International Journal of Web and Grid Services* 10, 1 (2014), 24–53.

[12] Jinfei Liu, Li Xiong, Jian Pei, Jun Luo, and Haoyu Zhang. 2015. Finding pareto optimal groups: group-based skyline. *Proc. VLDB* 8, 13 (2015), 2086–2097.

[13] Hua Lu, Christian S Jensen, and Zhenjie Zhang. 2011. Flexible and efficient resolution of skyline query size constraints. *TKDE* 23, 7 (2011), 991–1005.

[14] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. 2007. Probabilistic skylines on uncertain data. In *Proc. VLDB*. VLDB Endowment, 15–26.

[15] Yidong Yuan, Xuemin Lin, Qing Liu, Wei Wang, Jeffrey Xu Yu, and Qing Zhang. 2005. Efficient computation of the skyline cube. In *Proc. VLDB*. VLDB Endowment, 241–252.

[16] Nan Zhang, Chengkai Li, Norfaeza Hassan, Sanguthevar Rajasekaran, and Goutam Das. 2014. On skyline groups. *TKDE* 26, 4 (2014), 942–956.

[17] Peng Zhang, Chuan Zhou, Peng Wang, Byron J Gao, Xingquan Zhu, and Li Guo. 2015. E-tree: An efficient indexing structure for ensemble models on data streams. *TKDE* 27, 2 (2015), 461–474.

[18] Qing Zhang, Pengjie Ye, Xuemin Lin, and Ying Zhang. 2013. Skyline probability over uncertain preferences. In *Proc. EDBT*. ACM, 395–405.

[19] Shiming Zhang, Nikos Mamoulis, and David W Cheung. 2009. Scalable skyline computation using object-based space partitioning. In *Proc. SIGMOD*. ACM, 483–494.